HITACHI
Inspire the Next

ストリームデータ処理基盤

uCosminexus Stream Data Platform - Application Framework システム構築・運用ガイド

手引・文法・操作書

3020-3-V02-20

■ 対象製品

●適用 OS：Windows Server 2008 R2 Standard，Windows Server 2008 R2 Enterprise，Windows Server 2008 R2 Datacenter

P-2964-9B14 uCosminexus Stream Data Platform - Application Framework 01-05※

●適用 OS：Red Hat Enterprise Linux 5，Red Hat Enterprise Linux 6

P-9W64-9B11 uCosminexus Stream Data Platform - Application Framework 01-05

●適用 OS：Red Hat Enterprise Linux 6

P-9W64-9V11 uCosminexus Stream Data Platform - Application Framework 01-50

注※　この製品については，サポート時期をご確認ください。

これらのプログラムプロダクトのほかにもこのマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。

■ 輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

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 は，米国 EMC コーポレーションの RSA BSAFE(R)ソフトウェアを搭載しています。

■ マイクロソフト製品の表記について

このマニュアルでは，マイクロソフト製品の名称を次のように表記しています。

表記		製品名
Windows Server 2008 または Windows	Windows Server 2008 R2 Standard	Microsoft(R) Windows Server(R) 2008 R2 Standard x64
	Windows Server 2008 R2 Enterprise	Microsoft(R) Windows Server(R) 2008 R2 Enterprise x64
	Windows Server 2008 R2 Datacenter	Microsoft(R) Windows Server(R) 2008 R2 Datacenter x64
Internet Explorer		Windows(R) Internet Explorer(R)

■ 発行

2014 年 6 月　3020-3-V02-20

■ 著作権

変更内容

変更内容（3020-3-V02-20）　uCosminexus Stream Data Platform - Application Framework　01-50

追加・変更内容	変更個所
V01-50 以降の場合に設定する環境変数を追加した。	表 3-6
Linux の場合について，Dashboard Server の登録時に必要な設定を追加した。	3.8.2(2)，3.8.2(3)
Linux の V01-50 以降について，Dashboard Server の起動方法を追加した。	4.5.1
Linux の V01-50 以降について，Dashboard Server の停止方法を追加した。	4.5.3
V01-50 より前の場合，および V01-50 以降の場合について，Dashboard Server の登録解除方法を追加・変更した。	5.6.2
サーバ内設定時の定義ファイル（usrconf.properties）の編集方法を追加した。	12.2
Dashboard Server のサーバ内設定ファイルの配置先を追加した。	12.2
画面編集用のファイルの配置先を追加した。	12.3.2(1)

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

このマニュアルは，uCosminexus Stream Data Platform - Application Framework の設計・構築・運用方法，および構築時に設定できる機能の詳細について説明したものです。uCosminexus Stream Data Platform - Application Framework の設計・構築・運用をすることで，ストリームデータの分析ができるようになることを目的としています。

■ 対象読者

uCosminexus Stream Data Platform - Application Framework の設計・構築・運用を実施するシステム管理者の方を対象としています。

また，次の知識をお持ちの方を前提としています。

- OS の基礎的な知識
- Java の基礎的な知識（Java 関連の用語の意味を知っている）
- XML の基礎的な知識

なお，このマニュアルは，マニュアル「uCosminexus Stream Data Platform - Application Framework 解説」を前提としていますので，あらかじめお読みいただくことをお勧めします。

■ 適用 OS の違いによる機能相違点の表記

このマニュアルで説明する機能は，適用 OS の種類（Windows または Linux）によって，異なる場合があります。OS によって機能差がある場合，OS 名を明記しています。

なお，Windows のパスの区切り文字として使用している「¥」は，Linux の場合には，特に断りのないかぎり，「/」に読み替えてください。

■ このマニュアルで使用している記号

このマニュアルで使用している記号について次に示します。

記号	意味
|	横に並べられた複数の項目に対する項目間の区切りを示し，「または」を意味します。 （例）　A | B 　A または B を指定することを示します。
{ }	この記号で囲まれている複数の項目のうちから一つを選択することを示します。項目が横に並べられ，記号 | で区切られている場合は，そのうちの一つを選択します。 （例）　{A | B} 　A または B のどちらかを指定することを示します。
[ ]	この記号で囲まれている項目は省略してもよいことを示します。複数の項目が横に並べて記述されている場合には，すべてを省略するか，記号 { } と同じくどれか一つを選択します。 （例 1）　[A] 　「何も指定しない」か「A を指定する」ことを示します。 （例 2）　[B | C] 　「何も指定しない」か「B または C を指定する」ことを示します。
< >	この記号で囲まれている項目は，該当する要素やファイルなどを指定したり，該当する要素が表示されたりすることを示します。 （例）　<パラメーター名>

記号	意味
＜　＞	パラメーター名を指定します。またはパラメーター名が表示されます。

■ 図中で使用する記号

このマニュアルの図中で使用する記号を次のように定義します。

●ストリームデータの
流れ

目次

第 1 編　お読みいただく前に

第 2 編　設計・構築

第 4 編　リファレンス

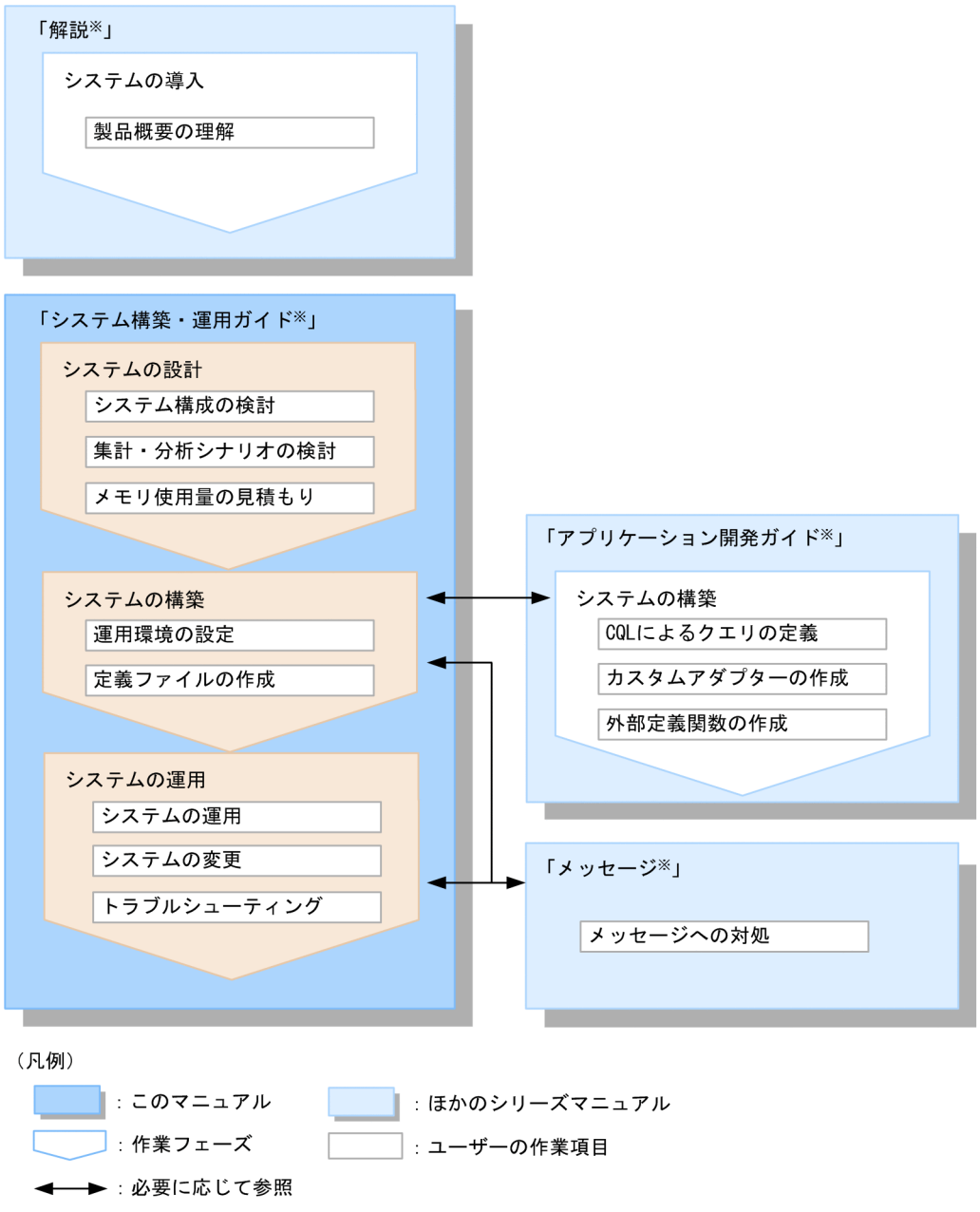
1 Stream Data Platform - AF でのシステム構築・運用の概要

この章では，このマニュアルを読み進める前に知っておいていただきたい，Stream Data Platform - AF の導入から運用までの流れ，およびこのマニュアルの構成について説明します。

1.1　導入から運用までの流れ

Stream Data Platform - AF の導入から運用までの流れと関連マニュアルとの関係を次の図に示します。

図 1-1　導入から運用までの流れと関連マニュアルとの関係



注※　マニュアル名称のうち，「uCosminexus Stream Data Platform - Application Framework」は省略して表記しています。

このマニュアルでは，導入から運用までの流れのうち，「システムの設計」，「システムの構築」，および「システムの運用」について説明します。

なお，このマニュアルを読む前に，マニュアル「uCosminexus Stream Data Platform - Application Framework 解説」を参照して，Stream Data Platform - AF の製品概要を理解しておいてください。

また，各作業フェーズで必要に応じて関連マニュアルを参照してください。例えば，「システムの構築」では，CQL によるクエリの定義とカスタムアダプターなどのアプリケーションの作成について，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。また，「システムの運用」では，トラブルシューティングでメッセージを確認する際に，マニュアル「uCosminexus Stream Data Platform - Application Framework メッセージ」を参照してください。

1.2　このマニュアルの構成

このマニュアルの構成を紹介します。このマニュアルは，次のように四つの編と付録で構成されています。

- 第 1 編：お読みいただく前に
  この編です。このマニュアルを読み進める前に知っておいていただきたい内容として，Stream Data Platform - AF の導入から運用までの流れ，およびこのマニュアルの構成について説明しています。
- 第 2 編：設計・構築
  ストリームデータ処理システムの設計，および構築について説明しています。
- 第 3 編：運用
  ストリームデータ処理システムの運用，変更，およびトラブルシューティングについて説明しています。
- 第 4 編：リファレンス
  コマンドや定義ファイルの詳細について説明しています。
- 付録
  このマニュアルの参考情報について説明しています。

それぞれの編に含まれる章と付録の記載内容について，次の表で説明します。

表 1-1　各章と付録の記載内容

編	章・付録	記載内容
第 1 編 お読みいただく前に	第 1 章　Stream Data Platform - AF でのシステム構築・運用の概要	この章です。Stream Data Platform - AF の導入から運用までの流れ，およびこのマニュアルの構成について説明しています。
第 2 編 設計・構築	第 2 章　システムの設計	システムの設計の流れと，設計工程の中で検討する項目について説明しています。
	第 3 章　システムの構築	システムの構築の流れ，Stream Data Platform - AF のディレクトリ構成，およびシステムの構築に必要な設定について説明しています。
第 3 編 運用	第 4 章　システムの運用	構築したシステムの運用方法について説明しています。
	第 5 章　システムの変更	システムの運用中にシステムの設定や構成を変更する手順について説明しています。
	第 6 章　トラブルシューティング	システムの運用中にトラブルが発生した場合に，採取する資料の種類や見方，およびトラブルへの対処方法について説明しています。
第 4 編 リファレンス	第 7 章　コマンド	システムを運用するためのコマンドについて説明しています。
	第 8 章　SDP サーバ用定義ファイル	SDP サーバ用定義ファイルについて説明しています。
	第 9 章　アダプター用定義ファイル	アダプター用定義ファイルについて説明しています。

編	章・付録	記載内容
第 4 編 リファレンス	第 10 章　外部定義関数定義ファイル	外部定義関数定義ファイルについて説明しています。
	第 11 章　定義ファイルでの定義内容の詳細	SDP サーバ用定義ファイル，およびアダプター用定義ファイルでの定義内容の詳細について説明しています。
	第 12 章　Flex Dashboard の設定内容の詳細	Dashboard Server のサーバ内設定ファイル，Dashboard Viewer の画面編集用ファイルの詳細について説明しています。
付録	付録 A　このマニュアルの参考情報	このマニュアルを読むに当たっての参考情報について説明しています。

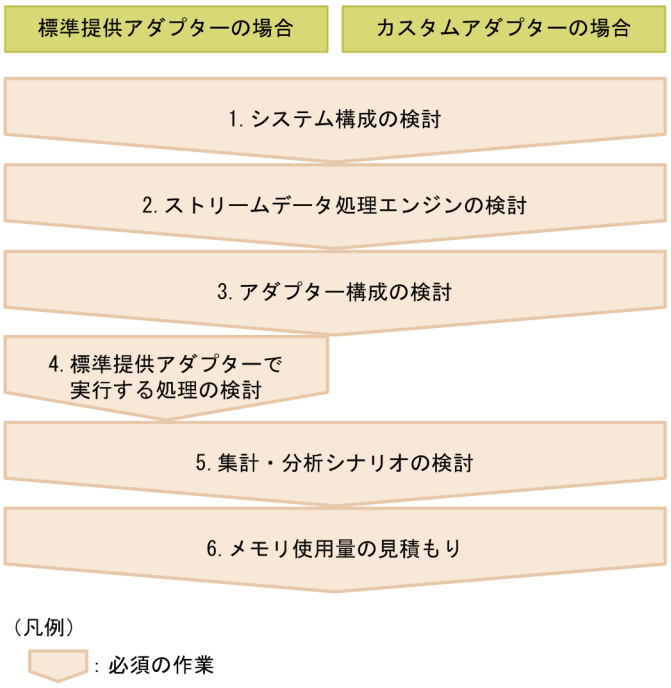
2 システムの設計

この章では，ストリームデータ処理システムの設計の流れと，設計工程の中で検討する項目について説明します。

2.1　設計の流れ

ストリームデータ処理システムの設計に必要な作業は，使用するアダプターの種類によって異なります。システムの設計の流れを次の図に示します。

図 2-1　システムの設計の流れ



それぞれの作業について，次に説明します。

1. **システム構成の検討**

   ストリームデータ処理システムの目的に応じて，プログラムの構成やシステムの構成要素の配置のしかたについて検討します。検討内容については，「2.2　システム構成の検討」を参照してください。

2. **ストリームデータ処理エンジンの検討**

   ストリームデータ処理システムで動作させる SDP サーバの数や，SDP サーバで使用する機能について検討します。検討内容については，「2.3　ストリームデータ処理エンジンの検討」を参照してください。

3. **アダプター構成の検討**

   ストリームデータ処理システムで動作させるアダプターの種類，作成するアダプターの数やアダプターで使用する機能について検討します。検討内容については，「2.4　アダプター構成の検討」を参照してください。

4. **標準提供アダプターで実行する処理の検討**

   標準提供アダプターを使用する場合に，データの入力，データの編集，およびデータの出力の各処理でどのような処理を実行させるかを検討します。検討内容については，「2.5　標準提供アダプターで実行する処理の検討」を参照してください。

5. **集計・分析シナリオの検討**

   ストリームデータを集計・分析するためのシナリオについて検討します。検討内容については，「2.6 集計・分析シナリオの検討」を参照してください。

6. **メモリ使用量の見積もり**

ストリームデータ処理システムに関するメモリ使用量を見積もります。見積もり方法については，「2.7　メモリ使用量の見積もり」を参照してください。

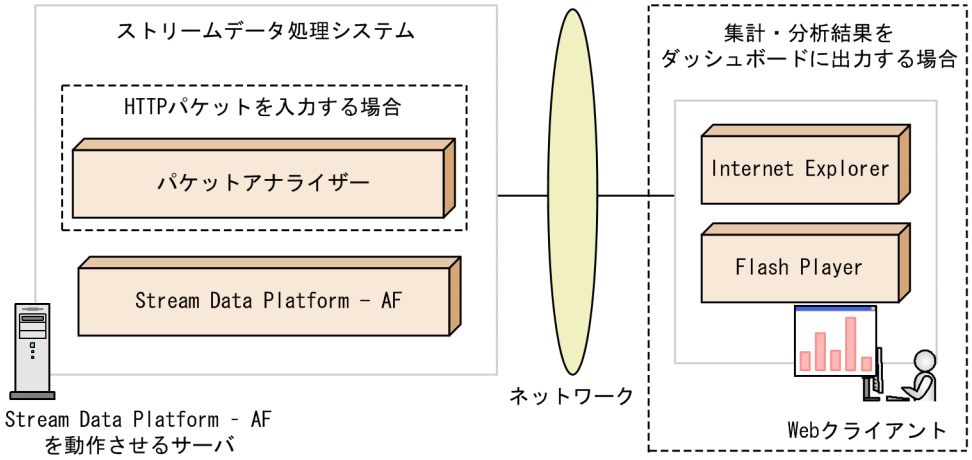
2.2　システム構成の検討

ここでは，ストリームデータ処理システムのプログラムの構成，構成要素，およびアダプターの種類ごとの構成について説明します。

2.2.1　ストリームデータ処理システムのプログラムの構成

ストリームデータ処理システムに必要なプログラムは，使用するアダプターの種類やアダプターで使用する機能によって異なります。ストリームデータ処理システムのプログラムの構成を次の図に示します。

図 2-2　ストリームデータ処理システムのプログラムの構成



基本的なシステム構成の場合，Stream Data Platform - AF を動作させるサーバに必要なプログラムは，Stream Data Platform - AF だけです。

ただし，次の場合には，前提プログラムも必要です。

- **標準提供アダプターを使用して HTTP パケットを入力する場合**

  Stream Data Platform - AF を動作させるサーバに，パケットアナライザーが必要です。Stream Data Platform - AF で使用できるパケットアナライザーについては，「11.3.1　HTTP パケットの入力の概要」を参照してください。

- **標準提供アダプターを使用して集計・分析結果をダッシュボードに出力する場合**

  ダッシュボードに出力したストリームデータの集計・分析結果を Web ブラウザで参照するためには，Web クライアントに，次のプログラムが必要です。

  - Internet Explorer
  - Flash Player

  プログラムのバージョンについては，Stream Data Platform - AF のリリースノートを参照してください。

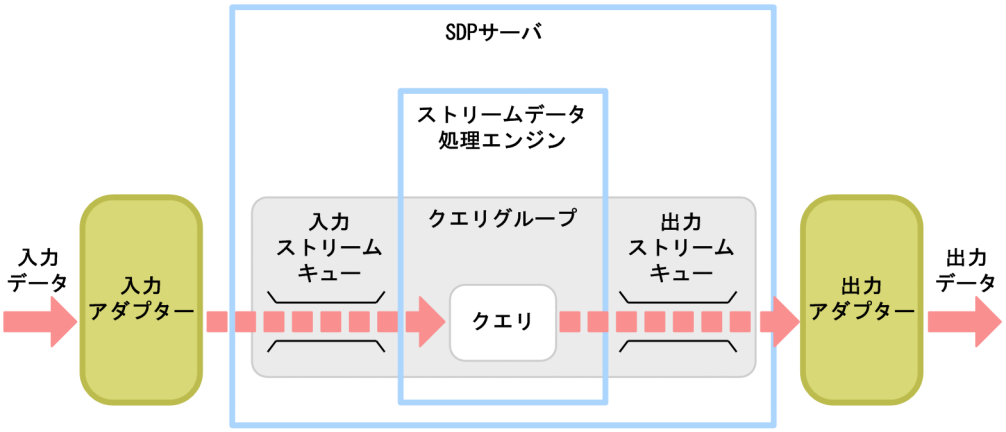
2.2.2　ストリームデータ処理システムの構成要素

ここでは，ストリームデータ処理システムの構成要素について説明します。また，アダプターと SDP サーバとの連携形態，およびストリームデータ処理システムに配置できる構成要素の数についても説明します。

(1)　ストリームデータ処理システムの構成要素

ストリームデータ処理システムの構成要素を次の図に示します。

図 2-3　ストリームデータ処理システムの構成要素



図中に示した構成要素について説明します。

- **アダプター（入力アダプター・出力アダプター）**

  入力アダプターは，入力するデータをストリームデータ処理エンジンで処理できる形式に変換し，ストリームデータ処理エンジンに送信します。出力アダプターは，ストリームデータ処理エンジンで処理されたデータを受信し，指定された形式に変換して出力します。

  アダプターには，Stream Data Platform - AF が標準提供しているアダプター（**標準提供アダプター**）と，ユーザーが Java でプログラミングして作成するアダプター（**カスタムアダプター**）があります。

  なお，標準提供アダプターでは，アダプターグループというまとまりでアダプターを運用します。アダプターグループについては，「2.2.3(1)　アダプターグループの構成」を参照してください。

  また，カスタムアダプターの場合は，入力アダプターのことを**データ送信 AP**，出力アダプターのことを**データ受信 AP** といいます。

- **ストリームデータ処理エンジン**

  ストリームデータ処理エンジンは，入力アダプターから送信されてきたデータをあらかじめ登録されているクエリに従って処理します。処理したデータは，出力アダプターに送信します。

  ストリームデータ処理エンジン上で動作し，ストリームデータを処理するサーバプロセスのことを **SDP サーバ**といいます。アダプターと SDP サーバは，同一プロセス，または別プロセスで動作します。上に示した図中では，アダプターと SDP サーバは別プロセスで動作しています。アダプターと SDP サーバの連携形態については，「(2)　アダプターと SDP サーバとの連携形態」を参照してください。

- **クエリグループ**

  クエリグループは，ストリームデータの集計・分析シナリオです。ストリームデータ処理システムでは，どのようなデータを入力して，どのように処理して出力するかという，ストリームデータの集計・分析シナリオをクエリグループとして定義します。

  クエリグループは，次の要素で構成されます。

  - **入力ストリームキュー（入力ストリーム）**
  - **クエリ**
  - **出力ストリームキュー（出力ストリーム）**

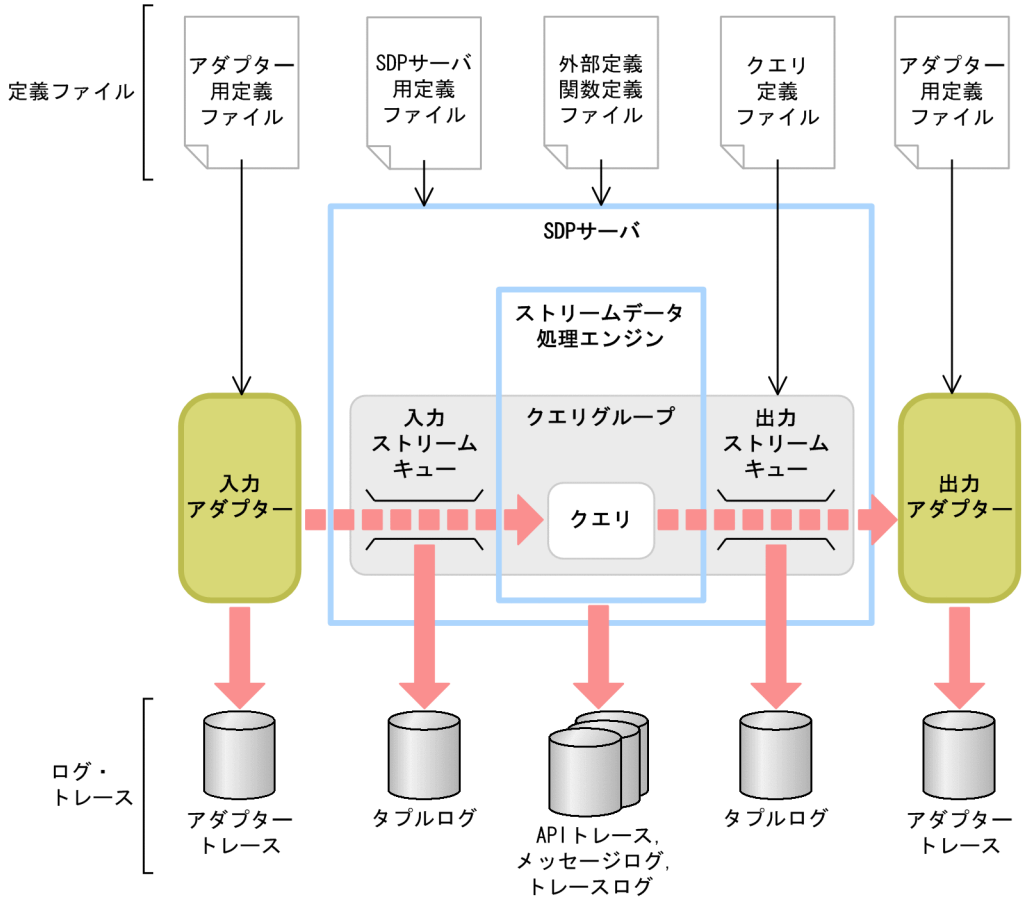
  クエリは，ストリームデータに対してどのような処理を実行するかを指定した定義です。ストリームデータは，入力ストリームキューを通過して，クエリに入力されます。また，クエリでの処理結果は，

ストリームデータに変換されたあと，出力ストリームキューを通過して出力されます。ストリームキューとストリームは，1 対 1 に対応します。

クエリグループについては，「2.6　集計・分析シナリオの検討」を参照してください。

アダプターと SDP サーバの動作は，定義ファイルで設定します。また，アダプターと SDP サーバからは，ログやトレースなどのファイルが出力されます。ストリームデータ処理システムの構成要素と各ファイルの関係を次の図に示します。

図 2-4　ストリームデータ処理システムの構成要素と各ファイルの関係



図中に示したファイルについて説明します。

- **定義ファイル**

  定義ファイルの種類を次に示します。

  - **SDP サーバ用定義ファイル**

    SDP サーバの動作を定義するファイルです。SDP サーバ用定義ファイルについては，「3.4　SDP サーバ用定義ファイルの作成」，および「8.　SDP サーバ用定義ファイル」を参照してください。

  - **クエリ定義ファイル**

    ストリームデータの集計・分析シナリオを定義するファイルです。クエリ定義ファイルについては，「3.5　クエリ定義ファイルの作成」を参照してください。

  - **アダプター用定義ファイル**

標準提供アダプターの動作を定義するファイルです。アダプター用定義ファイルについては，「3.6　アダプター用定義ファイルの作成」，および「9.　アダプター用定義ファイル」を参照してください。

- **外部定義関数定義ファイル**

  外部定義関数を定義するファイルです。外部定義関数定義ファイルについては，「3.7　外部定義関数定義ファイルの作成」，および「10.　外部定義関数定義ファイル」を参照してください。

- **ログ・トレース**

  アダプターの処理状況を出力するアダプタートレース，入力タプルおよび出力タプルの情報を出力するタプルログ，クエリの入り口情報および出口情報を出力する API トレースなどがあります。各ファイルの詳細については，「6.2　トラブル発生時に採取が必要な資料」を参照してください。

(2)　アダプターと SDP サーバとの連携形態

アダプターと SDP サーバは，同一プロセス，または別プロセスで動作します。アダプターと SDP サーバを同一プロセスで動作させる連携形態のことを**インプロセス連携**，別プロセスで動作させる連携形態のことを **RMI 連携**といいます。

それぞれの連携形態のメリットとデメリットを次の表に示します。

表 2-1　アダプターと SDP サーバの連携形態のメリットとデメリット

項番	連携形態	メリット	デメリット
1	インプロセス連携	SDP サーバとアダプターの通信コストが小さく，タプルの処理レイテンシを最も小さくできます。	1 プロセス上で動作するため，メモリの使用量に注意が必要です。
2	RMI 連携	アダプターで障害が発生したときに，ストリームデータ処理エンジンへの影響を最小限にできます。また，アダプターに多くのメモリを割り当てられます。	プロセス間通信を行うため，インプロセス連携の場合と比較すると，通信コストが掛かります。

それぞれの連携形態のメリットとデメリットを考慮した上で，SDP サーバとアダプターをどのように配置するかを決定してください。

連携形態ごとのシステムの構成は，標準提供アダプターの場合とカスタムアダプターの場合で異なります。それぞれのシステムの構成については，「2.2.3　標準提供アダプター使用時の構成」，および「2.2.4　カスタムアダプター使用時の構成」を参照してください。

(3)　ストリームデータ処理システムに配置できる構成要素の数

SDP サーバで起動できるアダプターの数や，アダプターグループで管理できるアダプターの数には，制限があります。ストリームデータ処理システムに配置できる構成要素の数は，標準提供アダプターの場合とカスタムアダプターの場合で異なります。ストリームデータ処理システムに配置できる構成要素の数を次の表に示します。

表 2-2　ストリームデータ処理システムに配置できる構成要素の数

項番	項目	標準提供アダプターの場合の最大値	カスタムアダプターの場合の最大値
1	一つの運用ディレクトリ※1 で起動できる SDP サーバの数	1	1
2	一つの SDP サーバで起動できる RMI 連携アダプターの数	−	32※2

項番	項目	標準提供アダプターの場合の最大値	カスタムアダプターの場合の最大値
3	一つの SDP サーバに登録できるインプロセス連携アダプターの数	－	16
4	一つの SDP サーバで実行できるアダプターグループの数	1	－
5	一つのアダプターグループで管理できるアダプターの数	64※2	－
6	一つの SDP サーバで実行できるクエリグループの数	8	8
7	一つの SDP サーバに登録できるストリームの数	合計で 1,024※3	合計で 1,024※3
8	一つの SDP サーバに登録できるクエリの数		
9	一つの SDP サーバに登録できる外部定義関数の数	1,024	1,024

（凡例）

－：該当しません。

注※1

運用ディレクトリとは，ストリームデータ処理システムの運用で使用するディレクトリのことです。運用ディレクトリの構成については，「2.6.2　クエリグループを実行する運用ディレクトリの構成の検討」を参照してください。

注※2

最大値は目安です。アダプターの数は，Stream Data Platform - AF が動作するサーバの処理性能や，ストリームデータ処理システムでの処理性能に応じて決定してください。

注※3

ストリームの数とクエリの数の合計で，1,024 以下となるように設定してください。

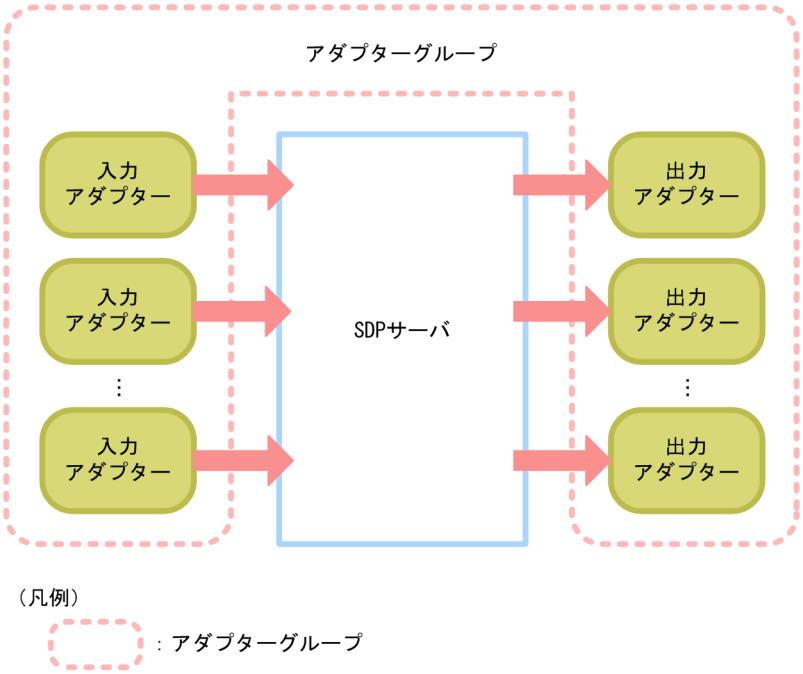
2.2.3　標準提供アダプター使用時の構成

ここでは，標準提供アダプターを使用する場合のアダプターグループの構成，標準提供アダプターと SDP サーバの連携形態について説明します。また，標準提供アダプターとストリームとの関係についても説明します。

(1)　アダプターグループの構成

標準提供アダプターでは，アダプターの集合のことを**アダプターグループ**といいます。アダプターグループの構成例を次の図に示します。

図 2-5　アダプターグループの構成例



一つの SDP サーバで実行できるアダプターグループは，一つだけです。標準提供アダプターは，アダプターグループ単位で運用します。

一つのアダプターグループでは，入力アダプターと出力アダプターをそれぞれ 64 個まで管理できます。アダプターグループ内では，アダプターごとにスレッドが割り当てられ，すべてのアダプターが 1 プロセスで動作します。

(2)　標準提供アダプターと SDP サーバとの連携形態

「2.2.2(2)　アダプターと SDP サーバとの連携形態」で説明したように，アダプターと SDP サーバの連携形態には，インプロセス連携と RMI 連携があります。

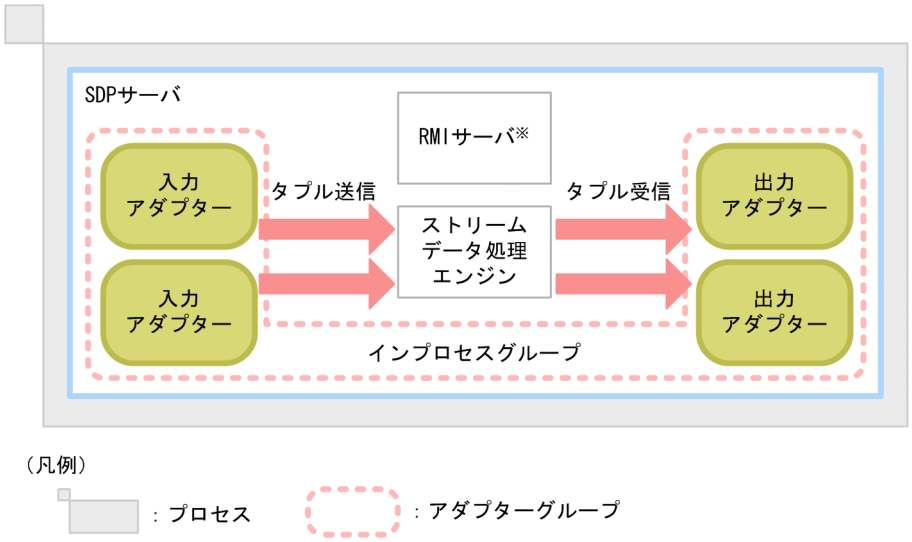
標準提供アダプターの場合，アダプターグループ単位で SDP サーバと連携します。インプロセス連携をするアダプターグループのことを**インプロセスグループ**，RMI 連携をするアダプターグループのことを **RMI グループ**といいます。

なお，一つのアダプターグループ内では，インプロセス連携をするアダプターと RMI 連携をするアダプターの混在はできません。例えば，入力アダプターを RMI グループ，出力アダプターをインプロセスグループという構成はできません。

インプロセスグループの構成と RMI グループの構成を次に示します。

- **インプロセスグループの構成**
  インプロセス連携では，アダプターと SDP サーバが同一プロセス上で動作し，プロセス内でタプルを送受信します。インプロセスグループの構成例を次の図に示します。

図 2-6　インプロセスグループの構成例

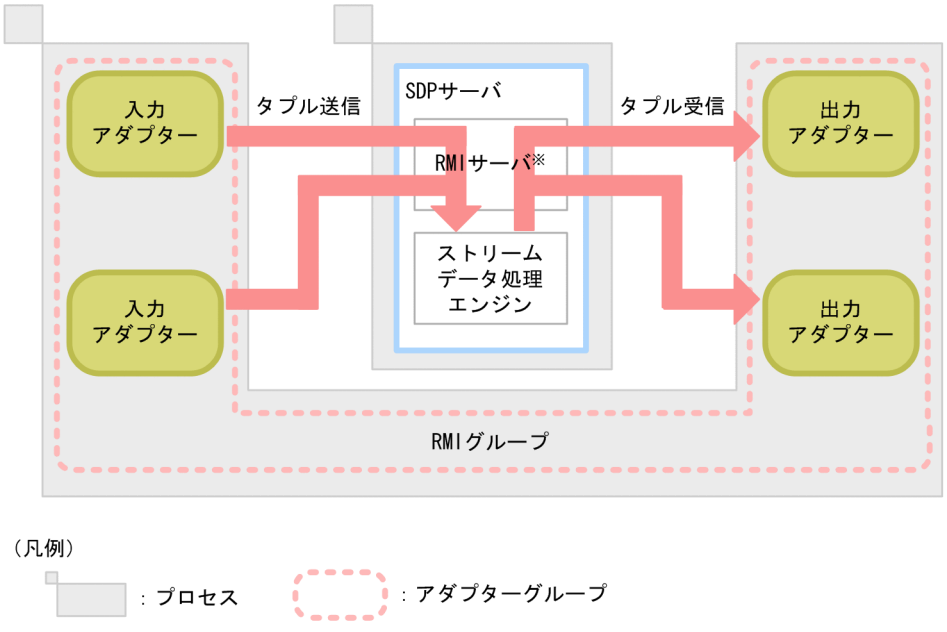


注※　RMIサーバは，RMI通信の処理を実行するサーバです。RMIサーバは，SDPサーバ上で動作します。インプロセスグループでは，RMIサーバを使用しません。

- RMI グループの構成

RMI 連携では，アダプターと SDP サーバが別プロセス上で動作し，Java の RMI を使用してプロセス間でタプルを送受信します。RMI グループの構成例を次の図に示します。

図 2-7　RMI グループの構成例

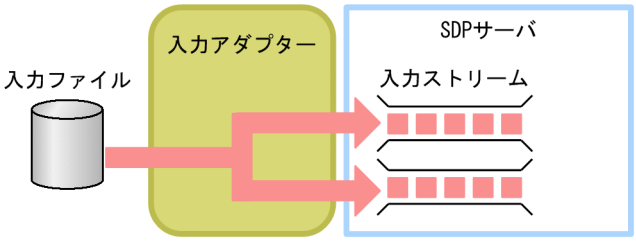


注※　RMIサーバは，RMI通信の処理を実行するサーバです。RMIサーバは，SDPサーバ上で動作します。

(3)　標準提供アダプターとストリームキュー（ストリーム）との関係

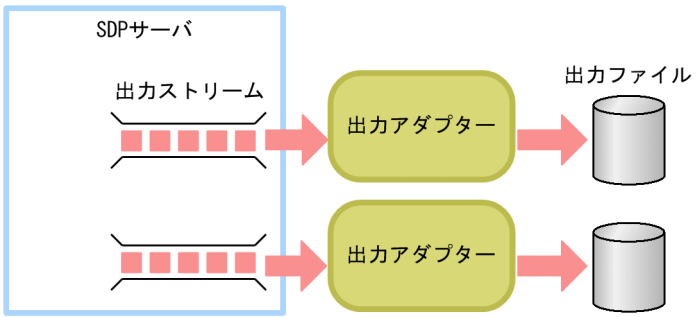
標準提供アダプターの場合，入力アダプターと入力ストリームキュー（入力ストリーム）の関係は，1：n です。一つの入力アダプターでは，複数の入力ストリームキューに対してストリームデータを送信できます。入力アダプターと入力ストリームとの関係を次の図に示します。

図 2-8　入力アダプターと入力ストリームとの関係



これに対して，出力アダプターと出力ストリームキュー（出力ストリーム）の関係は，1：1 です。一つの出力アダプターでは，一つの出力ストリームキューからストリームデータを受信します。このため，出力ストリームが複数ある場合には，同じ数だけ出力アダプターを作成する必要があります。出力アダプターと出力ストリームとの関係を次の図に示します。

図 2-9　出力アダプターと出力ストリームとの関係



2.2.4　カスタムアダプター使用時の構成

「2.2.2(2)　アダプターと SDP サーバとの連携形態」で説明したように，アダプターと SDP サーバの連携形態には，インプロセス連携と RMI 連携があります。カスタムアダプターを使用する場合には，インプロセス連携と RMI 連携を混在させることもできます。ストリームデータ処理システムの規模や目的に応じて，SDP サーバとアダプターをどのように配置するかを決定してください。それぞれの連携形態の場合のシステムの構成については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

2.3　ストリームデータ処理エンジンの検討

ここでは，ストリームデータ処理エンジンに関する検討項目として，SDP サーバの数や，SDP サーバで使用する機能の検討について説明します。

2.3.1　SDP サーバの数の検討

一つの運用ディレクトリで起動できる SDP サーバは，一つだけです。運用ディレクトリが複数ある場合，運用ディレクトリの数だけ SDP サーバを起動する必要があります。

標準提供アダプターを使用する場合，アダプターグループと SDP サーバの数が 1:1 になるようにしてください。

2.3.2　SDP サーバの時刻制御方式の検討

次に示すタイムスタンプモードのうち，どちらのモードでタプルに時刻を設定するかを検討してください。

- サーバモード
  タプルが SDP サーバに到達した時点の時刻をタプルに設定します。
- データソースモード
  データの発生した時点の時刻をタプルに設定します。

タイムスタンプモードについては，マニュアル「uCosminexus Stream Data Platform - Application Framework 解説」を参照してください。

タイムスタンプモードは，SDP サーバ用定義ファイルのシステムコンフィグプロパティファイル（system_config.properties），またはクエリグループ用プロパティファイルの stream.timestampMode パラメーターで指定します。詳細については，「8.6　システムコンフィグプロパティファイル（system_config.properties）」，または「8.7　クエリグループ用プロパティファイル」を参照してください。

2.3.3　タプルのタイムスタンプの調整の検討

SDP サーバの時刻制御方式にデータソースモードを使用している場合は，入力ストリームごとにタプルの時刻を昇順にする必要があります。しかし，複数の入力アダプターからタプルを送信する場合などに，タプルの時刻とタプルの到着の順序に誤差が生じて，時刻が逆転して到着したタプルが SDP サーバに破棄されることがあります。例えば，SDP サーバに到着したタプルの時刻が「09:00:04」→「09:00:05」→「09:00:02」の順番だった場合，時刻が逆転している「09:00:02」のタプルは破棄されます。

このような場合に，タイムスタンプ調整機能を使用して，タプルの到着時刻を調整する時間を設定しておくことで，時刻が逆転したタプルも SDP サーバで受け取れるようになります。例えば，上記の例だと，タプルの到着時刻を調整する時間を 5 秒に設定していれば，「09:00:05」のあとに到着した「09:00:02」のタプルは破棄されません。

タイムスタンプ調整機能の仕組みと，機能の設定方法については，「11.8　タプルのタイムスタンプの調整」を参照してください。

2.4　アダプター構成の検討

ここでは，アダプター構成に関する検討項目として，使用するアダプターの検討，アダプターの種類ごとの検討項目について説明します。

2.4.1　使用するアダプターの検討

ストリームデータを送受信するためのアダプターとして，標準提供アダプターとカスタムアダプターのどちらのアダプターを使用するかを検討します。

標準提供アダプターは，Stream Data Platform - AF が標準提供しているアダプターです。ファイルや HTTP パケットからのデータの入力，マッピングやレコードの抽出によるデータの編集，ファイルやダッシュボードへのデータの出力などの機能をサポートしています。

これに対して，カスタムアダプターは，ユーザーが Java でプログラミングして作成します。標準提供アダプターでサポートしていない機能を使用したい場合や，アダプターで特別な処理をさせたい場合などに，カスタムアダプターを作成してください。

ストリームデータの集計・分析の目的に応じて，標準提供アダプターを使用するか，またはカスタムアダプターを使用するかを決定してください。

2.4.2　標準提供アダプターを使用する場合の検討項目

標準提供アダプターを使用する場合の検討項目を次に示します。

- **作成するアダプターの数**

  アダプターは，データの入力元や出力先の数だけ作成します。「2.2.2(3)　ストリームデータ処理システムに配置できる構成要素の数」で示したとおり，一つのアダプターグループでは，入力アダプターと出力アダプターをそれぞれ 64 個まで管理できます。

  アダプターとストリームの数については，「2.2.3(3)　標準提供アダプターとストリームキュー（ストリーム）との関係」を参照してください。

- **アダプターで実行する処理**

  アダプターで実行する処理の検討については，「2.5　標準提供アダプターで実行する処理の検討」を参照してください。

- **作成するアダプターグループの数**

  アダプターグループと SDP サーバの数が 1:1 になるようにします。

- **アダプターと SDP サーバの連携形態**

  アダプターと SDP サーバをインプロセス連携と RMI 連携のどちらの形態で連携させるかを検討します。連携形態については，「2.2.3(2)　標準提供アダプターと SDP サーバとの連携形態」を参照してください。

標準提供アダプターの詳細な設定は，アダプター用定義ファイルのアダプター構成定義ファイルで設定します。アダプター構成定義ファイルについては，「3.6　アダプター用定義ファイルの作成」，および「9.　アダプター用定義ファイル」を参照してください。

2.4.3　カスタムアダプターを使用する場合の検討項目

カスタムアダプターを使用する場合の検討項目を次に示します。

- **作成するアダプターの数**

  アダプターは，データの入力元や出力先の数だけ作成します。作成できるアダプターの数については，「2.2.2(3)　ストリームデータ処理システムに配置できる構成要素の数」を参照してください。Stream Data Platform - AF が動作するサーバの処理性能や，ストリームデータ処理システムでの処理性能に応じて，作成するアダプターの数を決定してください。

- **アダプターで実行する処理**

  ストリームデータの入出力方法など，アダプターで実行する処理について検討してください。その上で，Stream Data Platform - AF が提供している Java の API などを使用して，それらの機能をどう実現するかを検討してください。

- **アダプターと SDP サーバの連携形態**

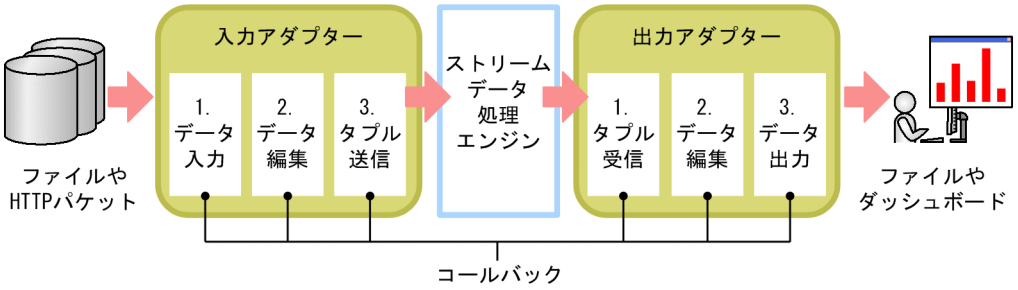
  アダプターと SDP サーバをインプロセス連携と RMI 連携のどちらの形態で連携させるかを検討します。連携形態については，「2.2.4　カスタムアダプター使用時の構成」を参照してください。

2.5　標準提供アダプターで実行する処理の検討

ここでは，標準提供アダプターで実行する処理の検討について説明します。

標準提供アダプターで実行する処理には，データの入出力，データの編集，およびタプルの送受信があります。標準提供アダプターで実行する処理の流れを次の図に示します。

図 2-10　標準提供アダプターで実行する処理の流れ



入力アダプターで実行する処理

1. ファイルや HTTP パケットからのデータ入力
2. マッピングやレコードの抽出などのデータ編集
3. ストリームデータ処理エンジンへのタプル送信

出力アダプターで実行する処理

1. ストリームデータ処理エンジンからのタプル受信
2. マッピングやフィルタリングなどのデータ編集
3. ファイルやダッシュボードへのデータ出力

標準提供アダプターでは，データの入出力，データの編集，およびタプルの送受信の各処理のことを**コールバック**といいます。コールバックの処理については，アダプター用定義ファイルのアダプター構成定義ファイルの各 CB 定義で設定します。

標準提供アダプターで実行するコールバックの処理のうち，データの入出力と編集については，どのような処理を実行させるかを検討しておく必要があります。ストリームデータの集計・分析の目的に応じて，どのようにデータを入出力，および編集するかを検討してください。

2.5.1　データの入力方法の検討

標準提供アダプターが提供するデータの入力方法の中から，どの方法でデータを入力するかを検討します。

データの入力は，入力アダプターで実行する処理です。標準提供アダプターでは，データ入力用のコネクター（入力コネクター）を使用して，ファイルや HTTP パケットのデータを入力できます。データの入力方法を次の表に示します。

表 2-3　データの入力方法

項番	データの入力方法	説明
1	ファイルの入力	**ファイル入力コネクター**を使用して，ストリームデータの集計・分析用のデータとして，エラーログやアクセスログなどのファイルを入力できます。

項番	データの入力方法	説明
1	ファイルの入力	ファイルの入力については，「11.2　ファイルの入力」を参照してください。
2	HTTP パケットの入力	**HTTP パケット入力コネクター**を使用して，ストリームデータの集計・分析用のデータとして，HTTP パケットを入力できます。 HTTP パケットの入力では，HTTP パケット入力コネクターを使用して，パケットアナライザーから出力された HTTP パケットを入力します。 HTTP パケットの入力については，「11.3　HTTP パケットの入力」を参照してください。

なお，データの入力については，アダプター構成定義ファイルの入力用 CB 定義で設定します。アダプター構成定義ファイルの CB 定義については，「9.　アダプター用定義ファイル」を参照してください。

2.5.2　データの編集方法の検討

標準提供アダプターが提供するデータの編集方法の中から，どの方法でデータを編集するかを検討します。

データの編集は，入力アダプターまたは出力アダプターで実行する処理です。標準提供アダプターでは，マッピングやレコードの抽出などのデータの編集ができます。データの編集方法を次の表に示します。

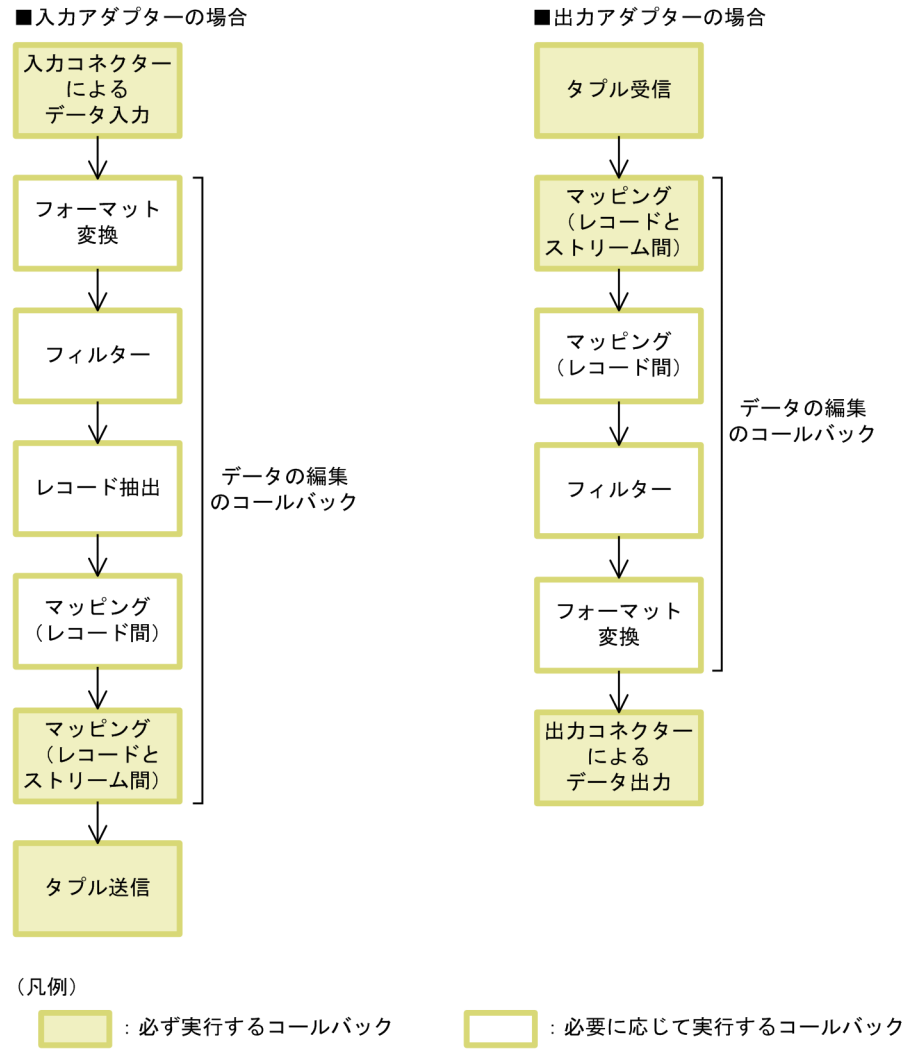
表 2-4　データの編集方法

項番	データの編集方法	説明
1	フォーマット変換	**フォーマット変換**を使用して，入力形式レコード，または出力形式レコードと，共通形式レコードを相互変換できます。 • 入力形式レコード，出力形式レコード： コンマで区切られた CSV ファイルなど，行単位のデータのことです。 • 共通形式レコード： 名前（フィールド名）と値（フィールド値）で構成されるデータの集合のことです。 入力アダプターのフォーマット変換では，入力形式レコードを共通形式レコードに変換します。出力アダプターのフォーマット変換では，共通形式レコードを出力形式レコードに変換します。それぞれのデータ形式については，「11.2.2(1)　入力アダプターで扱うデータ形式」，および「11.6.2(1)　出力アダプターで扱うデータ形式」を参照してください。 なお，フォーマット変換は，次の場合だけ実行する処理です。 • データの入力で，ファイルの入力を使用する場合 • データの出力で，ファイルへの出力を使用する場合 フォーマット変換については，「11.2.2(3)　フォーマット変換」または「11.6.2(4)　フォーマット変換」を参照してください。
2	マッピング	**マッピング**を使用して，マッピングの前のコールバックで出力される共通形式レコードと，マッピングの次のコールバックで入力される共通形式レコードとの関連づけができます。 マッピングには，レコード間のマッピング，およびレコードとストリーム間のマッピングがあります。

項番	データの編集方法	説明
2	マッピング	例えば，レコードとストリーム間のマッピングでは，フォーマット変換で入出力される共通形式レコードと，入力ストリームまたは出力ストリームの形式に従った共通形式レコードとを関連づけます。 マッピングについては，「11.2.2(4)　マッピング」，または「11.6.2(3)　マッピング」を参照してください。
3	レコードのフィルタリング	**フィルター**を使用して，フィルター対象のレコードを取捨選択できます。レコードのフィルタリングの対象は，共通形式レコードです。 レコードのフィルタリングについては，「11.4　レコードのフィルタリング」を参照してください。
4	レコードの抽出	**レコード抽出**を使用して，条件に一致するレコードを抽出したあと，抽出したレコードを結合して，新たなレコードを生成できます。レコードの抽出の対象は，共通形式レコードです。 なお，レコードの抽出は，入力アダプターの場合だけ使用できます。 レコードの抽出については，「11.5　レコードの抽出」を参照してください。

データの編集の処理は，次の図に示すタイミングで実行します。

図 2-11　データ編集の処理を実行するタイミング



フォーマット変換を実行するタイミング

入力アダプターの場合は，入力コネクターによるデータ入力のあと，出力アダプターの場合は，出力コネクターによるデータ出力の前に実行します。

マッピングを実行するタイミング※

レコードとストリーム間のマッピングは，入力アダプターの場合は，タプル送信の前，出力アダプターの場合は，タプル受信のあとに実行します。

レコード間のマッピングは，入力アダプターの場合は，入力コネクターによるデータ入力またはフォーマット変換のあと，出力アダプターの場合は，フォーマット変換または出力コネクターによるデータ出力の前に実行します。

フィルターを実行するタイミング※

入力アダプターの場合は，入力コネクターによるデータ入力またはフォーマット変換のあと，出力アダプターの場合は，フォーマット変換または出力コネクターによるデータ出力の前に実行します。

レコード抽出を実行するタイミング※

入力コネクターによるデータ入力またはフォーマット変換のあとに実行します。

注※

フィルター，レコード抽出，レコード間のマッピングを実行する場合，どれを先に実行してもかまいません。

なお，データの編集については，アダプター構成定義ファイルの編集用 CB 定義で設定します。アダプター構成定義ファイルの CB 定義については，「9.　アダプター用定義ファイル」を参照してください。

2.5.3　データの出力方法の検討

標準提供アダプターが提供するデータの出力方法の中から，どの方法でデータを出力するかを検討します。

データの出力は，出力アダプターで実行する処理です。標準提供アダプターでは，データ出力用のコネクター（出力コネクター）を使用して，ストリームデータの集計・分析結果のデータをファイルやダッシュボードに出力できます。データの出力方法を次の表に示します。

表 2-5　データの出力方法

項番	データの出力方法	説明
1	ファイルへの出力	**ファイル出力コネクター**を使用して，ストリームデータの集計・分析結果のデータをファイルに出力できます。 ファイルへの出力については，「11.6　ファイルへの出力」を参照してください。
2	ダッシュボードへの出力	**ダッシュボード出力コネクター**を使用して，ストリームデータの集計・分析結果のデータを出力して，Flex Dashboard の Dashboard Viewer で表示できます。 なお，Dashboard Viewer で表示するダッシュボードの画面を編集して，グラフの表示レイアウトなどを変更できます。 ダッシュボードへの出力については，「11.7　ダッシュボードへの出力」を参照してください。

なお，データの出力については，アダプター構成定義ファイルの出力用 CB 定義で設定します。アダプター構成定義ファイルの CB 定義については，「9.　アダプター用定義ファイル」を参照してください。

2.6　集計・分析シナリオの検討

ここでは，ストリームデータを集計・分析するためのシナリオの検討について説明します。また，シナリオを定義するクエリグループを実行するための運用ディレクトリの構成についても説明します。

2.6.1　ストリームデータの集計・分析シナリオの検討

ストリームデータ処理システムでは，どのようなデータを入力して，どのように処理して出力するかという，ストリームデータの集計・分析シナリオをクエリグループとして定義します。

クエリグループは，入力ストリーム，クエリ，および出力ストリームで構成されます。実行させるストリームデータの集計・分析の処理に応じて，幾つのクエリグループを定義するか，クエリグループ内で幾つのストリームやクエリを定義するかを検討してください。

クエリグループは，集計・分析シナリオの数だけ定義します。複数のクエリグループを実行することで，一つの SDP サーバで複数の集計・分析の処理を実行させることができます。一つの SDP サーバでは，最大で 8 個のクエリグループを実行できます。

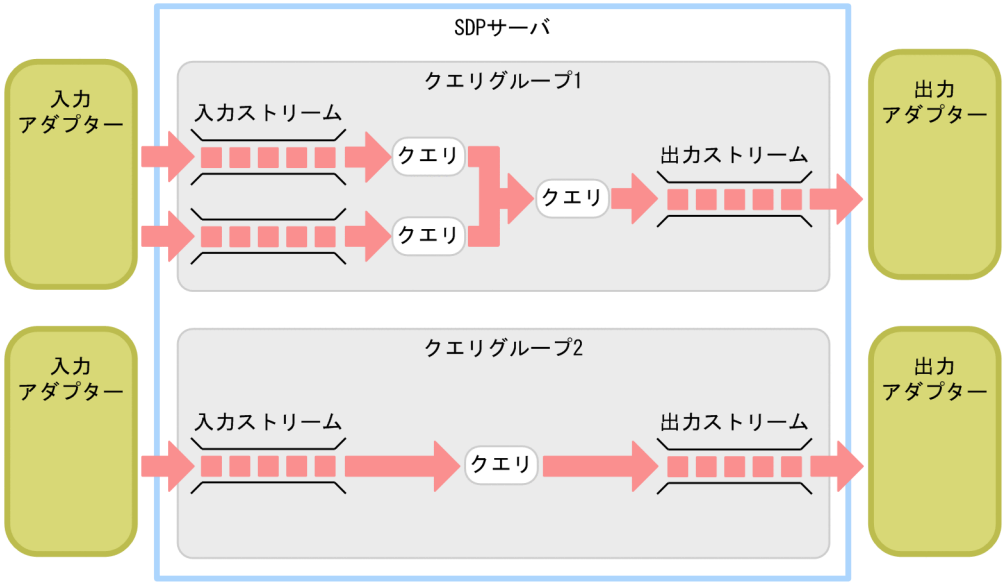
入力ストリームと出力ストリームは，対応する入出力アダプターの構成に応じて定義します。

クエリグループ内でどのような集計・分析処理を実行させるかに応じて，クエリを定義します。一つの集計・分析シナリオで複数の集計・分析処理を実行させたい場合には，クエリグループ内に複数のクエリを定義することもできます。

クエリグループ内で定義するクエリでは，ユーザーが Java でプログラミングした外部定義関数に集計および分析処理を実行させることもできます。なお，外部定義関数を使用する場合は，CQL で外部定義関数を定義するほかに，外部定義関数の作成が必要です。外部定義関数の作成については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

クエリグループの構成例を次の図に示します。

図 2-12　クエリグループの構成例



この例では，SDP サーバに二つのクエリグループ（クエリグループ 1 とクエリグループ 2）を登録し，それぞれのクエリグループが入力アダプターと出力アダプターに接続されています。クエリグループ 1 では，複数のクエリを定義して，複数の集計・分析の処理を実行しています。

2.6.2　クエリグループを実行する運用ディレクトリの構成の検討

複数のクエリグループを実行する場合には，幾つの運用ディレクトリでクエリグループを実行するかを検討してください。

複数のクエリグループを実行する場合，運用ディレクトリとクエリグループの数を 1：1 にする構成と 1：n にする構成があります。それぞれの構成について次の表に示します。

表 2-6　クエリグループと運用ディレクトリの構成

項番	構成	説明
1	運用ディレクトリとクエリグループの数を 1：1 にする構成	クエリグループごとに運用ディレクトリを作成し，一つの運用ディレクトリで一つのクエリグループを実行する構成です。 この構成のメリットは次のとおりです。 • サーバ間でメモリ空間を分割するため，ガーベージコレクションの発生を低減できるなど，性能面でのメリットがあります。 • ある SDP サーバで障害が発生した場合に，別の SDP サーバのクエリ実行に影響を及ぼさないため，障害を局所化できます。 なお，一つの運用ディレクトリで起動できる SDP サーバは，一つだけです。このため，この構成の場合は，クエリグループの数だけ SDP サーバを起動する必要があります。
2	運用ディレクトリとクエリグループの数を 1：n にする構成	運用ディレクトリを一つだけ作成し，一つの運用ディレクトリで複数のクエリグループを実行する構成です。 この構成にするメリットは次のとおりです。 • 運用ディレクトリの作成，定義ファイルの作成，アダプターの作成などの運用環境の構築が一度で済みます。 • 一つの運用ディレクトリで一つのクエリグループを実行する構成と比べて，少ない資源で複数のクエリグループを実行できます。

なお，一つの SDP サーバのメモリ使用量の見積もりが 1.6 ギガバイトを超える場合には，運用ディレクトリとクエリグループの数を 1：1 にする構成をお勧めします。

クエリグループの数が 9 個以上の場合には，「2.7　メモリ使用量の見積もり」で見積もるメモリ使用量を参考にして，幾つの運用ディレクトリでクエリグループを実行するか（1：1 の構成をクエリグループの数だけ作成するか，1：n の構成を複数作成するか）を決定してください。

2.6.3　外部定義関数で実行する処理の検討

外部定義関数を使用する場合には，外部定義関数で実行する，集計および分析処理について検討してください。その上で，Java の API を使用するなど，処理の実現方法を検討してください。

外部定義関数の作成については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

2.7　メモリ使用量の見積もり

ここでは，ストリームデータ処理エンジンと標準提供アダプターに関するメモリ使用量の見積もり方法，およびメモリ使用量の見積もり例について説明します。

メモリ使用量を見積もる項目と見積もり方法の参照先を次の表に示します。

表 2-7　メモリ使用量を見積もる項目と見積もり方法の参照先

項番	メモリ使用量を見積もる項目		参照先
1	ストリームデータ処理エンジン	SDP サーバの実行	2.7.1(1)
2		タプル	2.7.1(2)
3		タプルの保持	2.7.1(3)
4		ストリームキューで使用する要素数の上限値の変更	2.7.1(4)
5		API トレースのバッファーの内容の変更	2.7.1(5)
6		タプルログの取得	2.7.1(6)
7		クエリグループの実行	2.7.1(7)
8		タイムスタンプ調整	2.7.1(8)
9	標準提供アダプター	アダプターの実行	2.7.2(1)
10		インプロセスグループのアダプター	2.7.2(2)
11		RMI グループのアダプター	2.7.2(3)
12		レコードの処理（ファイル入力コネクターを使用する場合）	2.7.2(4)
13		レコードの処理（HTTP パケット入力コネクターを使用する場合）	2.7.2(5)
14		レコードの処理（ファイル出力コネクターまたはダッシュボード出力コネクターを使用する場合）	2.7.2(6)
15		レコードの蓄積（レコード抽出またはダッシュボード出力コネクターを使用する場合）	2.7.2(7)

2.7.1　ストリームデータ処理エンジンに関するメモリ使用量の見積もり

ここでは，ストリームデータ処理エンジンに関するメモリ使用量の見積もり方法について説明します。

ストリームデータ処理エンジンに関するメモリ使用量は，SDP サーバ用定義ファイルの SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）の次の表に示すパラメーターで指定します。

表 2-8　ストリームデータ処理エンジンに関するメモリ使用量を指定するパラメーター

項番	パラメーター	説明
1	SDP_INITIAL_MEM_SIZE	Java ヒープの初期サイズを設定します。デフォルト値は，512 メガバイトです。

項番	パラメーター	説明
2	SDP_MAX_MEM_SIZE	Java ヒープの最大サイズを設定します。デフォルト値は，1,024 メガバイトです。

デフォルト値ではメモリ使用量が不足する場合には，各パラメーターの値を変更してください。ストリームデータの処理には，以降で説明するストリームデータ処理エンジンに関するメモリ使用量が固定で必要となります。ガーベージコレクションの発生を低減させるため，ここで計算した値を基に，各パラメーターに余裕のある値を指定してください。

SDP サーバ用 JavaVM オプションファイルのパラメーターについては，「8.4　SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg)」を参照してください。

ストリームデータ処理エンジンに関するメモリ使用量について，項目ごとに説明します。

(1)　SDP サーバの実行に必要なメモリ使用量

SDP サーバの実行に必要なメモリ使用量は，145 メガバイトです。

(2)　タプル一つ当たりに必要なメモリ使用量

タプル一つ当たりに必要なメモリ使用量は，**タプル一つ当たりのサイズ**で決まります。タプル一つ当たりのサイズは，次に示す計算式で求められます。

タプル一つ当たりのサイズ（単位：バイト）＝
　500※＋ユーザー定義データ領域サイズ

注※

タプル固定領域サイズ（単位：バイト）です。

計算式中の「ユーザー定義データ領域サイズ」は，ユーザーが CQL のスキーマで定義したデータ型とその数によって決まります。ユーザー定義データ領域サイズは，次に示す計算式で求められます。

ユーザー定義データ領域サイズ（単位：バイト）＝
　（256※×ユーザー定義データ数）＋（文字列データで定義した文字列長×2）

注※

データ一つ当たりのサイズ（単位：バイト）です。

(3)　タプルの保持に必要なメモリ使用量

ウィンドウ演算に使用するウィンドウのサイズが大きくなるほど，SDP サーバ内に保持するタプルも増えるため，タプルの保持に必要なメモリ使用量が増加します。ウィンドウのサイズは，タプルの数（ROWS ウィンドウ），または時間（RANGE ウィンドウ）で指定します。

ウィンドウ演算を使用する場合のメモリ使用量は，次の計算式で求められます。なお，RANGE ウィンドウの場合は，時刻解像度を指定するかどうかで計算式が異なります。

ROWSウィンドウの場合
　メモリ使用量（単位：バイト）＝
　　A×ROWSウィンドウで指定したタプルの数
　　＋50,000※

RANGEウィンドウの場合（時刻解像度を指定しないとき）
　メモリ使用量（単位：バイト）＝
　　A×RANGEウィンドウで指定した時間
　　×単位時間当たりのタプルの入力数＋50,000※

RANGEウィンドウの場合（時刻解像度を指定するとき）
　メモリ使用量（単位：バイト）＝
　　A×RANGEウィンドウで指定した時間
　　÷時刻解像度で指定したメッシュ間隔
　　×メッシュ間隔当たりのタプル入力数＋50,000※

（凡例）

A：「(2)　タプル一つ当たりに必要なメモリ使用量」で示したタプル一つ当たりのサイズです。

注※

ウィンドウ固定領域サイズ（単位：バイト）です。

時刻解像度の指定は，データの入力頻度を予測することが困難な場合や，メモリ使用量を一定量に抑えたい場合に有効です。ただし，時刻解像度の指定には，前提条件や制約があるため注意してください。時刻解像度の指定については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

(4)　ストリームキューで使用する要素数の上限値の変更に必要なメモリ使用量

SDP サーバ用定義ファイルのシステムコンフィグプロパティファイル（system_config.properties）の engine.maxQueueSize パラメーターで，ストリームキューの要素数の上限値を変更できます。要素数の上限値を変更すると，ストリームキューのメモリ使用量に影響します。要素数の上限値を変更する場合の，ストリームキュー一つ当たりのメモリ使用量は，次に示す計算式で求められます。

ストリームキュー一つ当たりのメモリ使用量（単位：バイト）＝A×B

（凡例）

A：engine.maxQueueSize パラメーターの指定値（ストリームキューの要素数の上限値）です。

B：「(2)　タプル一つ当たりに必要なメモリ使用量」で示したタプル一つ当たりのサイズです。

システムコンフィグプロパティファイルのパラメーターについては，「8.6　システムコンフィグプロパティファイル（system_config.properties）」を参照してください。

(5)　API トレースのバッファーの内容の変更に必要なメモリ使用量

SDP サーバ用定義ファイルのシステムコンフィグプロパティファイル（system_config.properties）の次に示すパラメーターで，API トレースのバッファーの内容を変更できます。

- trc.api.bufferSize（API トレースのバッファーの最大サイズを指定）
- trc.api.bufferCount（API トレースのバッファーの面数を指定）
- trc.api.ioBufferSize（API トレースの I/O バッファーの最大サイズを指定）

これらのパラメーターのうち，どれか一つでも値を変更すると，API トレースのメモリ使用量に影響します。API トレースのバッファーの内容を変更する場合の，API トレースのメモリ使用量は，次に示す計算式で求められます。

APIトレースのメモリ使用量（単位：キロバイト）＝A×B＋C

（凡例）

A：trc.api.bufferSize パラメーターの指定値（API トレースのバッファーの最大サイズ）です。

B：trc.api.bufferCount パラメーターの指定値（API トレースのバッファーの面数）です。

C：trc.api.ioBufferSize パラメーターの指定値（API トレースの I/O バッファーの最大サイズ）です。

システムコンフィグプロパティファイルのパラメーターについては，「8.6　システムコンフィグプロパティファイル（system_config.properties）」を参照してください。

(6)　タプルログの取得に必要なメモリ使用量

SDP サーバ用定義ファイルのシステムコンフィグプロパティファイル（system_config.properties）の tpl.outputTrigger パラメーターで，タプルログのファイルへの出力契機を変更できます。

tpl.outputTrigger パラメーターで，タプルログのファイルへの出力契機に「BUFFER（デフォルト値）」を指定すると，ストリーム一つ当たりのタプルログの取得に必要なメモリ使用量に影響します。

タプルログのファイルへの出力契機に「BUFFER」を指定する場合の，ストリーム一つ当たりのタプルログの取得に必要なメモリ使用量は，次に示す計算式で求められます。

ストリーム一つ当たりのタプルログの取得に必要なメモリ使用量（単位：キロバイト）＝A×B

（凡例）

A：システムコンフィグプロパティファイルの tpl.bufferSize パラメーターの指定値（タプルログのバッファーの最大サイズ）です。

B：システムコンフィグプロパティファイルの tpl.bufferCount パラメーターの指定値（タプルログのバッファーの面数）です。

システムコンフィグプロパティファイルのパラメーターについては，「8.6　システムコンフィグプロパティファイル（system_config.properties）」を参照してください。

(7)　クエリグループの実行に必要なメモリ使用量

クエリグループ一つ当たりのメモリ使用量は，次に示す計算式で求められます。

クエリグループ一つ当たりのメモリ使用量（単位：メガバイト）＝
2※1×登録したクエリグループに含まれるクエリの数×128※2

注※1

クエリ一つ当たりのメモリ使用量（単位：メガバイト）です。

注※2

クエリグループの実行領域のサイズ（単位：メガバイト）です。

(8)　タイムスタンプ調整に必要なメモリ使用量

タイムスタンプ調整機能を使用する場合のメモリ使用量は，次に示す計算式で求められます。

タイムスタンプ調整に必要なメモリ使用量（単位：バイト）＝
入力ストリームの数×時刻単位当たりのタプル数×時刻調整範囲×A

（凡例）

A：「(2)　タプル一つ当たりに必要なメモリ使用量」で示したタプル一つ当たりのサイズです。

計算式中の「時刻単位当たりのタプル数」と「時刻調整範囲」の指定値は，SDP サーバ用定義ファイルのシステムコンフィグプロパティファイル（system_config.properties），クエリグループ用プロパティファイル，またはストリーム用プロパティファイルの stream.timestampAccuracy パラメーターの指定値によって変わります。stream.timestampAccuracy パラメーターの指定値と計算式中の値との関係を次の表に示します。

stream.timestampAccuracy パラメーターの指定値	時刻単位当たりのタプル数の指定値	時刻調整範囲の指定値
unuse	0	0

stream.timestampAccuracy パラメーターの指定値	時刻単位当たりのタプル数の指定値	時刻調整範囲の指定値
unuse 以外	時刻単位当たりのタプル数	時刻調整範囲※＋ 1

注※

stream.timestampAccuracy パラメーターの指定値です。

各ファイルのパラメーターについては，「8.6　システムコンフィグプロパティファイル（system_config.properties）」，「8.7　クエリグループ用プロパティファイル」，または「8.8　ストリーム用プロパティファイル」を参照してください。

2.7.2　標準提供アダプターに関するメモリ使用量の見積もり

ここでは，標準提供アダプターに関するメモリ使用量の見積もり方法について説明します。

標準提供アダプターに関するメモリ使用量は，SDP サーバ用定義ファイルの SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg），または RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）の次の表に示すパラメーターで指定します。

表 2-9　標準提供アダプターに関するメモリ使用量を指定するパラメーター（SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg））

項番	パラメーター	説明
1	SDP_INITIAL_MEM_SIZE	Java ヒープの初期サイズを設定します。デフォルト値は，512 メガバイトです。
2	SDP_MAX_MEM_SIZE	Java ヒープの最大サイズを設定します。デフォルト値は，1,024 メガバイトです。

表 2-10　標準提供アダプターに関するメモリ使用量を指定するパラメーター（RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg））

項番	パラメーター	説明
1	SDP_INITIAL_MEM_SIZE	Java ヒープの初期サイズを設定します。デフォルト値は，256 メガバイトです。
2	SDP_MAX_MEM_SIZE	Java ヒープの最大サイズを設定します。デフォルト値は，512 メガバイトです。

デフォルト値ではメモリ使用量が不足する場合には，各パラメーターの値を変更してください。標準提供アダプターの実行には，「2.7.1　ストリームデータ処理エンジンに関するメモリ使用量の見積もり」で見積もったメモリ使用量に加えて，以降で説明する標準提供アダプターに関するメモリ使用量が固定で必要となります。ガーベージコレクションの発生を低減させるため，ここで計算した値を基に，各パラメーターに余裕のある値を指定してください。

各ファイルのパラメーターについては，「8.4　SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）」，または「8.5　RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）」を参照してください。

標準提供アダプターに関するメモリ使用量について，項目ごとに説明します。

(1)　アダプターの実行に必要なメモリ使用量

アダプターの実行に必要なメモリ使用量は，5 メガバイト（固定値）です。

(2)　インプロセスグループで構成するアダプター一つ当たりに必要なメモリ使用量

インプロセスグループで構成するアダプター一つ当たりに必要なメモリ使用量は，10 メガバイト（固定値）です。

(3)　RMI グループで構成するアダプター一つ当たりに必要なメモリ使用量

RMI グループで構成するアダプター一つ当たりに必要なメモリ使用量は，20 メガバイト（固定値）です。

(4)　レコードの処理に必要なメモリ使用量（ファイル入力コネクターを使用する場合）

ファイル入力コネクターを使用する場合，レコードの処理に必要なメモリ使用量は，次に示す計算式で求められます。

```
レコードの処理に必要なメモリ使用量（単位：バイト）＝
　（1レコードの処理に必要なメモリ使用量×一度に処理するレコード数）
　＋（A×一度に処理するレコード数）
```

（凡例）

A：「2.7.1(2)　タプル一つ当たりに必要なメモリ使用量」で示したタプル一つ当たりのサイズです。

1 レコードの処理に必要なメモリ使用量

計算式中の「1 レコードの処理に必要なメモリ使用量」は，次に示す計算式で求められます。

```
1レコードの処理に必要なメモリ使用量（単位：バイト）＝
　（500×フォーマット変換定義のフィールド定義の数※）
　＋（共通形式レコードの最大長）
```

注※

アダプター用定義ファイルのアダプター構成定義ファイルにある，フォーマット変換定義（FormatDefinition タグ）のフィールド定義（field タグ）の数です。

一度に処理するレコード数

計算式中の「一度に処理するレコード数」は，次に示す計算式で求められます。

```
一度に処理するレコード数（単位：バイト）＝
　ファイル入力コネクターが一度に読み込むレコード数※
```

注※

アダプター用定義ファイルのアダプター構成定義ファイルにある，ファイル入力コネクター定義（FileInputConnectorDefinition）の読み込み定義（input タグ）の readUnit 属性の値です。

アダプター構成定義ファイルの定義については，「9.5　アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）」を参照してください。

(5)　レコードの処理に必要なメモリ使用量 (HTTP パケット入力コネクターを使用する場合)

HTTP パケット入力コネクターを使用する場合，レコードの処理に必要なメモリ使用量は，次に示す計算式で求められます。

```
レコードの処理に必要なメモリ使用量（単位：バイト）＝
　（1レコードの処理に必要なメモリ使用量×一度に処理するレコード数）
　＋（A×一度に処理するレコード数）
```

（凡例）

A：「2.7.1(2)　タプル一つ当たりに必要なメモリ使用量」で示したタプル一つ当たりのサイズです。

1 レコードの処理に必要なメモリ使用量

計算式中の「1 レコードの処理に必要なメモリ使用量」は，次に示す計算式で求められます。

1レコードの処理に必要なメモリ使用量（単位：バイト）＝
（500×HTTPパケット入力コネクター定義のフィールド定義の数※）
＋（共通形式レコードの最大長）

注※

アダプター用定義ファイルのアダプター構成定義ファイルにある，HTTP パケット入力コネクター定義（HttpPacketInputConnectorDefinition タグ）のフィールド定義（field タグ）の数です。

一度に処理するレコード数

計算式中の「一度に処理するレコード数」は，次に示す計算式で求められます。

一度に処理するレコード数（単位：バイト）＝
HTTPパケット入力コネクターが一度に読み込むレコード数※

注※

アダプター用定義ファイルのアダプター構成定義ファイルにある，HTTP パケット入力コネクター定義（HttpPacketInputConnectorDefinition タグ）の出力定義（output タグ）の unit 属性の値です。

アダプター構成定義ファイルの定義については，「9.5　アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）」を参照してください。

(6)　レコードの処理に必要なメモリ使用量（ファイル出力コネクターまたはダッシュボード出力コネクターを使用する場合）

ファイル出力コネクターまたはダッシュボード出力コネクターを使用する場合，レコードの処理に必要なメモリ使用量は，次に示す計算式で求められます。

レコードの処理に必要なメモリ使用量（単位：バイト）＝
（1レコードの処理に必要なメモリ使用量×一度に処理するレコード数）
＋（A×一度に処理するレコード数）

（凡例）

A：「2.7.1(2)　タプル一つ当たりに必要なメモリ使用量」で示したタプル一つ当たりのサイズです。

1 レコードの処理に必要なメモリ使用量

計算式中の「1 レコードの処理に必要なメモリ使用量」は，次に示す計算式で求められます。

1レコードの処理に必要なメモリ使用量（単位：バイト）＝
（500×マッピング定義のマッピング情報定義の数※）
＋（共通形式レコードの最大長）

注※

アダプター用定義ファイルのアダプター構成定義ファイルにある，マッピング定義（MappingDefinition タグ）のレコードとストリーム間のマッピング情報定義（map タグ）の数です。

アダプター構成定義ファイルの定義については，「9.5　アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）」を参照してください。

一度に処理するレコード数

計算式中の「一度に処理するレコード数」は，1,024（タプル受信の最大タプル取得数）です。

(7)　レコードの蓄積に必要なメモリ使用量（レコード抽出またはダッシュボード出力コネクターを使用する場合）

レコード抽出またはダッシュボード出力コネクターを使用する場合，レコードの蓄積に必要なメモリ使用量は，次に示す計算式で求められます。

レコードの蓄積に必要なメモリ使用量（単位：バイト）＝
（1レコードの処理に必要なメモリ使用量×蓄積するレコード数）

1 レコードの処理に必要なメモリ使用量

計算式中の「1 レコードの処理に必要なメモリ使用量」は，使用するコネクターによって異なります。

- ファイル入力コネクターを使用してレコード抽出する場合

  「(4)　レコードの処理に必要なメモリ使用量（ファイル入力コネクターを使用する場合)」の 1 レコードの処理に必要なメモリ使用量を参照してください。

- HTTP パケット入力コネクターを使用してレコード抽出する場合

  「(5)　レコードの処理に必要なメモリ使用量（HTTP パケット入力コネクターを使用する場合)」の 1 レコードの処理に必要なメモリ使用量を参照してください。

- ダッシュボード出力コネクターを使用する場合

  「(6)　レコードの処理に必要なメモリ使用量(ファイル出力コネクターまたはダッシュボード出力コネクターを使用する場合)」の 1 レコードの処理に必要なメモリ使用量を参照してください。

蓄積するレコード数

計算式中の「蓄積するレコード数」は，レコード抽出を使用するか，ダッシュボード出力コネクターを使用するかによって異なります。

- レコード抽出を使用する場合

  アダプター用定義ファイルのアダプター構成定義ファイルにある，レコード抽出定義（RecordExtractionDefinition タグ）の抽出条件群の定義（extractions タグ）の size 属性の値です。

- ダッシュボード出力コネクターを使用する場合

  アダプター用定義ファイルのアダプター構成定義ファイルにある，ダッシュボード出力コネクター定義（DashboardOutputConnectorDefinition タグ）の MaxNum 属性の値です。

アダプター構成定義ファイルの定義については，「9.5　アダプター構成定義ファイル（AdaptorCompositionDefinition.xml)」を参照してください。

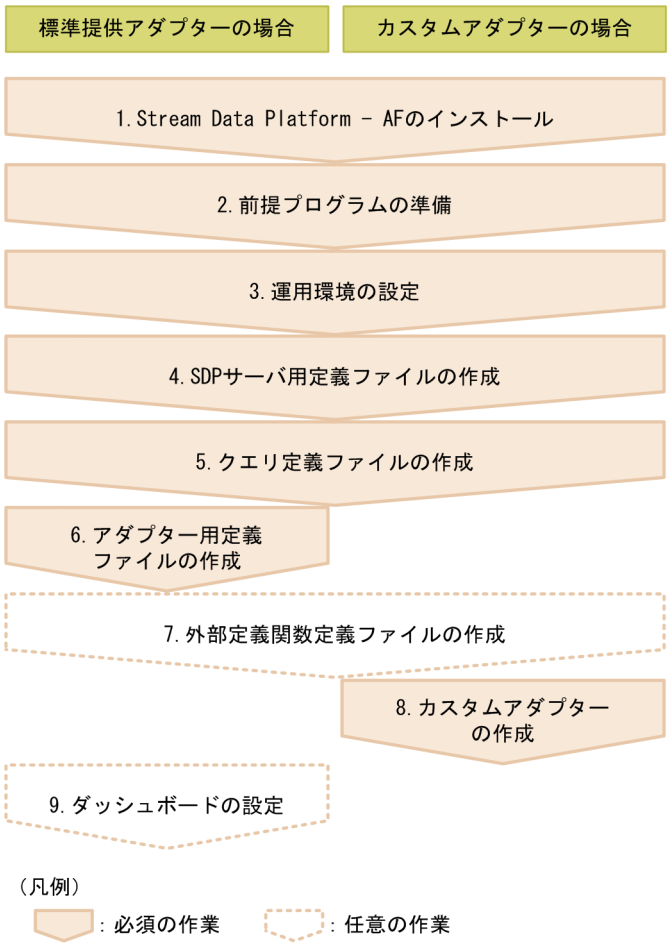
3 システムの構築

この章では，ストリームデータ処理システムの構築の流れ，Stream Data Platform - AF のディレクトリ構成，およびシステムの構築に必要な設定について説明します。

3.1　構築の流れ

ストリームデータ処理システムの構築に必要な作業は，アダプターと SDP サーバとの連携形態やアダプターで使用する機能によって異なります。システムの構築の流れを次の図に示します。

図 3-1　システムの構築の流れ



それぞれの作業について，次に説明します。

1. Stream Data Platform - AF のインストール

Stream Data Platform - AF をインストールします。インストール手順については，Stream Data Platform - AF のリリースノートを参照してください。また，Stream Data Platform - AF のインストールディレクトリの構成については，「3.2.1　インストールディレクトリの構成」を参照してください。

2. 前提プログラムの準備

HTTP パケットを入力したり，ダッシュボードに出力したりする場合には，前提プログラムが必要です。必要な前提プログラムについては，「2.2.1　ストリームデータ処理システムのプログラムの構成」を参照してください。

前提プログラムのドキュメントを参照して，プログラムのインストール，およびセットアップをしてください。

3. 運用環境の設定

運用ユーザーを登録して，運用ディレクトリを作成します。

設定方法については，「3.3　運用環境の設定」を参照してください。

4. SDP サーバ用定義ファイルの作成

SDP サーバ用定義ファイルを作成して，SDP サーバの動作を設定します。

ファイルの作成方法については，「3.4　SDP サーバ用定義ファイルの作成」を参照してください。

5. クエリ定義ファイルの作成

クエリ定義ファイルを作成して，ストリームデータの集計・分析シナリオを定義します。

ファイルの作成方法については，「3.5　クエリ定義ファイルの作成」を参照してください。

6. アダプター用定義ファイルの作成

標準提供アダプターを使用する場合は，アダプター用定義ファイルを作成して，標準提供アダプターの動作を設定します。カスタムアダプターを使用する場合は，作成する必要はありません。

ファイルの作成方法については，「3.6　アダプター用定義ファイルの作成」を参照してください。

7. 外部定義関数定義ファイルの作成

外部定義関数を使用する場合は，外部定義関数定義ファイルを作成して，外部定義関数の関数名や戻り値と，ユーザーが Java で作成したクラスとの関連づけについて定義します。外部定義関数を使用しない場合は，作成する必要はありません。

ファイルの作成方法については，「3.7　外部定義関数定義ファイルの作成」を参照してください。

8. カスタムアダプターの作成

Stream Data Platform - AF が提供するデータ送受信 API を使用して，カスタムアダプターを作成します。標準提供アダプターを使用する場合は，作成する必要はありません。

カスタムアダプターの作成方法については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

9. ダッシュボードの設定

標準提供アダプターでストリームデータの集計・分析結果をダッシュボードに出力する場合には，ダッシュボードの設定をしてください。ダッシュボードに出力しない場合は，設定する必要はありません。

設定方法については，「3.8　ダッシュボードの設定」を参照してください。

なお，構築済みのシステムの設定を変更する場合の手順については，「5.　システムの変更」を参照してください。

3.2　ディレクトリ構成

ここでは，Stream Data Platform - AF のインストールディレクトリと運用ディレクトリの構成について説明します。

参考

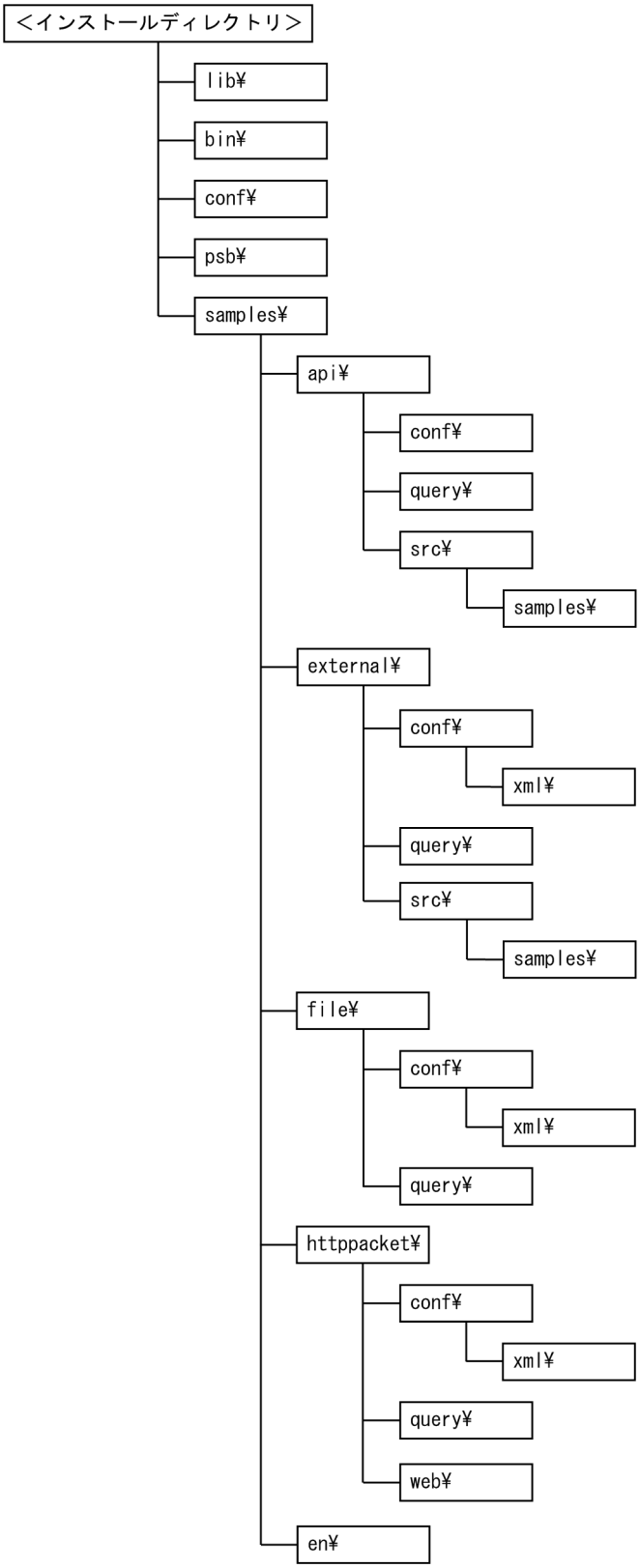
Stream Data Platform - AF をインストールすると，日立トレース共通ライブラリのディレクトリ（デフォルトのインストールディレクトリは，Windows の場合は「<OS インストールドライブ>:¥Program Files¥Hitachi¥HNTRLib2¥」，Linux の場合は「/opt/hitachi/HNTRLib2/」です）も作成されます。このディレクトリは，ユーザーが使用することはないため，このマニュアルでは説明しません。

3.2.1　インストールディレクトリの構成

ここでは，Stream Data Platform - AF のインストールディレクトリについて説明します。

Stream Data Platform - AF のインストールディレクトリの構成を次の図に示します。

図 3-2　インストールディレクトリの構成



参考

デフォルトのインストールディレクトリは，Windows の場合は「<OS インストールドライブ>:¥Program Files ¥Hitachi¥sdp¥」，Linux の場合は「/opt/hitachi/sdp/」です。

インストールディレクトリの各ディレクトリの説明を次の表に示します。ディレクトリ内のファイルについては，表中の参照先を参照してください。

表 3-1　インストールディレクトリの各ディレクトリの説明

ディレクトリ				ディレクトリの説明	参照先
lib¥				Stream Data Platform - AF のライブラリなどを格納しているディレクトリです。	－
bin¥				運用ディレクトリの作成時に使用する sdpsetup コマンドを格納しているディレクトリです。 運用ディレクトリを作成したあとは，＜運用ディレクトリ＞¥bin¥下のコマンドを実行してください。	7 章
conf¥				Stream Data Platform - AF のシステムで使用するファイルを格納しているディレクトリです。	－
psb¥				Stream Data Platform - AF で使用する Web サーバなどを格納しているディレクトリです。	－
samples¥	－			サンプルファイルを格納しているディレクトリです。ここに格納しているサンプルファイルの種類を次に示します。 • 標準提供アダプターを使用する場合のコールバックごとのサンプルファイル 格納先：「file」，「httppacket」ディレクトリ • カスタムアダプターを使用する場合のサンプルファイル 格納先：「api」ディレクトリ • 外部定義関数を使用する場合のサンプルファイル 格納先：「external」ディレクトリ • 英語版のサンプルファイル 格納先：「en」ディレクトリ	－
	api¥	－		カスタムアダプターを使用する場合のサンプルファイルを格納しているディレクトリです。	－
		conf¥		SDP サーバ用定義ファイルのサンプルファイルを格納しているディレクトリです。	3.4， 8 章
		query¥		クエリ定義ファイルのサンプルファイルを格納しているディレクトリです。	3.5
		src¥	samples¥	カスタムアダプターのサンプルプログラムを格納しているディレクトリです。	関連マニュアル※

ディレクトリ				ディレクトリの説明	参照先
samples¥	external¥	−		外部定義関数を使用する場合のサンプルファイルを格納しているディレクトリです。	−
		conf¥	−	SDP サーバ用定義ファイルのサンプルファイルを格納しているディレクトリです。	3.4，8 章
			xml¥	アダプター用定義ファイル，および外部定義関数定義ファイルのサンプルファイルを格納しているディレクトリです。	3.7，10 章
		query¥		クエリ定義ファイルのサンプルファイルを格納しているディレクトリです。	3.5
		src¥	samples¥	外部定義関数のサンプルプログラムを格納しているディレクトリです。	関連マニュアル※
	file¥	−		標準提供アダプターで次のコールバックを使用する場合のサンプルファイルを格納しているディレクトリです。 • ファイル入力コネクター • ファイル出力コネクター	−
		conf¥	−	SDP サーバ用定義ファイルのサンプルファイルを格納しているディレクトリです。	3.4， 8 章
			xml¥	アダプター用定義ファイルのサンプルファイルを格納しているディレクトリです。	3.6， 9 章
		query¥		クエリ定義ファイルのサンプルファイルを格納しているディレクトリです。	3.5
	httppacket¥	−		標準提供アダプターで次のコールバックを使用する場合のサンプルファイルを格納しているディレクトリです。 • HTTP パケット入力コネクター • フィルター • レコード抽出 • ダッシュボード出力コネクター	−
		conf¥	−	SDP サーバ用定義ファイルのサンプルファイルを格納しているディレクトリです。	3.4， 8 章
			xml¥	アダプター用定義ファイルのサンプルファイルを格納しているディレクトリです。	3.6， 9 章
		query¥		クエリ定義ファイルのサンプルファイルを格納しているディレクトリです。	3.5
		web¥		ダッシュボード出力コネクターのコールバックを使用する場合のサンプルファイルを格納しているディレクトリです。	3.8
	en¥	−		「5.7　メッセージの出力言語種別の変更」で出力言語種別を英語に変更する場合の，英語版の	−

ディレクトリ			ディレクトリの説明	参照先
samples¥	en¥	－	サンプルファイルを格納しているディレクトリです。ディレクトリ構成は「samples」ディレクトリ下と同じです（「en」ディレクトリを除く）。	－

（凡例）

－：該当しません。

注※

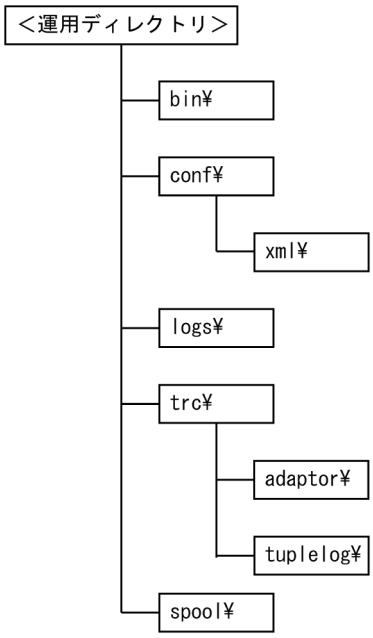
サンプルプログラムについては，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

3.2.2　運用ディレクトリの構成

運用ディレクトリは，ストリームデータ処理システムの運用で使用するディレクトリのことです。運用ディレクトリは，Stream Data Platform - AF のインストール後にユーザーが作成します。運用ディレクトリの作成方法については，「3.3.2　運用ディレクトリの作成」を参照してください。

運用ディレクトリの構成を次の図に示します。

図 3-3　運用ディレクトリの構成



運用ディレクトリの各ディレクトリの説明を次の表に示します。ディレクトリ内のファイルについては，表中の参照先を参照してください。

表 3-2　運用ディレクトリの各ディレクトリの説明

ディレクトリ	ディレクトリの説明	参照先
bin¥	運用時に使用するコマンド格納用のディレクトリです。	7 章

ディレクトリ		ディレクトリの説明	参照先
conf¥	－	SDP サーバ用定義ファイル格納用のディレクトリです。ユーザーが作成した SDP サーバ用定義ファイルは，ここに格納してください。	3.4, 8 章
	xml¥	アダプター用定義ファイルと外部定義関数定義ファイルの格納用のディレクトリです。ユーザーが作成したアダプター用定義ファイルと外部定義関数定義ファイルは，ここに格納してください。	3.6, 9 章，3.7，10 章
logs¥		ログファイル出力用のディレクトリです。	6.3.1
trc¥	－	API トレースなどの，各種トレース出力用のディレクトリです。	6.3.2
	adaptor¥	アダプタートレース出力用のディレクトリです。	6.3.3
	tuplelog¥	タプルログ出力用のディレクトリです。	6.3.4
spool¥		Stream Data Platform - AF の作業用のディレクトリです。	－

（凡例）

－：該当しません。

3.3　運用環境の設定

Stream Data Platform - AF のインストール後に，運用ユーザーを登録して，運用ディレクトリを作成してください。

3.3.1　運用ユーザーの登録

運用ユーザーは，ストリームデータ処理システムを構築，運用するユーザーのことです。Stream Data Platform - AF のコマンドを実行する権限があり，運用ディレクトリの所有者です。

運用ユーザーの登録方法を次に示します。

Windows の場合

コントロールパネルのユーザーアカウントで，運用ユーザーのユーザーアカウントを追加してください。ユーザーアカウントの追加方法については，Windows のヘルプを参照してください。

Linux の場合

スーパーユーザーでログインし，useradd コマンドで，運用ユーザーのユーザーアカウントを追加してください。ユーザーアカウントの追加方法については，useradd コマンドのマニュアルを参照してください。

また，パケットアナライザーの実行には，ユーザーアカウントに管理者権限が必要な場合があります。管理者権限が必要かどうかは，使用するパケットアナライザーのマニュアルを参照してください。なお，WinDump（Windows の場合）または tcpdump（Linux の場合）を使用するには，管理者権限が必要です。

3.3.2　運用ディレクトリの作成

運用ディレクトリは，ストリームデータ処理システムの構築と運用で使用するディレクトリのことです。運用ディレクトリは，「3.3.1　運用ユーザーの登録」で登録した運用ユーザーが sdpsetup コマンドを実行して作成します。

sdpsetup コマンドの実行例を次に示します。

Windows の場合

```
＜インストールディレクトリ＞¥bin¥sdpsetup ＜運用ディレクトリ＞
```

Linux の場合

```
/opt/hitachi/sdp/bin/sdpsetup ＜ユーザーID＞ ＜運用ディレクトリ＞
```

＜運用ディレクトリ＞には，運用ディレクトリの絶対パスを指定します。

運用ディレクトリには，すべての運用ユーザーに「フルコントロール」のアクセス権を設定してください。Stream Data Platform - AF のインストールディレクトリは，運用ディレクトリに指定できません。

なお，複数のクエリグループを実行する場合は，クエリグループ数分の運用ディレクトリを作成するか，または一つの運用ディレクトリで複数のクエリグループを実行します。それぞれの方法のメリット，およびデメリットについては，「2.6.2　クエリグループを実行する運用ディレクトリの構成の検討」を参照してください。

sdpsetup コマンドについては，「7.　コマンド」の「sdpsetup（運用環境のセットアップ）」を参照してください。

運用ディレクトリの構成については，「3.2.2　運用ディレクトリの構成」を参照してください。

3.4　SDP サーバ用定義ファイルの作成

SDP サーバ用定義ファイルを作成して，SDP サーバの動作環境を設定してください。

3.4.1　SDP サーバ用定義ファイルで設定できること

SDP サーバ用定義ファイルでは，SDP サーバやアダプターを実行する JavaVM の起動オプション，SDP サーバのポート番号や採取する API トレースとタプルログの詳細などを設定します。また，クエリグループやストリームなどのプロパティを設定します。

作成が必要なファイルは，アダプターと SDP サーバとの連携形態やプロパティの設定単位によって異なります。SDP サーバ用定義ファイルの作成の要否を次の表に示します。

表 3-3　SDP サーバ用定義ファイルの作成の要否

項番	SDP サーバ用定義ファイル	作成の要否	
1	SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）	○	運用ディレクトリごとに必ず作成します。 このファイルには，SDP サーバを実行する JavaVM オプションを設定します。なお，インプロセス連携アダプターは，このファイルで設定したオプションで動作します。
2	RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）	△	RMI 連携をする場合だけ，運用ディレクトリごと，またはアダプターごとに作成します。 このファイルには，RMI 連携アダプターを実行する JavaVM オプションを設定します。
3	システムコンフィグプロパティファイル（system_config.properties）	○	運用ディレクトリごとに必ず作成します。 このファイルには，SDP サーバで使用するポート番号，採取する API トレースとタプルログの詳細などを設定します。
4	クエリグループ用プロパティファイル	○	クエリグループごとに必ず作成します。 このファイルには，クエリ定義ファイルのパスやクエリグループを実行するときのチューニングパラメーターを設定します。
5	ストリーム用プロパティファイル	△	クエリグループ内のストリーム単位でチューニングパラメーターを設定したい場合だけ，ストリームごとに作成します。 このファイルを作成しない場合には，クエリグループ用プロパティファイルで設定した内容で動作します。
6	インプロセス連携用プロパティファイル	△	インプロセス連携をする場合だけ，標準提供アダプターはアダプターグループごと，カスタムアダプターはアダプターごとに作成します。 このファイルには，インプロセス連携アダプターのクラス名や jar ファイルのパスを設定します。
7	ログファイル出力用プロパティファイル（logger.properties）	△	運用ディレクトリごとに作成します。作成しない場合には，デフォルト値で動作します。 このファイルには，メッセージログやトレースログの面数やサイズを設定します。

（凡例）

○：必ず作成するファイルです。

△：必要に応じて作成するファイルです。

SDP サーバ用定義ファイルについては，「8.　SDP サーバ用定義ファイル」を参照してください。

3.4.2　SDP サーバ用定義ファイルの作成のしかた

ストリーム用プロパティファイル以外のファイルについては，インストール時に提供されるサンプルファイルを使用して作成することをお勧めします。サンプルファイルは，次のディレクトリに格納されていますので，コピーして編集してください。

- 標準提供アダプター用のサンプルファイル
  ＜インストールディレクトリ＞¥samples¥＜fileまたはhttppacket＞¥conf¥
- カスタムアダプター用のサンプルファイル
  ＜インストールディレクトリ＞¥samples¥api¥conf¥
- 外部定義関数用のサンプルファイル
  ＜インストールディレクトリ＞¥samples¥external¥conf¥

また，サンプルファイルには英語版もあります。サンプルファイルの格納ディレクトリについては，「3.2.1　インストールディレクトリの構成」を参照してください。

なお，サンプルファイルに設定されている値は，「8.　SDP サーバ用定義ファイル」の各ファイルのデフォルト値とは異なる場合がありますので，注意してください。

作成した SDP サーバ用定義ファイルの格納先は，ファイルによって異なります。SDP サーバ用定義ファイルの格納先については，「8.　SDP サーバ用定義ファイル」を参照してください。

3.5　クエリ定義ファイルの作成

クエリ定義ファイルを作成して，ストリームデータの集計・分析シナリオを定義してください。

3.5.1　クエリ定義ファイルで設定できること

クエリ定義ファイルでは，ストリームデータ処理システムへのストリームとクエリの登録や，ストリームデータの演算処理について定義します。クエリ定義ファイルは，クエリグループごとに作成します。

クエリ定義ファイルは，CQL で記述します。CQL によるクエリの定義方法については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

3.5.2　クエリ定義ファイルの作成のしかた

クエリ定義ファイルは, インストール時に提供されるサンプルファイルを使用して作成することをお勧めします。次のディレクトリに格納されているサンプルファイルをコピーして編集してください。

- 標準提供アダプター用のサンプルファイル
  ＜インストールディレクトリ＞¥samples¥＜fileまたはhttppacket＞¥query¥
- カスタムアダプター用のサンプルファイル
  ＜インストールディレクトリ＞¥samples¥api¥query¥
- 外部定義関数用のサンプルファイル
  ＜インストールディレクトリ＞¥samples¥external¥query¥

また，サンプルファイルには英語版もあります。サンプルファイルの格納ディレクトリについては，「3.2.1　インストールディレクトリの構成」を参照してください。

クエリ定義ファイルのサンプルファイルについては，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

作成したクエリ定義ファイルは任意の場所に格納し, ファイルのパスとファイル名をクエリグループ用プロパティファイルの querygroup.cqlFilePath パラメーターに指定します。クエリグループ用プロパティファイルについては，「8.7　クエリグループ用プロパティファイル」を参照してください。

3.6　アダプター用定義ファイルの作成

標準提供アダプターを使用する場合は，アダプター用定義ファイルを作成して，標準提供アダプターの動作を設定してください。カスタムアダプターを使用する場合は，作成する必要はありません。

3.6.1　アダプター用定義ファイルで設定できること

アダプター用定義ファイルでは，RMI 連携アダプターが使用するポート番号，アダプターグループの構成や，標準提供アダプターで実行するコールバックの処理などを設定します。

作成が必要なファイルは，アダプターと SDP サーバとの連携形態によって異なります。アダプター用定義ファイルの作成の要否を次の表に示します。

表 3-4　アダプター用定義ファイルの作成の要否

項番	アダプター用定義ファイル	作成の要否	
1	アダプターコマンド定義ファイル （AgentManagerDefinition.xml）	△	RMI 連携をする場合だけ，運用ディレクトリごとに作成します。 このファイルには，RMI 連携アダプターが使用するポート番号を設定します。
2	アダプター構成定義ファイル （AdaptorCompositionDefinition.xml）	○	運用ディレクトリごとに必ず作成します。 このファイルには，アダプターグループの構成や，標準提供アダプターで実行するコールバックの処理などを設定します。

（凡例）

○：必ず作成するファイルです。

△：必要に応じて作成するファイルです。

アダプター用定義ファイルについては，「9.　アダプター用定義ファイル」を参照してください。

3.6.2　アダプター用定義ファイルの作成のしかた

アダプター用定義ファイルは，「9.　アダプター用定義ファイル」の各ファイルの記述例を参考にして作成してください。なお，インストール時に提供されるサンプルファイルを使用して作成することをお勧めします。次のディレクトリに格納されているサンプルファイルをコピーして編集してください。

- 標準提供アダプター用のサンプルファイル
  ＜インストールディレクトリ＞¥samples¥＜fileまたはhttppacket＞¥conf¥xml¥
- 外部定義関数用のサンプルファイル
  ＜インストールディレクトリ＞¥samples¥external¥conf¥xml¥

また，サンプルファイルには英語版もあります。サンプルファイルの格納ディレクトリについては，「3.2.1　インストールディレクトリの構成」を参照してください。

なお，サンプルファイルに設定されている値は，「9.　アダプター用定義ファイル」の各ファイルのデフォルト値とは異なる場合がありますので，注意してください。

作成したアダプター用定義ファイルは，次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥xml¥

3.7　外部定義関数定義ファイルの作成

外部定義関数を使用する場合は，外部定義関数定義ファイルを作成して，外部定義関数を定義してください。

3.7.1　外部定義関数定義ファイルで設定できること

外部定義関数定義ファイルでは，外部定義関数の関数名や戻り値と，ユーザーが Java で作成したクラスとの関連づけを定義します。

外部定義関数定義ファイルについては，「10.　外部定義関数定義ファイル」を参照してください。

3.7.2　外部定義関数定義ファイルの作成のしかた

外部定義関数定義ファイルは，「10.　外部定義関数定義ファイル」の記述例を参考にして作成してください。なお，インストール時に提供されるサンプルファイルを使用して作成することをお勧めします。次のディレクトリに格納されているサンプルファイルをコピーして編集してください。

＜インストールディレクトリ＞¥samples¥external¥conf¥xml¥

なお，サンプルファイルに設定されている値は，「10.　外部定義関数定義ファイル」のデフォルト値とは異なる場合がありますので，注意してください。

作成した外部定義関数定義ファイルは，次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥xml¥

3.8　ダッシュボードの設定

標準提供アダプターでストリームデータの集計・分析結果をダッシュボードに出力する場合には，ダッシュボードの設定をしてください。ダッシュボードに出力しない場合は，設定する必要はありません。

ダッシュボードに出力する場合は，ストリームデータの集計・分析結果をダッシュボード表示用データとして出力するための設定が必要です。また，出力したダッシュボード表示用データを Flex Dashboard で表示するためには，Flex Dashboard の Dashboard Server と Dashboard Viewer の設定が必要です。ここでは，それぞれの設定方法について説明します。

なお，ダッシュボードへの出力については，「11.7　ダッシュボードへの出力」を参照してください。

3.8.1　ダッシュボード表示用データの出力の設定

ダッシュボード表示用データを出力するための設定は，アダプター用定義ファイルのアダプター構成定義ファイルに設定します。出力アダプターで，ダッシュボード出力コネクター定義を設定してください。

アダプター構成定義ファイルの作成については，「3.6　アダプター用定義ファイルの作成」，および「9. アダプター用定義ファイル」を参照してください。

3.8.2　Dashboard Server の設定

Dashboard Server は，ダッシュボード出力コネクターが出力したダッシュボード表示用データを取得するための Flex Dashboard のアプリケーションです。Dashboard Server は，Dashboard Viewer からの要求を受け付けて，ダッシュボード出力コネクターから取得したダッシュボード表示用データを Dashboard Viewer に送信します。

ここでは，Dashboard Server の設定方法について説明します。

(1)　環境変数の設定

次のシステム環境変数を設定します。

表 3-5　設定する環境変数（Windows の場合）

環境変数名	設定値
PRFSPOOL	＜インストールディレクトリ＞¥psb¥CC¥web¥redirector¥logs

表 3-6　設定する環境変数（Linux の場合）

環境変数名	設定値
PRFSPOOL	/opt/hitachi/sdp/psb/CC/web/redirector/logs
PATH※1	/opt/hitachi/sdp/psb/jdk/bin
	/opt/hitachi/sdp/psb/PRF/bin
LD_LIBRARY_PATH	/opt/hitachi/sdp/psb/PRF/lib
	/opt/hitachi/common/lib
CSCCFJ_SERVER_HOME	/opt/hitachi/sdp/psb/CC

環境変数名	設定値
COSMINEXUS_PSB_HOME※2	/opt/hitachi/sdp/psb
PRF_AUDITLOG_USE※2	0

注※1

環境変数「PATH」の先頭に「/opt/hitachi/sdp/psb/jdk/bin」を指定してください。

注※2

V01-50 以降の場合に設定します。

(2)　Dashboard Server の登録

Dashboard Server を使用するためには，Web サーバのサービスを登録して，Web コンテナサーバをセットアップします。

なお，Linux の場合，Web サーバのサービスの登録は不要です。

(a)　Windows の場合

- Web サーバのサービスの登録

  次のコマンドを実行して，Web サーバ（Hitachi Web Server）を Windows のサービスとして登録します。登録するサービス名は「Hitachi Web Server for uCSDPAF」（固定値）です。

  ＜インストールディレクトリ＞¥psb¥httpsd¥httpsd.exe -k install -n ”Hitachi Web Server for uCSDPAF”

- Web コンテナサーバのセットアップ

  次のコマンドを実行して，Dashboard Server を実行する Web コンテナサーバをセットアップします。セットアップするサーバ名は「uCSDPAF_Server」（固定値）です。

  ＜インストールディレクトリ＞¥psb¥CC¥web¥bin¥cjwebsetup.exe uCSDPAF_Server

  このセットアップ後に次のディレクトリが作成されます。

  ＜インストールディレクトリ＞¥psb¥CC¥web¥containers¥uCSDPAF_Server¥

Web サーバと Web コンテナサーバを設定するコマンドの実行例を次に示します。

```
C:¥> ＜インストールディレクトリ＞¥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:¥>＜インストールディレクトリ＞¥psb¥CC¥web¥bin¥cjwebsetup.exe uCSDPAF_Server

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

(b)　Linux の場合

V01-50 より前の場合

- Web コンテナサーバのセットアップ

  次のコマンドを実行して，Dashboard Server を実行する Web コンテナサーバをセットアップします。セットアップするサーバ名は「uCSDPAF_Server」（固定値）です。

  ＜インストールディレクトリ＞/psb/CC/web/bin/cjwebsetup.exe uCSDPAF_Server

  このセットアップ後に次のディレクトリが作成されます。

  ＜インストールディレクトリ＞/psb/CC/web/containers/uCSDPAF_Server/

V01-50 以降の場合

- J2EE サーバのセットアップ
  次のコマンドを実行して，J2EEサーバの環境をセットアップします。
  セットアップするJ2EEサーバ名は「uCSDPAF_Server」（固定値）です。
  /opt/hitachi/sdp/psb/CC/server/bin/cjsetup uCSDPAF_Server

(3)　Dashboard Server で使用するファイルの準備

Dashboard Server で使用するファイルを準備します。

なお，V01-50 より前の Linux の場合，インストーラが httpsd.conf ファイルの指定値を自動で更新するため，httpsd.conf ファイルの編集は不要です。

(a)　Windows の場合

- Dashboard Server で使用する httpsd.conf ファイルの編集
  <uCSDP-AF インストール先ディレクトリ>¥psb¥httpsd¥conf¥httpsd.conf ファイルを編集します。
  - Web ブラウザからアクセスする Web サーバのポート番号を設定します。httpsd.conf の Listen ディレクティブを 20430 に変更します。
  - httpsd.conf の末尾に，"Include ..¥CC¥web¥redirector¥mod_jk.conf"の行を追加します。

  設定例

```
##################################################################
##
## httpsd.conf - Cosminexus HTTP Server configuration file
##
##  All Rights Reserved. Copyright (C) 2000, 2012, Hitachi, Ltd.
##################################################################

Listen 20430
StartServers 5
MinSpareServers 5
MaxSpareServers 10
MaxClients 150
MaxRequestsPerChild 10000
      :
(略)
      :

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

- Dashboard Server で使用するファイルのコピーと編集
  次の表に示すコピー元のファイルをコピー先のディレクトリにコピーします。

コピー元のファイル	コピー先のディレクトリ	コピー後のファイル編集の要否
<インストールディレクトリ>¥samples¥httppacket¥web¥redirector¥workers.properties	<インストールディレクトリ>¥psb¥CC¥web¥redirector¥	×
<インストールディレクトリ>¥samples¥httppacket¥web¥redirector¥mod_jk.conf	<インストールディレクトリ>¥psb¥CC¥web¥redirector¥	×
<インストールディレクトリ>¥samples¥httppacket¥web¥containers	<インストールディレクトリ>¥psb¥CC¥web¥containers¥uCSDPAF_Server¥usrconf¥	×

コピー元のファイル	コピー先のディレクトリ	コピー後のファイル編集の要否
¥uCSDPAF_Server¥usrconf¥usrconf.cfg	＜インストールディレクトリ＞¥psb¥CC¥web¥containers¥uCSDPAF_Server¥usrconf¥	×
＜インストールディレクトリ＞¥samples¥httppacket¥web¥containers¥uCSDPAF_Server¥usrconf¥usrconf.properties	＜インストールディレクトリ＞¥psb¥CC¥web¥containers¥uCSDPAF_Server¥usrconf¥	○
＜インストールディレクトリ＞¥lib¥dashboard.war	＜インストールディレクトリ＞¥psb¥CC¥web¥containers¥uCSDPAF_Server¥webapps¥	×

（凡例）

○：コピー後にファイルを編集します。

×：コピー後にファイルを編集する必要はありません。

コピーしたファイルのうち，Dashboard Server のサーバ内設定ファイル（usrconf.properties）には，Dashboard Server がダッシュボード出力コネクターにアクセスして，ダッシュボード表示用データを更新するための間隔とリトライ回数を指定します。

サーバ内設定ファイルの編集については，「12.2　Dashboard Server のサーバ内設定ファイル（usrconf.properties）」を参照してください。

(b)　Linux の場合

V01-50 より前の場合

- Dashboard Server で使用する httpsd.conf ファイルのコピーと編集

  次の表に示すコピー元のファイルをコピー先のディレクトリにコピーします。

コピー元のファイル	コピー先のディレクトリ	コピー後のファイル編集の要否
＜インストールディレクトリ＞/samples/httppacket/web/redirector/workers.properties	＜インストールディレクトリ＞/psb/CC/web/redirector/	×
＜インストールディレクトリ＞/samples/httppacket/web/redirector/mod_jk.conf	＜インストールディレクトリ＞/psb/CC/web/redirector/	×
＜インストールディレクトリ＞/samples/httppacket/web/containers/uCSDPAF_Server/usrconf/usrconf.cfg	＜インストールディレクトリ＞/psb/CC/web/containers/uCSDPAF_Server/usrconf/	×
＜インストールディレクトリ＞/samples/httppacket/web/containers/uCSDPAF_Server/usrconf/usrconf.properties	＜インストールディレクトリ＞/psb/CC/web/containers/uCSDPAF_Server/usrconf/	○
＜インストールディレクトリ＞/lib/dashboard.war	＜インストールディレクトリ＞/psb/CC/web/containers/uCSDPAF_Server/webapps/	×

（凡例）

○：コピー後にファイルを編集します。

×：コピー後にファイルを編集する必要はありません。

コピーしたファイルのうち，**Dashboard Server のサーバ内設定ファイル**（usrconf.properties）には，Dashboard Server がダッシュボード出力コネクターにアクセスして，ダッシュボード表示用データを更新するための間隔とリトライ回数を指定します。

サーバ内設定ファイルの編集については，「12.2　Dashboard Server のサーバ内設定ファイル(usrconf.properties)」を参照してください。

V01-50 以降の場合

- **Dashboard Server で使用する httpsd.conf ファイルの編集**

  <uCSDP-AF インストール先ディレクトリ>/psb/httpsd/conf/httpsd.conf ファイルを編集します。

  - Web ブラウザからアクセスする Web サーバのポート番号を設定します。また，httpsd.conf の Listen ディレクティブを 20430 に変更します。
  - User ディレクティブに指定されているユーザーが属するグループを Group ディレクティブに指定します。
  - httpsd.conf の末尾に，"Include ../CC/web/redirector/mod_jk.conf"の行を追加します。

  設定例

```
##################################################################
##
## httpsd.conf - Cosminexus HTTP Server configuration file
##
##  All Rights Reserved. Copyright (C) 2000, 2012, Hitachi, Ltd.
##################################################################

Listen 20430
StartServers 5
MinSpareServers 5
MaxSpareServers 10
MaxClients 150
MaxRequestsPerChild 10000
      :
(略)
      :
User nobody
Group nobody
      :
(略)
      :

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

- **Dashboard Server で使用するファイルのコピーと編集**

ファイルのコピー

次の表に示すコピー元のファイルをコピー先のディレクトリにコピーします。

コピー元のファイル	コピー先のディレクトリ	コピー後のファイル編集の要否
＜インストールディレクトリ＞/samples/httppacket/web/redirector/workers.properties	＜インストールディレクトリ＞/psb/CC/web/redirector/	×
＜インストールディレクトリ＞/samples/httppacket/web/redirector/mod_jk.conf	＜インストールディレクトリ＞/psb/CC/web/redirector/	×
＜インストールディレクトリ＞/samples/httppacket/web/containers/uCSDPAF_Server/usrconf/usrconf.cfg	＜インストールディレクトリ＞/psb/CC/server/usrconf/ejb/uCSDPAF_Server	×

コピー元のファイル	コピー先のディレクトリ	コピー後のファイル編集の要否
＜インストールディレクトリ＞/samples/httppacket/web/containers/uCSDPAF_Server/usrconf/usrconf.properties	＜インストールディレクトリ＞/psb/CC/server/usrconf/ejb/uCSDPAF_Server	○

（凡例）

○：コピー後にファイルを編集します。

×：コピー後にファイルを編集する必要はありません。

コピーしたファイルのうち，Dashboard Server のサーバ内設定ファイル（usrconf.properties）には，Dashboard Server がダッシュボード出力コネクターにアクセスして，ダッシュボード表示用データを更新するための間隔とリトライ回数を指定します。

サーバ内設定ファイルの編集については，「12.2　Dashboard Server のサーバ内設定ファイル（usrconf.properties）」を参照してください。

- ファイルの編集

  /opt/hitachi/sdp/psb/CC/admin/usrconf 下の usrconf.properties に対し，次のプロパティを追加します。

  ```
  ejbserver.rmi.naming.port=20432
  ```

- J2EE サーバの開始と J2EE アプリケーションのデプロイ

  J2EE サーバの開始

  次のコマンドを実行して，J2EEサーバを開始します。

  セットアップするJ2EEサーバ名は「uCSDPAF_Server」（固定値）です。

  ```
  # /opt/hitachi/sdp/psb/CC/server/bin/cjstartsv uCSDPAF_Server -nosecurity
  ```

  J2EE アプリケーションのデプロイ

  起動した J2EE サーバに，J2EE アプリケーション（dashboard）をインポートします。次のコマンドライン固定です。

  ```
  /opt/hitachi/sdp/psb/CC/admin/bin/cjimportapp uCSDPAF_Server -f /opt/hitachi/sdp/lib/dashboard.ear
  ```

3.8.3　Dashboard Viewer の設定

Dashboard Viewer は，Web ブラウザ上でダッシュボードを閲覧するための Flex Dashboard のアプリケーションです。Dashboard Viewer は，Dashboard Server に定期的にアクセスしてダッシュボード表示用データを更新し，リアルタイムなストリームデータの集計・分析結果を表示します。

ここでは，Dashboard Viewer の設定方法について説明します。

(1)　Dashboard Viewer の画面の編集

Dashboard Viewer で表示するダッシュボードの画面は，ユーザーが編集できます。ダッシュボードの画面を編集することで，特定のデータからグラフを作成して表示したり，グラフの表示レイアウトを変更したりできます。

Dashboard Viewer の画面編集用ファイルの作成については，「12.3.2　Dashboard Viewer の画面編集用ファイルの詳細」を参照してください。

なお，ダッシュボードの画面を編集するためには，Dashboard Viewer に表示されるダッシュボードの画面の構成や，画面の構成要素を理解しておく必要があります。ダッシュボードの画面については，「12.3.1 Dashboard Viewer の画面構成」を参照してください。

(2) Dashboard Viewer の URL の連絡

Dashboard Viewer の画面の編集が完了したら，ダッシュボード画面を閲覧するユーザーに，Dashboard Viewer の URL を連絡してください。Dashboard Viewer の URL を次に示します。

http://＜ホスト名＞:＜ポート番号＞/dashboard/viewer.jsp?layout=＜画面名※＞

注※

Dashboard Viewer の画面編集用ファイルのファイル名（＜画面名＞.xml）に指定した画面名です。

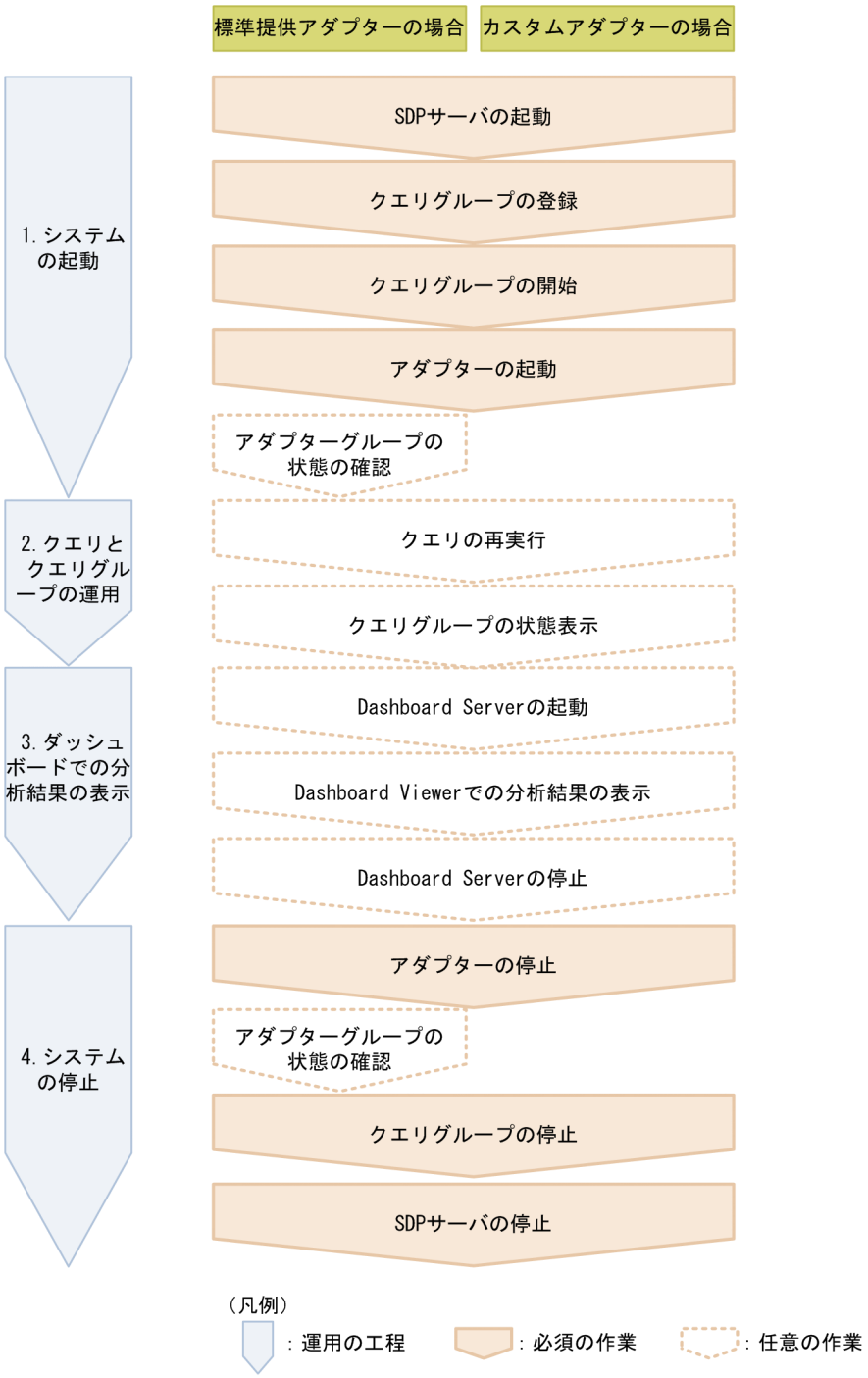
4 システムの運用

この章では，ストリームデータ処理システムの運用の流れ，システムの起動・停止手順，およびクエリとクエリグループの運用について説明します。また，ダッシュボードで集計・分析結果を表示する場合の手順についても説明します。

4.1　システムの運用の流れ

システムの運用では，システムの起動・停止，クエリグループの運用などをします。ストリームデータ処理システムの運用の流れを次の図に示します。

図 4-1　システムの運用の流れ



それぞれの作業について，次に説明します。

(1)　システムの起動

SDP サーバ，クエリグループ，およびアダプターを次の順序で起動または開始して，システムを起動します。

1. SDP サーバの起動

   SDP サーバを起動します。SDP サーバの起動方法については，「4.2.1　SDP サーバの起動」を参照してください。

2. クエリグループの登録

   クエリグループを SDP サーバに登録します。クエリグループの登録方法については，「4.2.2　クエリグループの登録」を参照してください。

3. クエリグループの開始

   SDP サーバに登録済みのクエリグループを開始します。クエリグループの開始方法については，「4.2.3　クエリグループの開始」を参照してください。

4. アダプターの起動

   標準アダプターまたはカスタムアダプターを起動します。標準アダプターの起動方法については，「4.2.4　アダプターの起動（標準提供アダプター）」を参照してください。カスタムアダプターの起動方法については，「4.2.5　アダプターの起動（カスタムアダプター）」を参照してください。

5. アダプターグループの状態の確認

   標準提供アダプターの場合に，アダプターグループが正常に起動しているかどうかを確認します。アダプターグループの状態の確認方法については，「4.2.6　アダプターグループの状態の確認」を参照してください。

(2)　クエリとクエリグループの運用

必要に応じて，次の操作をしてクエリとクエリグループを運用します。

1. クエリの再実行

   実行済みのクエリを再度実行します。クエリの再実行の方法については，「4.3.1　クエリの再実行」を参照してください。

2. クエリグループの状態表示

   クエリグループの状態や入出力ストリームの滞留状態などを表示します。クエリグループの状態表示の方法については，「4.3.2　クエリグループの状態表示」を参照してください。

(3)　ダッシュボードでの分析結果の表示

標準提供アダプターでストリームデータの集計・分析結果をダッシュボードに出力する場合，次の手順で表示します。

1. Dashboard Server の起動

   PRF デーモン，Web コンテナサーバ，および Web サーバを起動して，Dashboard Server を起動します。Dashboard Server の起動方法については，「4.5.1　Dashboard Server の起動」を参照してください。

2. Dashboard Viewer での分析結果の表示

   Dashboard Viewer を使用してダッシュボードの画面を開き，分析結果を表示します。Dashboard Viewer での分析結果の表示方法については，「4.5.2　Dashboard Viewer での分析結果の表示」を参照してください。

3. Dashboard Server の停止

Web サーバ，Web コンテナサーバ，および PRF デーモンを停止して，Dashboard Server を停止します。Dashboard Server の停止方法については，「4.5.3　Dashboard Server の停止」を参照してください。

(4)　システムの停止

アダプター，クエリグループ，および SDP サーバを次の順序で停止して，システムを停止します。

1. アダプターの停止

   標準提供アダプターまたはカスタムアダプターを停止します。なお，RMI 連携のカスタムアダプターの場合は，Stream Data Platform - AF のコマンドでは終了できないため，カスタムアダプターの中で終了処理を実装してください。

   標準提供アダプターの停止方法については，「4.4.1　アダプターの停止（標準提供アダプター)」を参照してください。カスタムアダプターの停止方法については，「4.4.2　アダプターの停止（カスタムアダプター)」を参照してください。

2. アダプターグループの状態の確認

   アダプターグループが正常に停止しているかどうかを確認します。アダプターグループの状態の確認方法については，「4.2.6　アダプターグループの状態の確認」を参照してください。

3. クエリグループの停止

   クエリグループを停止します。クエリグループの停止方法については，「4.4.3　クエリグループの停止」を参照してください。

4. SDP サーバの停止

   SDP サーバを停止します。SDP サーバの停止方法については，「4.4.4　SDP サーバの停止」を参照してください。

4.2　システムの起動

ここでは，システムの起動について説明します。システムは次の順序で起動します。

4.2.1　SDP サーバの起動

sdpstart コマンドを使用して SDP サーバを起動します。コマンドの実行例を次に示します。

```
＜運用ディレクトリ＞¥bin¥sdpstart
```

sdpstart コマンドについては，「7.　コマンド」の「sdpstart（SDP サーバの起動）」を参照してください。

4.2.2　クエリグループの登録

クエリグループを SDP サーバに登録します。クエリグループを登録するには，構築時に作成した定義ファイルを指定して sdpcql コマンドを実行します。定義ファイルの作成については，「3.4　SDP サーバ用定義ファイルの作成」，および「3.5　クエリ定義ファイルの作成」を参照してください。

クエリグループの登録に必要なファイルを次の表に示します。なお，ストリーム用プロパティファイルについては，必要に応じて登録します。

- クエリ定義ファイル
- クエリグループ用プロパティファイル
- ストリーム用プロパティファイル

コマンドの実行例を次に示します。この例では，クエリグループ用プロパティファイル名を QueryGroupSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpcql QueryGroupSample
```

なお，クエリグループの登録と開始を同時にすることもできます。クエリグループの登録と開始を同時にする場合は，sdpcql コマンドに，-autostart オプションを指定して実行します。

クエリグループの登録と開始を同時にする場合のコマンドの実行例を次に示します。この例では，クエリグループ用プロパティファイル名を QueryGroupSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpcql -autostart QueryGroupSample
```

sdpcql コマンドについては，「7.　コマンド」の「sdpcql（クエリグループの登録）」を参照してください。

4.2.3　クエリグループの開始

SDP サーバに登録済みのクエリグループを開始します。

クエリグループを開始すると，アダプターからの送信データを受け付け，クエリが実行されるようになります。

クエリグループを開始するには，sdpcqlstart コマンドを実行します。コマンドの実行例を次に示します。この例では，クエリグループ用プロパティファイル名を QueryGroupSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpcqlstart QueryGroupSample
```

なお，クエリグループの登録と開始を同時にする場合は，sdpcql コマンドに-autostart オプションを指定して実行します。

sdpcqlstart コマンド，または sdpcql コマンドについては，「7.　コマンド」の「sdpcqlstart（クエリグループの開始)」，または「sdpcql（クエリグループの登録)」を参照してください。

4.2.4　アダプターの起動（標準提供アダプター）

標準提供アダプターを起動します。アダプターの起動前にクエリグループの登録および開始が完了している必要があります。

インプロセス連携の場合と RMI 連携の場合では，使用するコマンドが異なります。それぞれの場合の起動方法について説明します。

(1)　インプロセス連携の場合

sdpstartinpro コマンドを実行して標準提供アダプターを起動します。

コマンドの引数には，アダプター構成定義ファイルのインプロセスグループ定義の name 属性に指定したアダプターグループ名を指定します。インプロセスグループ定義については，「9.7.1　インプロセスグループ定義」を参照してください。

コマンドの実行例を次に示します。この例では，アダプターグループ名を InprocessAPSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpstartinpro InprocessAPSample
```

sdpstartinpro コマンドについては，「7.　コマンド」の「sdpstartinpro（インプロセス連携アダプターの起動)」を参照してください。

(2)　RMI 連携の場合

sdpstartap コマンドを実行してアダプターを起動します。

コマンドの引数には，アダプター構成定義ファイルの RMI グループ定義の name 属性に指定したアダプターグループ名を指定します。RMI グループ定義については，「9.7.2　RMI グループ定義」を参照してください。

コマンドの実行例を次に示します。この例では，アダプターグループ名を RMIGroupSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpstartap jp.co.Hitachi.soft.sdp.adaptor.AdaptorManager
RMIGroupSample
```

sdpstartap コマンドについては，「7.　コマンド」の「sdpstartap（RMI 連携アダプターの起動)」を参照してください。

(3)　アダプターの状態遷移

標準提供アダプターを起動すると，アダプターグループがオンライン状態となり，SDP サーバとのデータの送受信ができる状態になります。アダプターグループの状態には，次の 2 種類があります。

- 未起動状態
  アダプターが起動していない状態です。この状態のときに定義ファイルの作成や修正ができます。インプロセス連携の場合は sdpstopinpro コマンド，RMI 連携の場合は sdpstopap コマンドを実行してアダプターグループを停止すると，未起動状態になります。
- オンライン状態

アダプターが起動し，データの入出力ができる状態です。インプロセス連携の場合は sdpstartinpro コマンド，RMI 連携の場合は sdpstartap コマンドを実行してアダプターグループを起動すると，オンライン状態になります。

また，ファイル入力の場合に，バッチ処理モードでレコードの入力が完了したとき，または処理を続行できない障害が発生したときに，未起動状態になります。このとき，インプロセス連携の場合は，必ず sdpstopinpro コマンドを実行してください。

4.2.5　アダプターの起動（カスタムアダプター）

カスタムアダプターを起動します。アダプターの起動前にクエリグループの登録および開始が完了している必要があります。

インプロセス連携の場合と RMI 連携の場合では，使用するコマンドが異なります。それぞれの場合の起動方法について説明します。

(1)　インプロセス連携の場合

sdpstartinpro コマンドを実行してアダプターを起動します。

コマンドの引数には，インプロセス連携用プロパティファイル名に設定したカスタムアダプター名を指定します。インプロセス連携用プロパティファイルについては，「8.9　インプロセス連携用プロパティファイル」を参照してください。

コマンドの実行例を次に示します。この例では，カスタムアダプター名を InproAppSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpstartinpro InproAppSample
```

sdpstartinpro コマンドについては，「7.　コマンド」の「sdpstartinpro（インプロセス連携アダプターの起動）」を参照してください。

(2)　RMI 連携の場合

sdpstartap コマンドを実行してアダプターを起動します。

コマンドの引数には，RMI 連携用 JavaVM オプションファイルのパス名，Java のアプリケーションのクラス名，main メソッドに渡される引数などを指定します。RMI 連携用 JavaVM オプションファイルについては，「8.5　RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）」を参照してください。

コマンドの実行例を次に示します。この例では，Java のアプリケーションのクラス名を AppSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpstartap -clientcfg conf¥jvm_client_options.cfg AppSample
```

sdpstartap コマンドについては，「7.　コマンド」の「sdpstartap（RMI 連携アダプターの起動）」を参照してください。

4.2.6　アダプターグループの状態の確認

標準提供アダプターを起動または停止したあと，アダプターグループの起動状態およびデータ処理の状態を確認し，標準提供アダプターが正常に起動または停止しているか，およびデータの送受信が正常に行われているかを確認します。

アダプターグループの起動・停止の確認，およびデータ処理の確認の手順について説明します。

(1)　アダプターグループの起動・停止の確認

コマンドを実行して標準提供アダプターを起動または停止したあと，標準提供アダプターが正常に起動または停止したかどうかを確認します。アダプターグループの起動状態は，エラーメッセージおよびコマンドの戻り値で確認します。

- エラーメッセージの確認

  次の二つの項目を参照して，E（Error）または W（Warning）のメッセージがないことを確認します。

  - 標準提供アダプターの起動または停止のコマンドを実行したコンソール
  - ＜運用ディレクトリ＞¥logs に出力されるアダプターのログファイル

  なお，メッセージ KFSP56101-E が出力された場合，アダプター構成定義ファイルの XML スキーマの検証でエラーが起きていることを示しています。この場合，メッセージの詳細情報に表示されているエラー識別子とメッセージを基に，エラーの原因を特定します。識別子の一覧は，W3C による XML Schema の仕様書の Part1: Structures Appendices C に記載されています。

  例えば，インプロセスグループ定義の InprocessGroupDefinition タグに name 属性を記述しなかった場合，メッセージ KFSP56101-E の詳細情報は次のようになります。

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

  この詳細情報の場合，識別子は”cvc-complex-type.4”となります。これは，「Validation Rule: Element Locally Valid (Complex Type)」の 4 に記載された規則に違反していることを示します。

- コマンドの戻り値の確認

  標準提供アダプターの起動または停止のコマンドを実行したコンソールで次のコマンドを実行し，結果が「0」となることを確認します。

  ```
  echo %ERRORLEVEL%
  ```

(2)　データ送受信の結果の確認

標準提供アダプターの起動の場合，SDP サーバとアダプターとのデータの送受信が正常に行われていることを次のことから確認します。

- エラーメッセージの確認

  次の二つの項目を参照して，メッセージ ID に E または W があるメッセージがないことを確認します。

  - 標準提供アダプターの起動または SDP サーバの起動コマンドを実行したコンソール
  - ＜運用ディレクトリ＞¥logs に出力されるアダプターおよび SDP サーバのメッセージログファイル

    ログファイルについては，「6.2(5)　ログファイル」，および「6.3.1　ログファイルの詳細」を参照してください。

- データ送受信の結果の確認

  入力元ファイルとタプルログの結果を突き合わせて，入力アダプターから送信したタプルが SDP サーバに送信されているかを確認します。また，クエリとファイル出力の出力データから期待通りの結果が出力されていることを確認します。

  タプルログについては，「6.2(9)　タプルログ」，および「6.3.4　タプルログの詳細」を参照してください。

4.3　クエリとクエリグループの運用

クエリとクエリグループの運用では，クエリの再実行およびクエリグループの状態表示をします。クエリの再実行およびクエリグループの状態表示は任意のタイミングで実行します。

4.3.1　クエリの再実行

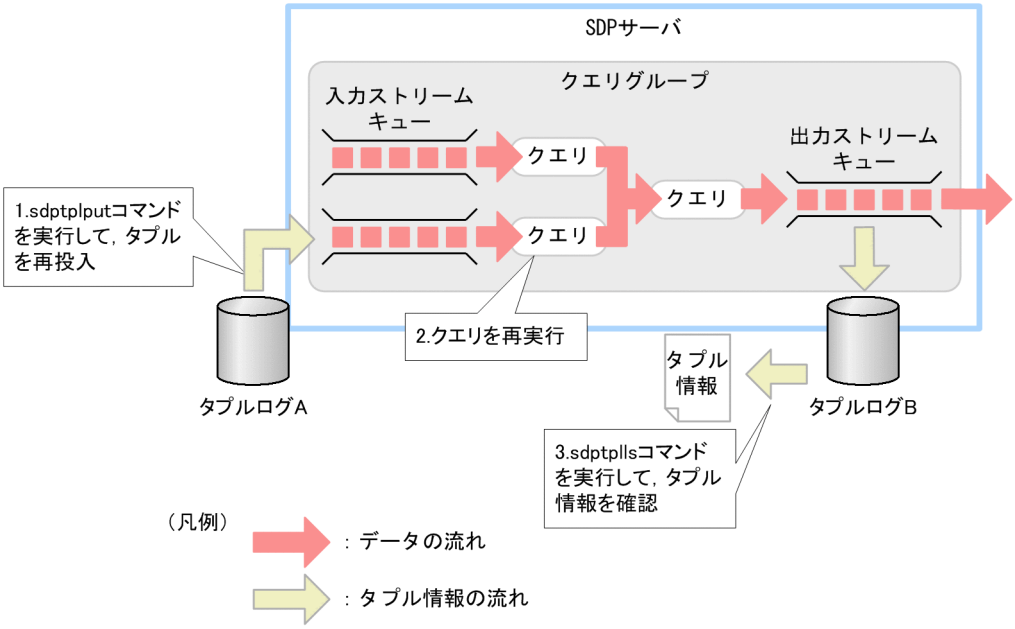
トラブル発生時にトラブルの現象を再現したい場合や，集計・分析結果を再度確認したい場合など，特定の時間だけデータを再入力してクエリを実行したいときにクエリの再実行をします。

クエリの再実行には，sdptplput コマンドを使用します。再実行する対象のデータとして，タプルログファイルに取得した入力タプルの情報を使用します。コマンドを実行すると，タプルログファイルに取得したタプル情報から，タプルが入力ストリームキューに再投入されます。また，タプルの再投入が終了したあと，すべての入力ストリームキューに対して putEnd メソッドが実行され，クエリグループが初期化されます。

クエリの再実行の結果は，sdptplls コマンドを使用してタプルログに取得した出力ストリームキューのタプル情報を表示して確認します。

クエリを再実行し，実行結果を確認する場合のデータの流れを次の図に示します。

図 4-2　クエリの再実行のデータの流れ



1. sdptplput コマンドを実行して（タプルログ A を指定），タプルを再投入します。
2. 投入したタプルに対し，クエリが再実行されます。
3. sdptplls コマンドを実行して（タプルログ B を指定），タプル情報を確認します。

クエリを再実行できる範囲，クエリを再実行する手順，およびクエリを再実行するときの注意事項について説明します。

(1)　クエリを再実行できる範囲

クエリの再実行は，サーバモードの場合とデータソースモードの場合で再実行できる範囲が異なります。再実行できる範囲を次に示します。

- サーバモードの場合
  クエリの実行
- データソースモードの場合
  タイムスタンプ調整からクエリの実行まで

タイムスタンプ調整の機能については，「11.8　タプルのタイムスタンプの調整」を参照してください。

(2)　クエリを再実行する手順

取得したタプルログファイルを使用してクエリを再実行します。クエリの再実行は，タプルログを取得した環境と異なる環境でも実行できますが，デフォルト文字コードが同じ環境で実行してください。

なお，この例では，クエリの再実行をする環境はタプルログを取得した環境と異なる環境とします。

1. **クエリを再実行する環境に必要なファイルをコピーします。**
   再実行するクエリグループのクエリ定義ファイル，クエリグループ用プロパティファイルおよびタプルログファイルを，タプルを取得した環境から再実行する環境にコピーします。
2. **タプルログを取得するための設定をします。**
   クエリグループ用プロパティファイルの tpl.outputTrigger パラメーターに BUFFER を指定し，出力ストリームキューのタプルログを取得するようにします。また，サーバモードで取得したタプルログファイルを使用する場合だけ，stream.tupleLogMode パラメーターに true を指定します。
3. **再実行するクエリグループを登録します。**
   sdpcql コマンドで再実行するクエリグループを登録します。
4. **再実行するクエリグループを開始します。**
   sdpcqlstart コマンドで再実行するクエリグループを開始します。
5. **クエリを再実行します。**
   sdptplput コマンドでタプルログファイルからタプルを再投入し，クエリを再実行します。
6. **結果を確認します。**
   出力ストリームキューのタプルログファイルの内容を sdptplls コマンドで表示し，結果を確認します。

sdpcql コマンド，sdpcqlstart コマンド，sdptplput コマンド，または sdptplls コマンドについては，「7.　コマンド」の「sdpcql（クエリグループの登録)」，「sdpcqlstart（クエリグループの開始)」，「sdptplput（タプルの再投入)」，または「sdptplls（タプル情報の表示)」を参照してください。

(3)　クエリを再実行するときの注意事項

- アダプターが最初に投入したタプル以外のタプルから再投入してクエリを再実行した場合，その結果はアダプターを使用してクエリを実行した結果と異なることがあります。
- sdptplput コマンドは，タプルを再投入する間隔を再現しないため，入力ストリームキューでキューあふれが発生する場合があります。キューあふれが発生した場合の動作は，サーバモードまたはデータソースモードで異なります。
  - サーバモードの場合
    再投入に失敗したタプルを-interval オプションに指定した間隔で再投入します。このとき，メッセージ KFSP42005-E および KFSP52003-E が出力されますが，クエリの再実行には影響しません。
  - データソースモードの場合

キューあふれが発生するとクエリグループが閉塞し，クエリの再実行に失敗します。そのため，-interval オプションおよび-count オプションを使用してタプルを再投入する間隔と個数を調整してください。

- sdptplput コマンドの実行中はアダプターを起動しないでください。アダプターを起動した場合，次のようになります。
  - 入力アダプターを起動した場合

    再投入したタプルと入力アダプターが投入したタプルが混在してしまい，クエリを再実行した結果が入力アダプターを使用してクエリを実行した結果と異なります。
  - 出力アダプターを起動した場合

    出力ストリームキューのキューあふれが発生し，クエリグループが閉塞するおそれがあります。

4.3.2　クエリグループの状態表示

SDP サーバに登録されているクエリグループの状態は，sdpls コマンドで取得できます。sdpls コマンドを実行すると，クエリグループの状態や入出力ストリームキューの滞留状態などが表示されます。

コマンドの実行例を次に示します。この例では，クエリグループ用プロパティファイルを QueryGroupSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpls QueryGroupSample
```

sdpls コマンドについては，「7.　コマンド」の「sdpls（クエリグループの状態表示)」を参照してください。

4.4　システムの停止

ここでは，システムの停止について説明します。アダプターグループの状態の確認方法については，「4.2.6(1)　アダプターグループの起動・停止の確認」を参照してください。

4.4.1　アダプターの停止（標準提供アダプター）

標準提供アダプターを停止します。アダプターを停止するとアダプターグループが未起動状態となります。アダプターグループの状態については，「4.2.4(3)　アダプターの状態遷移」を参照してください。

インプロセス連携の場合と RMI 連携の場合では，使用するコマンドが異なります。それぞれの場合の停止方法について説明します。

(1)　インプロセス連携の場合

sdpstopinpro コマンドを実行してアダプターを停止します。

コマンドの引数には，アダプター構成定義ファイルのインプロセスグループ定義の name 属性に指定したアダプターグループ名を指定します。インプロセスグループ定義については，「9.7.1　インプロセスグループ定義」を参照してください。

コマンドの実行例を次に示します。この例では，アダプターグループ名を InprocessAPSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpstopinpro InprocessAPSample
```

sdpstopinpro コマンドについては，「7.　コマンド」の「sdpstopinpro（インプロセス連携アダプターの停止)」を参照してください。

(2)　RMI 連携の場合

sdpstopap コマンドを実行してアダプターを停止します。

コマンドの引数には，アダプター構成定義ファイルの RMI グループ定義の name 属性に指定したアダプターグループ名を指定します。RMI グループ定義については，「9.7.2　RMI グループ定義」を参照してください。

コマンドの実行例を次に示します。この例では，アダプターグループ名を RMIGroupSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpstopap RMIGroupSample
```

sdpstopap コマンドについては，「7.　コマンド」の「sdpstopap（RMI 連携アダプターの停止)」を参照してください。

4.4.2　アダプターの停止（カスタムアダプター）

カスタムアダプターを停止します。インプロセス連携の場合と RMI 連携の場合では停止方法が異なります。それぞれの場合の停止方法について説明します。

(1)　インプロセス連携の場合

sdpstopinpro コマンドを実行してアダプターを停止します。

コマンドの引数には，インプロセス連携用プロパティファイル名に指定したカスタムアダプター名を指定します。

インプロセス連携用プロパティファイルについては，「8.9　インプロセス連携用プロパティファイル」を参照してください。

コマンドの実行例を次に示します。この例では，カスタムアダプター名を InproAppSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpstopinpro InproAppSample
```

sdpstopinpro コマンドについては，「7.　コマンド」の「sdpstopinpro（インプロセス連携アダプターの停止)」を参照してください。

(2)　RMI 連携の場合

RMI 連携の場合，Stream Data Platform - AF のコマンドでは終了できないため，カスタムアダプターの中で終了処理を実装してください。

4.4.3　クエリグループの停止

SDP サーバに登録されているクエリグループを停止します。クエリグループの停止前にアダプターの停止が完了している必要があります。

一日のうちの特定の時間だけ動かすクエリグループを停止させる場合や，連続で稼働しているクエリグループをメンテナンスする場合に実行します。クエリグループを停止すると，アダプターからの送信データを受け付けないようになります。

クエリグループの停止には次の二つの方法があります。

- **正常停止**
- **強制停止**

ここでは，正常停止，強制停止の詳細，およびクエリグループの停止時のリレーションの破棄について説明します。

(1)　クエリグループの正常停止

日常運用の中でクエリグループを停止する場合は，正常停止します。正常停止を実行すると，入力ストリームキュー内のタプルの処理が完了してから，クエリグループを停止します。

クエリグループを正常停止するには，sdpcqlstop コマンドを実行します。コマンドの実行例を次に示します。この例では，クエリグループ用プロパティファイルを QueryGroupSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpcqlstop QueryGroupSample
```

sdpcqlstop コマンドについては，「7.　コマンド」の「sdpcqlstop（クエリグループの停止)」を参照してください。

(2)　クエリグループの強制停止

クエリグループの実行中にトラブルが発生したときなど，直ちにクエリグループを停止する場合に強制停止します。強制停止を実行すると，入力ストリームキュー内のタプルを破棄して，クエリグループを停止します。

クエリグループを強制停止するには，sdpcqlstop コマンドに-force オプションを指定して実行します。コマンドの実行例を次に示します。この例では，クエリグループ用プロパティファイルを QueryGroupSample としています。

```
<運用ディレクトリ>¥bin¥sdpcqlstop -force QueryGroupSample
```

sdpcqlstop コマンドについては，「7.　コマンド」の「sdpcqlstop（クエリグループの停止)」を参照してください。

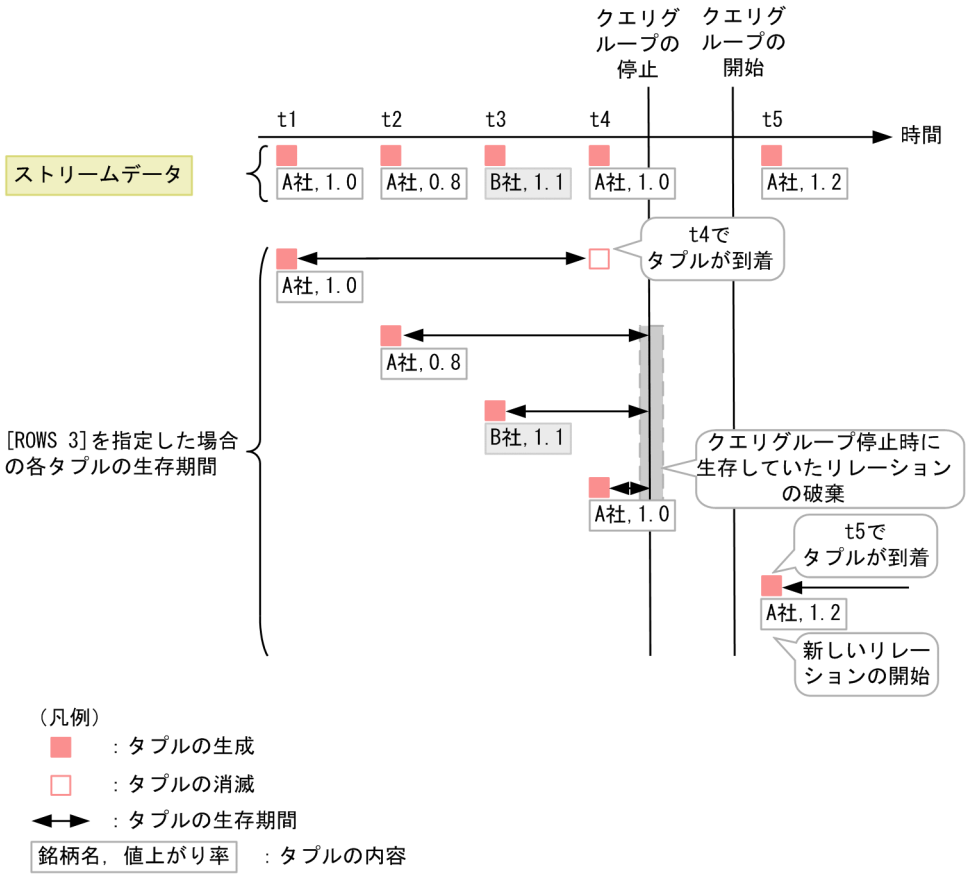
(3)　リレーションの破棄

リレーションとは，ウィンドウ演算によって取り出されたタプルの集合です。このタプルの集合が，データ操作演算の対象となります。ウィンドウ演算とは，ストリームデータに対し，集計・分析を行う範囲を指定する演算で，CQL で定義します。CQL でのウィンドウ演算については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

クエリグループを停止したときに残っていた入力リレーションは，停止処理の中で破棄されます。また，次にクエリグループを開始した際には前回の入力リレーションを引き継がないで，新たに入力リレーションを開始します。

クエリグループの停止時の入力リレーションの状態を次の図に示します。

図 4-3　クエリグループの停止時の入力リレーションの状態



この図では，ウィンドウ演算に[ROWS 3]を指定した場合の，クエリグループの停止時の入力リレーションの状態を示しています。[ROWS 3]は，入力リレーション内に同時に生存するタプル数が 3 であることを

示します。図の横軸は時間軸です。右に行くほど時間が経過しています。t1～t5 は，タプルが到着した時刻を示します。また，各タプルは，「銘柄名，値上がり率」で構成されています。

クエリグループを停止したとき，「(A 社,0.8)」，「(B 社,1.1)」および「(A 社,1.0)」のタプルが生存する入力リレーションが破棄されます。また，クエリグループが開始されたあとの t5 で到着した「(A 社,1.2)」のタプルが生成され，新しい入力リレーションが開始されます。

4.4.4　SDP サーバの停止

sdpstop コマンドを使用して SDP サーバを停止します。

SDP サーバの停止には次の二つの方法があります。

- **正常停止**
- **強制停止**

(1)　SDP サーバの正常停止

実行中のクエリグループおよびインプロセス連携のカスタムアダプターをすべて正常停止してから，SDP サーバを停止します。

インプロセス連携のカスタムアダプターが起動している場合，SDP サーバ側から StreamInprocessUP インタフェースを実装したクラスの stop メソッドが呼び出され，インプロセス連携のカスタムアダプターの終了処理が行われます。

SDP サーバを正常停止するには，sdpstop コマンドを実行します。コマンドの実行例を次に示します。

```
＜運用ディレクトリ＞¥bin¥sdpstop
```

sdpstop コマンドについては，「7.　コマンド」の「sdpstop（SDP サーバの停止)」を参照してください。

(2)　SDP サーバの強制停止

実行中のクエリグループおよびインプロセス連携のカスタムアダプターの停止を待たないで，直ちに SDP サーバを停止します。入力ストリームキュー内のタプルはすべて破棄されます。

SDP サーバを強制停止するには，sdpstop コマンドに-force オプションを指定して実行します。コマンドの実行例を次に示します。

```
＜運用ディレクトリ＞¥bin¥sdpstop -force
```

sdpstop コマンドについては，「7.　コマンド」の「sdpstop（SDP サーバの停止)」を参照してください。

4.5　ダッシュボードでの分析結果の表示

ここでは，標準提供アダプターで，ストリームデータの集計・分析結果をダッシュボードに表示する手順について説明します。

4.5.1　Dashboard Server の起動

Dashboard Server を起動するには，次の順序で，PRF デーモン（トレース情報を出力するプロセス），Web コンテナサーバ，および Web サーバを起動します。

Windows の場合

1. cprfstart コマンドを実行して，PRF デーモンを起動します。
   ＜インストールディレクトリ＞¥psb¥PRF¥bin¥cprfstart.exe
2. cjstartweb コマンドを実行して，Web コンテナサーバを起動します。
   ＜インストールディレクトリ＞¥psb¥CC¥web¥bin¥cjstartweb.exe uCSDPAF_Server
   なお，コマンドを実行したコマンドプロンプトは，Web コンテナサーバが停止するまで制御が戻りません。
3. httpsd コマンドを実行して，Web サーバを起動します。
   ＜インストールディレクトリ＞¥psb¥httpsd¥httpsd.exe -k start -n "Hitachi Web Server for uCSDPAF"
   また，次のコマンドでも Web サーバを起動できます。
   net start "Hitachi Web Server for uCSDPAF"

Linux の場合

- V01-50 より前

1. cprfstart コマンドを実行して，PRF デーモンを起動します。
   /opt/hitachi/sdp/psb/PRF/bin/cprfstart
2. cjstartweb コマンドを実行して，Web コンテナサーバを起動します。
   /opt/hitachi/sdp/psb/CC/web/bin/cjstartweb uCSDPAF_Server
3. httpsd コマンドを実行して，Web サーバを起動します。
   /opt/hitachi/sdp/psb/httpsd/sbin/httpsd -d /opt/hitachi/sdp/psb/httpsd -R /opt/hitachi/sdp/psb/httpsd/libexec

- V01-50 以降

1. cprfstart コマンドを実行して，PRF デーモンを起動します。
   /opt/hitachi/sdp/psb/PRF/bin/cprfstart
2. cjstartsv コマンドを実行して，J2EE サーバを起動します。
   /opt/hitachi/sdp/psb/CC/server/bin/cjstartsv uCSDPAF_Server -nosecurity
3. cjstartapp コマンドを実行して，デプロイした J2EE アプリケーション（dashboard）を開始します。
   /opt/hitachi/sdp/psb/CC/admin/bin/cjstartapp uCSDPAF_Server -name dashboard
4. httpsd コマンドを実行して，Web サーバを起動します。
   /opt/hitachi/sdp/psb/httpsd/sbin/httpsd -d /opt/hitachi/sdp/psb/httpsd -R /opt/hitachi/sdp/psb/httpsd/libexec

Windows の場合の実行例を次に示します。

```
＜インストールディレクトリ＞¥psb¥PRF¥bin¥cprfstart.exe

 Mon Jul 22 19:17:37 2013:KFCT73410-I 4092 2680:now starting cprfd.
 Mon Jul 22 19:17:38 2013:KFCT73412-I 4092 2680:cprfd is now online.

＜インストールディレクトリ＞¥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 = rBEH_1x9H)
KDJE39219-I The ClassLoader for the web application was initialized. (context root = /
dashboard, initialized time = 2013/07/22 19:17:44.084)
KDJE39003-I The web container started. (server name = uCSDPAF_Server)

＜インストールディレクトリ＞¥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.
```

4.5.2　Dashboard Viewer での分析結果の表示

Dashboard Viewer を使用して分析結果を表示するには，次の手順で操作します。

1. ダッシュボードの画面を開きます。

   Web ブラウザに特定の URL を入力し，保存されているダッシュボードの画面を表示します。URL については，「3.8.3(2)　Dashboard Viewer の URL の連絡」を参照してください。

2. データを閲覧します。

   ダッシュボードの画面に表示された分析結果を閲覧します。また，必要に応じて，分析結果の画面から別の画面を表示します。ダッシュボードの画面については，「12.3.1　Dashboard Viewer の画面構成」を参照してください。

4.5.3　Dashboard Server の停止

Dashboard Server を停止するには，次の順序で，Web サーバ，Web コンテナサーバ，および PRF デーモンを停止します。

Windows の場合

1. httpsd コマンドを実行して，Web サーバを停止します。

   ＜インストールディレクトリ＞¥psb¥httpsd¥httpsd.exe -k stop -n "Hitachi Web Server for uCSDPAF"

   また，次のコマンドでも Web サーバを停止できます。

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

2. cjstopweb コマンドを実行して，Web コンテナサーバを停止します。

   ＜インストールディレクトリ＞¥psb¥CC¥web¥bin¥cjstopweb.exe uCSDPAF_Server

   コマンドを実行すると，Web コンテナサーバを実行していたコマンドプロンプトに次のメッセージが表示され，制御が戻ります。

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

3. cprfstop コマンドを実行して，PRF デーモンを停止します。

   ＜インストールディレクトリ＞¥psb¥PRF¥bin¥cprfstop.exe

Linux の場合

- V01-50 より前

1. kill コマンドを実行して，Web サーバを停止します。

```
kill -TERM `cat /opt/hitachi/sdp/psb/httpsd/logs/httpd.pid`
```

2. cjstopweb コマンドを実行して，Web コンテナサーバを停止します。

```
/opt/hitachi/sdp/psb/CC/web/bin/cjstopweb uCSDPAF_Server
```

3. cprfstop コマンドを実行して，PRF デーモンを停止します。

```
/opt/hitachi/sdp/psb/PRF/bin/cprfstop
```

- V01-50 以降

1. kill コマンドを実行して，Web サーバを停止します。

```
kill -TERM `cat /opt/hitachi/sdp/psb/httpsd/logs/httpd.pid`
```

2. cjstopapp コマンドを実行して，開始中の J2EE アプリケーション（dashboard）を停止します。

```
/opt/hitachi/sdp/psb/CC/admin/bin/cjstopapp uCSDPAF_Server -name dashboard
```

3. cjstopweb コマンドを実行して，J2EE サーバを停止します。

```
/opt/hitachi/sdp/psb/CC/server/bin/cjstopsv uCSDPAF_Server
```

4. cprfstop コマンドを実行して，PRF デーモンを停止します。

```
/opt/hitachi/sdp/psb/PRF/bin/cprfstop
```

Windows の場合の実行例を次に示します。

```
＜インストールディレクトリ＞¥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 uCSDPAF service has stopped.

＜インストールディレクトリ＞¥psb¥CC¥web¥bin¥cjstopweb.exe uCSDPAF_Server

＜インストールディレクトリ＞¥psb¥PRF¥bin¥cprfstop.exe

 Mon Jul 22 19:21:31 2013:KFCT73413-I 2580 652:now terminating cprfd.  terminate type = NORMAL
STOP
 Mon Jul 22 19:21:32 2013:KFCT73001-I 2580 652:prf tracing service stopped. ID:PRF_ID
 Mon Jul 22 19:21:32 2013:KFCT73414-I 2580 652:CPRFD stop.
```

5 システムの変更

この章では, システムの運用中にシステムの設定や構成を変更する手順について説明します。

5.1　システムの変更の概要

運用中のシステムを変更したい場合に，定義ファイルやコマンドを使用して変更します。システムの変更には次の項目があります。

- **ストリームデータ処理エンジンの変更**
  運用中のストリームデータ処理エンジンの情報を変更します。
- **クエリグループの変更**
  運用中のクエリグループの情報を変更します。
- **アダプターの変更**
  運用中のアダプターの情報を変更します。
- **外部定義関数の変更**
  運用中の外部定義関数の情報を変更します。
- **ダッシュボードの変更**
  ストリームデータの集計・分析結果をダッシュボードに表示している場合に，Dashboard Viewer の画面の構成などを変更します。また，Dashboard Server の登録を解除します。
- **メッセージの出力言語種別の変更**
  メッセージの出力言語種別を変更します。

5.2　ストリームデータ処理エンジンの変更

運用中のストリームデータ処理エンジンの変更をする場合には，次の表に示す定義ファイルの内容を変更します。

変更する定義ファイル	参照先
SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）	8.4
システムコンフィグプロパティファイル（system_config.properties）	8.6
ログファイル出力用プロパティファイル（logger.properties）	8.10

定義ファイルを変更する手順を次に示します。

1. sdpstop コマンドで SDP サーバを停止します。
   SDP サーバの停止については，「4.4.4　SDP サーバの停止」を参照してください。
2. 定義ファイルの内容を変更します。
3. sdpstart コマンドで SDP サーバを起動します。
   SDP サーバの起動については，「4.2.1　SDP サーバの起動」を参照してください。

5.3　クエリグループの変更

運用中のクエリグループを変更する場合や定義の内容を変更する場合に，クエリグループの変更をします。

変更するクエリグループの定義ファイル名と参照先を次の表に示します。

クエリグループの定義ファイル名	参照先
クエリグループ用プロパティファイル	8.7
ストリーム用プロパティファイル	8.8
クエリ定義ファイル	マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」

クエリグループ用プロパティファイルまたはストリーム用プロパティファイルの設定値の変更の場合と，クエリ定義ファイルの変更の場合では，変更する手順が異なります。ここでは，プロパティファイルの設定値の変更手順，およびクエリ定義ファイルの設定の変更手順について説明します。

5.3.1　プロパティファイルの設定値の変更

クエリグループの登録後にプロパティファイルの設定値を変更する場合，どのパラメーターの値を変更するかによって，プロパティファイルの変更のしかたが異なります。

次に示すプロパティファイルのパラメーターの一覧で，「再開始時の変更可否」が「○」になっているパラメーターの値を変更する場合，クエリグループを削除する必要はありません。「再開始時の変更可否」が「×」になっているパラメーターの値を変更する場合，クエリグループを削除する必要があります。

- クエリグループ用プロパティファイルのパラメーターの一覧（参照先：8.7.1(5)）
- ストリーム用プロパティファイルのパラメーターの一覧（参照先：8.8.1(5)）

それぞれの場合の手順を次に示します。

「再開始時の変更可否」が「○」になっているパラメーターの値を変更する場合

1. sdpstopinpro コマンドまたは sdpstopap コマンドでアダプターグループを停止します。
   アダプターグループの停止については，「4.4.1　アダプターの停止（標準提供アダプター）」を参照してください。
2. sdpcqlstop コマンドでクエリグループを停止します。
   クエリグループの停止については，「4.4.3　クエリグループの停止」を参照してください。
3. クエリグループ用またはストリーム用プロパティファイルの内容を変更します。
   クエリグループ用またはストリーム用プロパティファイルについては，「8.7　クエリグループ用プロパティファイル」，または「8.8　ストリーム用プロパティファイル」を参照してください。
4. sdpcqlstart コマンドに-reload オプションを指定して実行し，クエリグループを再開始します。
   クエリグループの開始については，「4.2.3　クエリグループの開始」を参照してください。
5. sdpstartinpro コマンドまたは sdpstartap コマンドでアダプターグループを起動します。
   アダプターグループの起動については，「4.2.4　アダプターの起動（標準提供アダプター）」を参照してください。

「再開始時の変更可否」が「×」になっているパラメーターの値を変更する場合

1. sdpstopinpro コマンドまたは sdpstopap コマンドでアダプターグループを停止します。
   アダプターグループの停止については，「4.4.1　アダプターの停止（標準提供アダプター）」を参照してください。
2. sdpcqlstop コマンドでクエリグループを停止します。
   クエリグループの停止については，「4.4.3　クエリグループの停止」を参照してください。
3. sdpcqldel コマンドで登録済みのクエリグループを削除します。
   クエリグループの削除については，「5.3.2(1)　クエリグループの削除」を参照してください。
4. クエリグループ用またはストリーム用プロパティファイルの内容を変更します。
   クエリグループ用またはストリーム用プロパティファイルについては，「8.7　クエリグループ用プロパティファイル」，または「8.8　ストリーム用プロパティファイル」を参照してください。
5. sdpcql コマンドでクエリグループを再登録します。
   クエリグループの登録については，「4.2.2　クエリグループの登録」を参照してください。
6. sdpcqlstart コマンドでクエリグループを開始します。
   クエリグループの開始については，「4.2.3　クエリグループの開始」を参照してください。
7. sdpstartinpro コマンドまたは sdpstartap コマンドでアダプターグループを起動します。
   アダプターグループの起動については，「4.2.4　アダプターの起動（標準提供アダプター）」を参照してください。

5.3.2　クエリ定義ファイルの変更

クエリ定義ファイルで定義した CQL 文を変更する場合に実行する，クエリグループの削除，およびクエリ定義ファイルを変更する手順について説明します。

ポイント

外部定義関数を使用している場合に，クエリ定義ファイルを変更するときの手順については，「5.5　外部定義関数の変更」を参照してください。

(1)　クエリグループの削除

SDP サーバからクエリグループを削除します。クエリグループを削除するには，sdpcqldel コマンドを実行します。クエリグループを削除すると，削除されたクエリグループに対する操作やアダプターからのデータの送受信はできなくなります。

クエリグループを削除する場合の実行例を次に示します。この例では，クエリグループ用プロパティファイル名を QueryGroupSample としています。

```
＜運用ディレクトリ＞¥bin¥sdpcqldel QueryGroupSample
```

sdpcqldel コマンドについては，「7.　コマンド」の「sdpcqldel（クエリグループの削除）」を参照してください。

(2)　クエリ定義ファイルの変更

クエリ定義ファイルを変更する手順について説明します。

1. sdpstopinpro コマンドまたは sdpstopap コマンドでアダプターグループを停止します。

アダプターグループの停止については，「4.4.1　アダプターの停止（標準提供アダプター）」を参照してください。

2. sdpcqlstop コマンドでクエリグループを停止します。

クエリグループの停止については，「4.4.3　クエリグループの停止」を参照してください。

3. sdpcqldel コマンドでクエリグループを削除します。

クエリグループの削除については，「(1)　クエリグループの削除」を参照してください。

4. クエリ定義ファイルの内容を変更します。

クエリの定義については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

5. sdpcql コマンドでクエリグループを登録します。

クエリグループの登録については，「4.2.2　クエリグループの登録」を参照してください。

5.4　アダプターの変更

標準提供アダプターを変更する場合，アダプター用定義ファイルの内容を変更します。

変更するアダプター用定義ファイル名と参照先を次の表に示します。

アダプター用定義ファイル名	参照先
アダプター構成定義ファイル	9.5
アダプターコマンド定義ファイル	9.4

定義ファイルを変更する手順を次に示します。

1. sdpstopinpro コマンドまたは sdpstopap コマンドでアダプターグループを停止します。
   アダプターグループの停止については，「4.4.1　アダプターの停止（標準提供アダプター）」を参照してください。
2. 定義ファイルの内容を変更します。
   アダプター構成定義ファイルについては「9.5　アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）」，アダプターコマンド定義ファイルについては「9.4　アダプターコマンド定義ファイル（AgentManagerDefinition.xml）」を参照してください。
3. sdpstartinpro コマンドまたは sdpstartap コマンドでアダプターグループを起動します。
   アダプターグループの起動については，「4.2.4　アダプターの起動（標準提供アダプター）」を参照してください。

5.5　外部定義関数の変更

外部定義関数を変更する場合は，次のファイルを変更します。

- クラスファイル
- 外部定義関数定義ファイル（ExternalFunctionDefinition.xml）
- クエリ定義ファイル

外部定義関数を変更する手順を次に示します。

1. **sdpstopinpro コマンドまたは sdpstopap コマンドでアダプターグループを停止します。**
   アダプターグループの停止については，「4.4.1　アダプターの停止（標準提供アダプター）」を参照してください。
2. **sdpcqlstop コマンドでクエリグループを停止します。**
   クエリグループの停止については，「4.4.3　クエリグループの停止」を参照してください。
3. **sdpcqldel コマンドでクエリグループを削除します。**
   クエリグループの削除については，「sdpcqldel（クエリグループの削除）」を参照してください。
4. **外部定義関数の処理を記述したクラスファイルを修正して，コンパイルします。**
   外部定義関数の作成については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。
5. **外部定義関数定義ファイルの内容を変更します。**
   外部定義関数定義ファイルについては，「10.1　外部定義関数定義ファイル（ExternalFunctionDefinition.xml）」を参照してください。
6. **クエリ定義ファイルの内容を変更します。**
   クエリ定義ファイルについては，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。
7. **sdpcql コマンドでクエリグループを登録します。**
   クエリグループの登録については，「4.2.2　クエリグループの登録」を参照してください。
8. **sdpcqlstart コマンドでクエリグループを開始します。**
   クエリグループの開始については，「4.2.3　クエリグループの開始」を参照してください。
9. **sdpstartinpro コマンドまたは sdpstartap コマンドでアダプターグループを起動します。**
   アダプターグループの起動については，「4.2.4　アダプターの起動（標準提供アダプター）」を参照してください。

5.6　ダッシュボードの変更

ストリームデータの集計・分析結果をダッシュボードに表示している場合の，Dashboard Viewer の画面を変更する手順，および Dashboard Server の登録を解除する手順について説明します。

5.6.1　Dashboard Viewer の画面の変更

ダッシュボードの画面に表示されるグラフの表示レイアウトなどを変更したい場合は，次の手順で Dashboard Viewer の画面編集用ファイルを編集します。

1. Web ブラウザでダッシュボードの画面を閉じます。
2. Dashboard Server を停止します。
   詳細は，「4.5.3　Dashboard Server の停止」を参照してください。
3. Dashboard Viewer の画面編集用ファイルを編集します。
   詳細は，「3.8.3(1)　Dashboard Viewer の画面の編集」を参照してください。
4. Dashboard Server を再起動します。
   詳細は，「4.5.1　Dashboard Server の起動」を参照してください。
5. Web ブラウザでダッシュボードの画面を開きます。
   詳細は，「4.5.2　Dashboard Viewer での分析結果の表示」を参照してください。

5.6.2　Dashboard Server の登録解除

ダッシュボードへの出力を使用しなくなった場合や Stream Data Platform - AF をアンインストールする場合には，Dashboard Server の登録を解除します。

Dashboard Server の登録を解除する場合，次の手順で Web コンテナサーバをアンセットアップし，Web サーバ（Hitachi Web Server）のサービスの登録を解除します。

(1)　Windows の場合

1. Web コンテナサーバをアンセットアップします。
   次のコマンドを実行して，Dashboard Server と通信する Web コンテナサーバをアンセットアップします。
   ＜インストールディレクトリ＞¥psb¥CC¥web¥bin¥cjwebsetup.exe -d uCSDPAF_Server
2. Web サーバの登録を解除します。
   次のコマンドを実行して，Web サーバ（Hitachi Web Server）のサービスの登録を解除します。
   ＜インストールディレクトリ＞¥psb¥httpsd¥httpsd.exe -k uninstall -n "Hitachi Web Server for uCSDPAF"

(2)　Linux の場合

V01-50 より前

1. Web コンテナサーバをアンセットアップします。
   次のコマンドを実行して，Dashboard Server と通信する Web コンテナサーバをアンセットアップします。
   /opt/hitachi/sdp/psb/CC/web/bin/cjwebsetup -d uCSDPAF_Server

V01-50 以降

1. J2EE アプリケーションを停止します。

   次のコマンドを実行して，J2EE アプリケーションを停止します。

   /opt/hitachi/sdp/psb/CC/admin/bin/cjstopapp uCSDPAF_Server -name dashboard

2. J2EE アプリケーションのアンデプロイをします。

   次のコマンドを実行して，起動した J2EE サーバから J2EE アプリケーション（dashboard）を削除します。

   /opt/hitachi/sdp/psb/CC/admin/bin/cjdeleteapp uCSDPAF_Server -name dashboard

3. J2EE サーバを停止します。

   次のコマンドを実行して，J2EE サーバを停止します。

   /opt/hitachi/sdp/psb/CC/server/bin/cjstopsv uCSDPAF_Server

4. J2EE サーバをアンセットアップします。

   次のコマンドを実行して，J2EE サーバの環境をアンセットアップします。

   /opt/hitachi/sdp/psb/CC/server/bin/cjsetup -d uCSDPAF_Server

5.7　メッセージの出力言語種別の変更

メッセージの出力言語種別（日本語または英語）を変更する方法を次に示します。なお，デフォルトの設定では，メッセージは日本語で出力されます。

Windows の場合

コントロールパネルの地域と言語で，使用したい言語を選択してください。

なお，ダッシュボードのメッセージの出力言語を切り替える場合は，Internet Explorer のツールのインターネットオプションで変更します。全般タブで言語をクリックし，言語の優先順位を指定する画面で，使用したい言語を先頭に設定してください。

Linux の場合

メッセージの出力言語種別は，カーネルの LANG 環境変数の設定値に従います。使用したい言語に合わせて LANG 環境変数の設定値を変更してください。

日本語出力（UTF-8）の場合：LANG=ja_JP.UTF-8

英語出力の場合：LANG=C

6 トラブルシューティング

この章では，システムの運用中にトラブルが発生した場合に，採取する資料の種類や見方，およびトラブルへの対処方法について説明します。

6.1　トラブル発生時の対処の手順

システムの運用中にトラブルが発生した場合は，次の手順で対処してください。

1. メッセージが出力されているかどうかを確認します。

   トラブルが発生すると，コンソール，またはメッセージログにエラーメッセージが出力されます。

2. トラブルの事象を確認し，要因に対処します。

   出力されたエラーメッセージを確認します。マニュアル「uCosminexus Stream Data Platform - Application Framework メッセージ」を参照して，対処方法に従って障害の要因を取り除いてください。主なトラブルへの対処方法については，「6.4　主なトラブルへの対処方法」を参照してください。

   対処できない場合は，保守員に連絡してください。「保守員に連絡する」とは，購入時の契約に基づいて，弊社の問い合わせ窓口へ連絡することです。

3. トラブル発生時に採取が必要な資料を採取します。

   「6.2　トラブル発生時に採取が必要な資料」を参照して，トラブルの要因を調べるための資料を採取してください。

6.2　トラブル発生時に採取が必要な資料

ここでは，トラブル発生時に採取が必要な資料について説明します。次のような事象が発生したときに，資料を採取してください。

- **不正なタイムアウト発生時**
  CPU 利用率などでシステムへの負荷が多く掛かっていない状況で，クエリの処理で不正に時間が掛かってタイムアウトが発生した場合を指します。
- **サーバ異常停止時**
  システム内部でのエラー発生，および不当な実行時例外が発生し，サーバが停止した場合を指します。
- **サーバプロセスダウン発生時**
  SDP サーバの JavaVM プロセスが，メッセージを出力しないで突然ダウンした場合を指します。

発生した事象ごとに採取が必要な資料を次の表に示します。

表 6-1　発生した事象ごとに採取が必要な資料

項番	資料	発生した事象		
		不正なタイムアウト発生時	サーバ異常停止時	サーバプロセスダウン発生時
1	標準出力，標準エラー出力	○	○	○
2	エラーリポートファイル	－	－	○
3	SDP サーバ用定義ファイル	○	○	○
4	アダプター用定義ファイル	○	○	○
5	ログファイル	○	○	○
6	外部定義関数定義ファイル	○	○	○
7	トレースファイル	○※1	○	○
8	アダプタートレース	○※2	○	○
9	タプルログ	○※3	○	○
10	スレッドダンプ	○	○	○
11	OS の状態	○	－	－
12	OS の統計情報	○	○	○
13	Hitachi Web Server のエラーログファイル※4	○	○	○

（凡例）
○：採取します。
－：採取しません。

注※1
SDP サーバの停止後に採取できます。不正なタイムアウト発生時の，SDP サーバの停止については，「6.4.1(5)　クエリの処理でタイムアウトが発生した」を参照してください。

注※2
　アダプターの停止後に採取できます。
注※3
　クエリグループを停止することで採取できます。
注※4
　Hitachi Web Server のエラーログファイルは，Dashboard Server でエラーが発生した場合に採取します。

それぞれの資料の採取方法について，以降で説明します。

(1)　標準出力，標準エラー出力

標準出力，標準エラー出力で出力されたメッセージを控えておいてください。

なお，SDP サーバ，およびアダプターの起動・停止に関するすべてのメッセージは，標準出力または標準エラー出力に出力されます。これらのメッセージには，ログファイルに出力されるメッセージと出力されないメッセージがあります。

(2)　エラーリポートファイル

エラーリポートファイルとは，JavaVM が異常終了した場合に出力されるファイルです。プロセスダウン時のダウン位置や要因などを調査できます。

次のファイルを採取してください。

＜運用ディレクトリ＞¥hs_err_pid＜プロセスID＞.log

(3)　SDP サーバ用定義ファイル

SDP サーバ用定義ファイルとは，次のファイルを指します。

- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）
- RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）
- システムコンフィグプロパティファイル（system_config.properties）
- クエリグループ用プロパティファイル
- ストリーム用プロパティファイル
- インプロセス連携用プロパティファイル
- ログファイル出力用プロパティファイル（logger.properties）

次に示すディレクトリ下のファイルを採取してください。

＜運用ディレクトリ＞¥conf¥

なお，上記のディレクトリ下に RMI 連携用 JavaVM オプションファイルを格納していない場合は，あわせて採取してください。

(4)　アダプター用定義ファイル

アダプター用定義ファイルとは，次のファイルを指します。

- アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）
- アダプターコマンド定義ファイル（AgentManagerDefinition.xml）

次に示すディレクトリ下のファイルを採取してください。

＜運用ディレクトリ＞¥conf¥xml¥

(5)　ログファイル

ログファイルとは，次のファイルを指します。

- メッセージログ
- トレースログ
- 日立 JavaVM ログファイル

次に示すディレクトリ下のすべてのファイルを採取してください。

＜運用ディレクトリ＞¥logs¥

メッセージログ，およびトレースログについては，「6.3.1　ログファイルの詳細」を参照してください。

日立 JavaVM ログファイルは，標準の JavaVM に日立が追加した拡張オプションを使用して取得できる，日立固有の JavaVM ログが出力されたファイルです。標準の JavaVM よりも多くのトラブルシュート情報が取得できます。

(6)　外部定義関数定義ファイル

外部定義関数定義ファイルとは，次のファイルを指します。

- 外部定義関数定義ファイル（ExternalFunctionDefinition.xml）

次に示すディレクトリ下のファイルを採取してください。

＜運用ディレクトリ＞¥conf¥xml¥

(7)　トレースファイル

トレースファイルとは，API トレースなどの各種トレースファイルです。

次に示すディレクトリ下のすべてのファイルを採取してください。

＜運用ディレクトリ＞¥trc¥

API トレースについては，「6.3.2　API トレースの詳細」を参照してください。

(8)　アダプタートレース

アダプタートレースとは，アダプターの処理状況が出力されるトレースファイルです。

次のディレクトリ下のファイルを採取してください。

＜運用ディレクトリ＞¥trc¥adaptor¥

アダプタートレースについては，「6.3.3　アダプタートレースの詳細」を参照してください。

(9)　タプルログ

タプルログとは，入力タプルおよび出力タプルの情報が出力されるログファイルです。

次のディレクトリ下のファイルを採取してください。

＜運用ディレクトリ＞¥trc¥tuplelog¥

タプルログについては，「6.3.4　タプルログの詳細」を参照してください。

(10)　スレッドダンプ

スレッドダンプとは，Java のプロセス内で動作しているスレッドの情報が出力されるファイルです。

スレッドダンプは，jheapprof コマンドを実行して採取します。

スレッドダンプの採取手順，および jheapprof コマンドについては，「6.3.5　スレッドダンプの詳細」を参照してください。

(11)　OS の状態

OS の状態として，ネットワークの情報や環境変数を採取します。次のように，netstat コマンドを実行して採取してください。

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

set > set.txt
```

(12)　OS の統計情報

OS の統計情報とは，システムの負荷やパフォーマンスなどの状態を確認できる情報です。OS のパフォーマンスモニターによって採取します。

SDP サーバの実行中に，次の表に示すパフォーマンスモニターのログを一定間隔で採取します。

表 6-2　パフォーマンスモニターのログ

オブジェクト	インスタンス	カウンタ
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※	%Processor Time
		%Privileged Time
		%User Time

オブジェクト	インスタンス	カウンタ
Process	java※	Page Faults/sec
		Thread Count
		Private Bytes
		Virtual Bytes
		Working Set Bytes

（凡例）

－：該当しません。

注※

事前にパフォーマンスログの採取を開始している場合だけ採取できます。

ログの採取間隔は，60 秒間隔を推奨しますが，ディスク容量に応じて決めてください。採取間隔を長くすると，OS の統計情報の採取による性能劣化を少なくできますが，OS の統計情報の精度が悪化することがあります。

具体的な設定方法については，OS のマニュアルなどを参照してください。

(13)　Hitachi Web Server のエラーログファイル

Hitachi Web Server のエラーログファイルとは，Web コンテナの動作状況を確認および監視するためのログファイルです。エラー，警告，インフォメーションの各メッセージが出力されます。

標準提供アダプターを使用して，ストリームデータの集計・分析結果をダッシュボードに出力している場合，Dashboard Server でエラーが発生したときに，次のログファイルを採取してください。なお，ログファイル名の?には，1 または 2 の整数が入ります。

＜インストールディレクトリ＞¥psb¥CC¥web¥containers¥uCSDPAF_Server¥logs¥user_err?.log

6.3　ログファイルおよびトレースファイルの詳細

ここでは，ログファイル，トレースファイルなどのうち，次のファイルの詳細や取得方法について説明します。

- ログファイル
- API トレース
- アダプタートレース
- タプルログ
- スレッドダンプ

6.3.1　ログファイルの詳細

Stream Data Platform - AF では，次のログファイルが出力されます。

- **メッセージログ**（SDPServerMessage ＜通番※＞.log）
  サーバの動作状況を確認・監視するためのログファイルです。SDP サーバが出力するエラー，警告，インフォメーションの各メッセージが格納されます。
- **トレースログ**（SDPServerTrace ＜通番※＞.log）
  Stream Data Platform - AF で発生した問題を調査するためのログファイルです。SDP サーバが出力するスタックトレース，および保守情報が格納されます。

注※
ログファイルの通番を指します。ファイルの面数が n の場合は，n（1～16）の数値が入ります。

なお，これらのファイルに出力されるメッセージは，SDP サーバが出力するメッセージだけです。コマンド，およびアダプターが出力するメッセージは出力されません。

(1)　ログファイルの概要

ログファイルのサイズと出力先ファイルの切り替えタイミングなどを次の表に示します。

表 6-3　ログファイルの概要

項目	説明
ファイルサイズ	ファイルサイズは，ログファイル出力用プロパティファイル（logger.properties）で指定します。ファイルサイズの限界に近くなると，出力するメッセージ内容によっては指定サイズを超えることがあります。
出力先ファイルの切り替えタイミング	ファイルの切り替えタイミングは，出力中のファイルサイズが指定サイズを超えたときです。出力先ファイルの面数は，ログファイル出力用プロパティファイル（logger.properties）で変更できます。
ファイルの切り替え時のサイズ	ファイルの切り替え発生時の，切り替え先のファイルサイズは，0（全行削除）になります。
ファイルの切り替え時の動作	ファイルの切り替え発生時には全行削除し，ファイルの先頭から出力し直します。
プロセス再起動時の出力ファイル判定	プロセス再起動時にファイルのタイムスタンプが最新のものに対して，追加で出力を行います。

ログファイル出力用プロパティファイルについては，「8.10　ログファイル出力用プロパティファイル(logger.properties)」を参照してください。

(2)　ログファイルの出力形式

ログファイルの出力形式について説明します。ログファイルは，次のフォーマットで出力されます。

＜番号＞ ＜日付＞ ＜時刻＞ ＜アプリケーション名＞ ＜pid＞ ＜tid＞ ＜メッセージID＞ ＜メッセージ種別＞ ＜メッセージテキスト＞

（凡例）

番号：トレースレコードの通番（4 けた）

日付：トレースの取得日付（yyyy/mm/dd）

時刻：トレースの取得時刻（hh:mm:ss.sss）

アプリケーション名：SDPServer vvrrss（vvrrss：バージョン番号，リビジョン番号，および限定コード）

pid：プロセス ID

tid：スレッド識別子

メッセージ ID：KFSPnnnnn-x

メッセージ種別：次の値が出力されます。

- EC：例外発生
- ER：エラーメッセージ
- 設定なし：EC と ER 以外

メッセージテキスト：メッセージ ID に対応するメッセージ

メッセージについては，マニュアル「uCosminexus Stream Data Platform - Application Framework メッセージ」を参照してください。

ログファイルの出力例を次に示します。

```
0000 2010/10/15 18:46:07.723     SDPServer 0100    01642BD6 0179F36B KFSP81002-I  サーバを起動しました。
0001 2010/10/15 18:46:21.614     SDPServer 0100    01642BD6 0179F36B KFSP81003-I  サーバを停止します。
0002 2010/10/15 18:46:21.630     SDPServer 0100    01642BD6 0179F36B KFSP81004-I  サーバを停止しました。
```

(3)　標準出力に出力するメッセージ

SDP サーバ，およびアダプターの起動・停止に関するメッセージは，標準出力（または標準エラー出力）に出力されます。また，メッセージによってはログファイルにも出力されます。

なお，ログファイル出力処理の初期化前に出力される次のメッセージは，すべて標準出力または標準エラー出力に出力されます。

- KFSP81001-I（「サーバを起動します」というメッセージ）
- プロパティファイルおよび二重起動のエラーメッセージ

標準出力または標準エラー出力に出力されるメッセージには，ログファイルに出力されるメッセージと出力されないメッセージがあります。SDP サーバの起動・停止に関するメッセージ，およびアダプターの起動・停止に関するメッセージのログファイルへの出力可否を次の表に示します。なお，メッセージの内容については，マニュアル「uCosminexus Stream Data Platform - Application Framework メッセージ」を参照してください。

表 6-4　標準出力に出力されるメッセージのログファイルへの出力可否（SDP サーバの起動・停止のメッセージ）

項番	メッセージ ID	ログファイル出力
1	KFSP41001-E	×
2	KFSP41002-E	×
3	KFSP41003-E	×
4	KFSP51001-W	○
5	KFSP51002-W	○
6	KFSP51003-W	○
7	KFSP51004-E	○
8	KFSP61004-E	○
9	KFSP61005-E	○※
10	KFSP61006-E	×
11	KFSP61007-E	○
12	KFSP61008-E	×
13	KFSP61009-E	×
14	KFSP62000-E	○
15	KFSP81001-I	×
16	KFSP81002-I	○
17	KFSP81003-I	○
18	KFSP81004-I	○
19	KFSP81005-I	○
20	KFSP81006-I	○

（凡例）

○：ログファイルに出力されます。

×：ログファイルに出力されません。

注※

ログ初期化前の場合は標準出力または標準エラー出力にだけ出力されます。

表 6-5　標準出力に出力されるメッセージのログファイルへの出力可否（アダプターの起動・停止のメッセージ）

項番	メッセージ ID	ログファイル出力
1	KFSP46002-W	×
2	KFSP46101-E	×
3	KFSP46115-E	×

項番	メッセージ ID	ログファイル出力
4	KFSP56003-E	×
5	KFSP56004-I	×
6	KFSP56101-E	×
7	KFSP86001-I	○
8	KFSP86002-I	○
9	KFSP86003-I	○
10	KFSP86004-I	○
11	KFSP86005-I	○
12	KFSP86006-I	○
13	KFSP86007-I	○
14	KFSP86008-I	○
15	KFSP86009-I	○
16	KFSP86011-I	○
17	KFSP86018-E	○
18	KFSP86019-I	○
19	KFSP86020-I	○
20	KFSP86022-E	×
21	KFSP86101-I	×
22	KFSP86102-I	×

（凡例）

○：ログファイルに出力されます。

×：ログファイルに出力されません。

(4)　ログファイル処理中のエラー

ログファイルの初期化処理でエラーが発生した場合, エラーメッセージを出力します。メッセージの出力処理でエラーが発生した場合は，ログファイルの出力先を標準出力または標準エラー出力に切り替えてメッセージを出力します。

このとき，デフォルトの設定では SDP サーバが停止されます。

6.3.2　API トレースの詳細

API トレースとは, クエリの入り口情報および出口情報が出力されるトレースファイルです。アダプターおよびクエリの開始・終了を判断するために使用できます。

API トレースに出力された情報は，sdptrced コマンドで編集し，出力できます。sdptrced コマンドでの, トレースの出力例を次の図に示します。この例は，依存関係があるクエリ Q1～Q3（Q1 の出力タプルを Q2 に入力，Q2 の出力タプルを Q3 に入力）に対するトレースの出力例です。

図 6-1　API トレースの出力例

```
Date       Time     nSec     Current          Parent       Root                 EventID    Rc Data
2010/02/20 12:16:47 61542000 ＜トレース通番＞ ＜予備領域＞ ＜ルートトレース通番＞ 0x00020000 0  Q1┣━ 1.
2010/02/20 12:16:47 61700000 ＜トレース通番＞ ＜予備領域＞ ＜ルートトレース通番＞ 0x00020001 0  Q1┣━ 2.
2010/02/20 12:16:47 61774000 ＜トレース通番＞ ＜予備領域＞ ＜ルートトレース通番＞ 0x00020000 0  Q2
2010/02/20 12:16:47 61921000 ＜トレース通番＞ ＜予備領域＞ ＜ルートトレース通番＞ 0x00020001 0  Q2
2010/02/20 12:16:47 61995000 ＜トレース通番＞ ＜予備領域＞ ＜ルートトレース通番＞ 0x00020000 0  Q3
2010/02/20 12:16:47 62139000 ＜トレース通番＞ ＜予備領域＞ ＜ルートトレース通番＞ 0x00020001 0  Q3
```

注意

図で示した出力内容の上部には，トレースファイルを出力した Stream Data Platform - AF のバージョン情報，トレースファイルの種別情報（API），および API トレースファイルのパス（絶対パス）が出力されます。

また，図中の＜トレース通番＞＜予備領域＞＜ルートトレース通番＞には，実際はそれぞれ 30 けたの値が入ります。

この例から，次のことがわかります。以降の番号と図中の番号は対応しています。

1. Data の値が Q1，EventID の値が 0x00020000（クエリへのタプル入力）
   このトレースが示す時刻で，クエリ Q1 にタプルが入力されたことを意味します。
2. Data の値が Q1，EventID の値が 0x00020001（クエリからのタプル出力）
   このトレースが示す時刻で，クエリ Q1 からタプルが出力されたことを意味します。

1.および 2.以降の情報からは，EventID と Data の値によって，クエリ Q2 の入力・出力，クエリ Q3 の入力・出力の順に処理されたことがわかります。

なお，入出力が発生した時刻は，nSec の値からナノ秒単位までわかります。

sdptrced コマンドについては，「7.　コマンド」の「sdptrced（トレース情報の編集)」を参照してください。

6.3.3　アダプタートレースの詳細

アダプタートレースとは，アダプターの処理状況が出力されるトレースファイルです。Stream Data Platform - AF の運用前のテスト段階で，各コールバックの処理状況を把握するために使用します。

アダプタートレースは，アダプターのスループットに影響するため，デフォルトでは出力しない設定になっています。使用するためには，アダプタートレース定義の trace 属性に「ON」を指定します。また，実運用の前には，アダプタートレースを出力しない設定に戻してください。アダプタートレース定義については，「9.6.2　アダプタートレース定義」を参照してください。

アダプタートレースは，アダプターごとに作成され，1 ファイル当たり 2 メガバイト分のトレース情報がラップアラウンドで出力されます。

(1)　アダプタートレースの名称

アダプタートレースは，次の名称で出力されます。

adp_＜種別＞-＜アダプターグループ名＞-＜アダプター名＞_＜通番＞

ファイル名中の各項目について，次に示します。

- 種別

次のどちらかが設定されます。
IN：入力アダプターの場合に設定されます。
OUT：出力アダプターの場合に設定されます。

- アダプターグループ名
  アダプターグループ定義の name 属性に指定した値です。
- アダプター名
  アダプター定義の name 属性に指定した値です。
- 通番
  ファイルの通番です。001～004 が設定されます。

なお，アダプター起動時にアダプタートレースファイルがすでに存在していた場合，ファイル名は「ファイル名.bk」に変換されます。

アダプター用定義ファイルについては，「9.　アダプター用定義ファイル」を参照してください。

(2)　アダプタートレースの出力形式

アダプタートレースは，CSV 形式で出力されます。出力形式を次に示します。

＜No＞,＜Date＞,＜Time＞,＜Thread＞,＜Event＞

それぞれの出力項目について，次の表に示します。

表 6-6　アダプタートレースの出力項目

項番	項目	説明
1	No	4 けたの通番です。9999 に達すると，0001 に戻ります。
2	Date	トレースを採取した日付です。 YYYY/MM/DD の形式で出力されます。
3	Time	トレースを採取した時刻です。 hh:mm:ss.fff の形式で出力されます。
4	Thread	アダプター名（アダプター定義の name 属性で指定した値）が出力されます。
5	Event	コールバック名および詳細情報が次の形式で出力されます。 ＜コールバック名＞△｛entry｜exit｝ • コールバック名：CB 定義の name 属性で指定した値 • △：半角スペース • {entry ｜ exit}：次のどちらかが出力されます。 entry：コールバック処理を開始した場合 exit：コールバック処理を終了した場合

出力例を次に示します。1 行目がヘッダーで，2 行目以降がレコードです。

```
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)　アダプタートレースの出力例（レコードを破棄する場合）

各コールバックの処理でレコードを破棄する場合，コールバックの実行ごとに，実行終了直前に破棄したレコードの総数がアダプタートレースに出力されます。

この場合，「表 6-6　アダプタートレースの出力項目」の出力項目の Event が，次の形式で出力されます。

＜コールバック名＞△execute delete△＜詳細情報＞

（凡例）

コールバック名：CB 定義の name 属性で指定した値

△：半角スペース

execute delete：動作種別

＜詳細情報＞：次の表に示すコールバックごとの詳細情報

表 6-7　コールバックごとのアダプタートレースの詳細情報（レコードを破棄する場合）

コールバック種別	詳細情報	関連するアダプター構成定義ファイルの定義
HTTP パケット入力コネクター	• 出力バッファーに格納する HTTP パケット数の上限到達で破棄した場合 record△overflow△破棄数	「9.10.2　HTTP パケット入力コネクター定義」の input タグの buffersize 属性
フィルター	• フィールド条件不一致で破棄した場合 condition△レコード名△破棄数	「9.11.3　フィルター定義」の field タグ
レコード抽出	• 抽出対象条件不一致で破棄した場合 condition△レコード名※△破棄数	「9.11.4　レコード抽出定義」の targetrecord タグ
	• レコードの重複で破棄した場合 samerecord△抽出対象レコード名△破棄数	−
	• タイムアウトで破棄した場合 timeout△抽出条件名△破棄数	「9.11.4　レコード抽出定義」の extraction タグの timelimit 属性
	• レコード最大保持数の上限到達で破棄した場合 overflow△破棄数	「9.11.4　レコード抽出定義」の extractions タグの size 属性
	• レコードの時刻の逆転で破棄した場合 timereverse△破棄数	−
ダッシュボード出力コネクター	• レコード保持期間の超過で破棄した場合 timeout△破棄数	「9.10.4　ダッシュボード出力コネクター定義」の RecordHoldTime タグ
	• 最大レコード保持数の上限到達で破棄した場合 overflow△破棄数	「9.10.4　ダッシュボード出力コネクター定義」の

コールバック種別	詳細情報	関連するアダプター構成定義ファイルの定義
ダッシュボード出力コネクター	• 最大レコード保持数の上限到達で破棄した場合 overflow△破棄数	DashboardOutputConnectorDefinition タグの MaxNum 属性
	• 取得済みレコードの削除で破棄した場合 read△破棄数	「9.10.4　ダッシュボード出力コネクター定義」の DashboardOutputConnectorDefinition タグの ReadRecordRemoveFlag 属性

（凡例）

－：該当しません。

注※

すべての抽出対象条件に不一致な場合は破棄するレコード名，抽出条件に当てはまらない場合は抽出対象レコード名が出力されます。

各コールバックのアダプタートレースの出力例を示します。

- HTTP パケット入力コネクター

  HTTP パケット入力コネクター（コールバック名：inputer）の出力バッファーに格納するレコードが上限を 50 件超えたため，あふれたレコードを破棄した場合

```
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
```

- フィルター

  フィルター（コールバック名：filter1）のフィールド条件に不一致なレコード（R1）を 100 件破棄した場合

```
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
```

- レコード抽出

  レコード抽出（コールバック名：extract1）の抽出対象レコード条件（conditionE2）に不一致なレコードを 10 件破棄した場合

```
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
```

- ダッシュボード出力コネクター

  ダッシュボード出力コネクター（outputer）が持つレコードが最大レコード保持数に到達したため，20 件破棄した場合

```
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
```

(4)　アダプタートレースの出力例（HTTP パケットまたは HTTP メッセージを破棄する場合）

HTTP パケット入力コネクターの処理で HTTP パケット，または HTTP メッセージを破棄する場合，HTTP パケット入力コネクターの処理ごとに，処理終了直前に破棄した HTTP パケット，または HTTP メッセージの総数がアダプタートレースに出力されます。

この場合，「表 6-6　アダプタートレースの出力項目」の出力項目の Event が，次の形式で出力されます。

＜コールバック名＞△execute delete△＜詳細情報＞

（凡例）

コールバック名：CB 定義の name 属性で指定した値

△：半角スペース

execute delete：動作種別

＜詳細情報＞：次の表に示す詳細情報

表 6-8　アダプタートレースの詳細情報（HTTP パケットまたは HTTP メッセージを破棄する場合）

コールバック種別	詳細情報	関連するアダプター構成定義ファイルの定義
HTTP パケット入力コネクター	• 入力バッファーに格納する HTTP パケット数の上限到達で，HTTP パケットを破棄した場合 packet△overflow△破棄数	「9.10.2　HTTP パケット入力コネクター定義」の input タグの buffersize 属性
	• 分割パケットの組み立て時間がタイムアウトして，HTTP メッセージを破棄した場合 httpmessage△timeout△破棄数	「9.10.2　HTTP パケット入力コネクター定義」の input タグの assemblingtime 属性

次の場合のアダプタートレースの出力例を示します。

- 入力バッファーに格納するパケットが上限を 20 件超えたため，あふれたパケットを破棄した場合

```
3288,2010/10/15,10:16:11.223,InputAdaptor1,inputer entry
3289,2010/10/15,10:16:11.223,InputAdaptor1,inputer execute delete packet overflow 20
3291,2010/10/15,10:16:11.223,InputAdaptor1,inputer exit
```

- 分割パケットの組み立て時間がタイムアウトした 5 件の HTTP メッセージを破棄した場合

```
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　タプルログの詳細

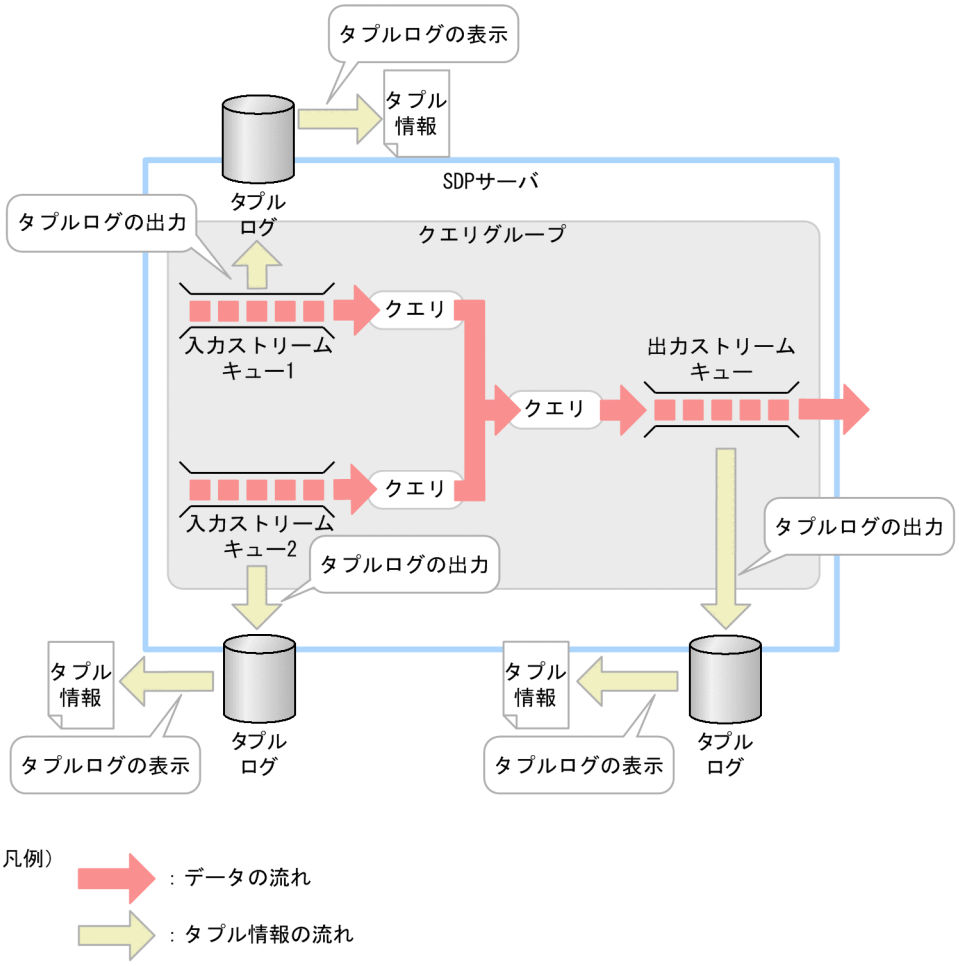
タプルログとは，入力タプルおよび出力タプルの情報が出力されるログファイルです。タプルログは，ストリームキューごとに出力されます。タプルログは次の目的で使用できます。

- SDP サーバに到着したタプルの確認
- クエリの再実行，および結果の確認

クエリの再実行については，「4.3.1　クエリの再実行」を参照してください。

タプルログの出力と表示の流れを次の図に示します。

図 6-2　タプルログの出力と表示の流れ



- タプルログの出力

  入力タプル，および出力タプルをタプルログファイルに出力します。タプルログファイルは，ストリームキューごとに出力します。

- タプルログの表示

  sdptplls コマンドでタプルログファイルを表示します。

  タプルログファイルの内容から，SDP サーバに到着したタプルの確認ができます。

  sdptplls コマンドについては，「7.　コマンド」の「sdptplls（タプル情報の表示）」を参照してください。

(1)　タプルログの出力

タプルログを出力するかどうかは，次に示すファイルの「tpl.」で始まるパラメーターで指定します。

- system_config.properties（システムコンフィグプロパティファイル）
- クエリグループ用プロパティファイル
- ストリーム用プロパティファイル

各ファイルについては，「8.　SDP サーバ用定義ファイル」を参照してください。

タプルログは，ストリームキューごとに用意されたタプルログバッファーにバッファリングされ，ファイルに出力されます。タプルログの出力契機を次の表に示します。

表 6-9　タプルログの出力契機

項番	タプルログの出力契機	ファイル出力されるタプルログ
1	tpl.bufferSize パラメーターの指定値を超えたとき	バッファリングしているタプルログ。
2	入力ストリームキューに対して，putEnd メソッドを実行したとき	入力ストリームキューでバッファリングしているすべてのタプルログ。
3	クエリグループ内のすべての入力ストリームキューに対して，putEnd メソッドを実行したとき	クエリグループ内の，出力ストリームキューでバッファリングしているすべてのタプルログ。
4	クエリグループを停止したとき	クエリグループ内の，ストリームキューでバッファリングしているすべてのタプルログ。
5	SDP サーバを終了したとき	SDP サーバ内の，ストリームキューでバッファリングしているすべてのタプルログ。

タプルログは，tpl.fileCount パラメーターで指定した面数のファイルに対してラップアラウンドで出力されます。タプルログの切り替え契機を次の表に示します。

表 6-10　タプルログの切り替え契機

項番	タプルログの切り替え契機	切り替えるタプルログ
1	tpl.fileSize パラメーターの指定値を超えたとき	満杯になったタプルログ。
2	クエリグループ内のすべての入力ストリームキューに対して，putEnd メソッドを実行したとき	クエリグループ内のストリームキューのタプルログ。

なお，次の場合は，タプルログが出力されません。

- put メソッドで例外が発生した場合

  ただし，ArrayList パラメーターに複数タプルを指定した put メソッドで例外が発生した場合，例外発生までに投入したタプルが出力されます。どのタプルまで出力したかは sdptplls コマンドで確認してください。

  sdptplls コマンドについては，「7.　コマンド」の「sdptplls（タプル情報の表示）」を参照してください。
- タプルの投入中にクエリグループを強制停止，または SDP サーバを強制停止した場合

  これらの場合，投入中のタプルを取得できない場合があります。

(2)　タプルログの名称

タプルログの出力先ディレクトリには，次のファイルが出力されます。なお，ファイルのアクセス権限は，sdpstart コマンドを実行したユーザーの権限に従います。

- タプルログ
- タプルログのバックアップファイル

それぞれのファイルは，次の名称で出力されます。

タプルログ

tpl_＜クエリグループ名＞-＜ストリーム名＞_＜ファイル通番＞

タプルログのバックアップファイル

tpl_＜クエリグループ名＞-＜ストリーム名＞_＜ファイル通番＞.bk＜バックアップ世代番号＞

ファイル名中の各項目について，次に示します。

- クエリグループ名
  取得対象のストリームが属するクエリグループ名です。
- ストリーム名
  タプルログの取得対象になっているストリームキューに対応するストリーム名です。
- ファイル通番
  ファイルの通番です。001 から tpl.fileCount パラメーターの指定値の 3 けたの数字が設定されます。
- バックアップ世代番号
  バックアップファイルの世代番号です。01 から tpl.backupFileCount パラメーターの指定値の 2 けたの数字が設定されます。

(3)　タプルログの表示

出力されたタプルログの情報は，sdptplls コマンドを実行することで確認できます。

sdptplls コマンドについては，「7.　コマンド」の「sdptplls（タプル情報の表示）」を参照してください。

6.3.5　スレッドダンプの詳細

スレッドダンプとは，Java のプロセス内で動作しているスレッドの情報が出力されるファイルです。

ここでは，スレッドダンプの採取手順および jheapprof コマンドについて説明します。

(1)　スレッドダンプの採取手順

次に示す手順に従って採取してください。

1. 次のコマンドを実行し，運用ディレクトリで起動しているサーバプロセスを特定します。
   ＜インストールディレクトリ＞¥psb¥jdk¥bin¥jps -v

   表示結果の例を次に示します。SDPManager および運用ディレクトリが表示されている行が，SDP サーバを示します。この例では，SDP サーバのプロセス ID は「4616」となります。

   ```
   3980 Program -Xms40m -Xmx256m
   5216 Jps -Dapplication.home=＜インストールディレクトリ＞¥psb¥jdk -Xms8m
   4616 SDPManager -Dsdp.home=＜運用ディレクトリ＞ -Dsdp.serverLogging
   ```

2. 手順 1.で表示された SDP サーバのプロセス ID に対して次のコマンドを実行し，スレッドダンプを採取します。
   ＜インストールディレクトリ＞¥psb¥jdk¥jre¥bin¥jheapprof -f -p 4616

   コマンドを実行すると，運用ディレクトリ下に，「javacore<プロセス ID>.<日時>.txt」というファイル名のスレッドダンプファイルが生成されます。

スレッドダンプは，3 秒おきに 10 回程度取得することを推奨しています。複数回スレッドダンプを取得することによって，スローダウンや無応答がどのメソッドの処理で起こっているかを，時系列順に調査できます。

スレッドダンプを取得する際は，前回のスレッドダンプの出力が終了してから取得してください。

(2) jheapprof（日立クラス別統計情報付き拡張スレッドダンプの出力）の詳細

jheapprof コマンドの形式や機能，使用例などについて説明します。

形式

```
jheapprof [-i|-f] [-class ＜クラス名＞] [-explicit|-noexplicit] [-fullgc|-copygc|-nogc] -p ＜プロセスID＞
```

機能

Java プロセスについて，日立クラス別統計情報を含んだ拡張スレッドダンプを出力します。日立クラス別統計情報によって，各クラスのインスタンスが持つメンバの配下にあるすべてのインスタンスのサイズが取得できます。

実行権限

なし

コマンド実行の前提条件

なし

格納先ディレクトリ

＜インストールディレクトリ＞¥psb¥jdk¥jre¥bin¥

引数

-i

指定されたプロセス ID のプロセスに対して，このコマンドを実行してもよいかどうかをユーザーに確認します。

この指定を省略しても，-f オプションが指定されないかぎり，このオプションが有効になります。

-f

指定されたプロセス ID のプロセスに対して，このコマンドを実行してもよいかどうかをユーザーに確認しません。

-class ＜クラス名＞

＜クラス名＞に指定されたクラス（インスタンス）をメンバに持つクラスの構造を一覧にしてスレッドダンプ中に出力します。指定するクラスのパッケージ名は引用符（"）で囲みます。

-explicit

インスタンス統計機能の統計対象に Explicit ヒープを含めます。このオプションと-noexplicit オプションを同時に指定している場合，最後に指定しているオプションが有効になります。なお，SDP サーバではこのオプションを指定する必要はありません。

-noexplicit

インスタンス統計機能の統計対象に Explicit ヒープを含めません。このオプションと-explicit オプションを同時に指定している場合，最後に指定しているオプションが有効になります。なお，SDP サーバではこのオプションを指定する必要はありません。

-fullgc

統計情報を出力する前に，フルガーベージコレクションを実行します。

このオプションと-copygc オプションまたは-nogc オプションを同時に指定している場合，最後に指定しているオプションが有効になります。

-copygc

統計情報を出力する前に，コピーガーベージコレクションを実行します。

このオプションと-fullgc オプションまたは-nogc オプションを同時に指定している場合，最後に指定しているオプションが有効になります。

-nogc

統計情報を出力する前に，ガーベージコレクションを実行しません。

このオプションと-fullgc オプションまたは-copygc オプションを同時に指定している場合，最後に指定しているオプションが有効になります。

-p <プロセス ID >

日立クラス別統計情報を出力する Java プログラムのプロセス ID を指定します。

注意事項

- 同じ Java プロセスに対して，2 回以上このコマンドを実行できません。同じ Java プロセスに対して 2 回以上このコマンドを実行したい場合は，先に実行した jheapprof コマンドによってクラス別統計情報が拡張スレッドダンプに出力されたあとに実行してください。
- Java プロセスは起動時に MailSlot を使った通信の初期化処理を実行します。初期化に失敗した場合，メッセージを出力して処理を中断します。
- 引数に指定したプロセス ID の Java プロセス所有者以外でもこのコマンドを実行できます。

戻り値

jheapprof コマンドの戻り値を次に示します。

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。
2	一定時間内に日立クラス別統計情報出力処理終了の応答がありませんでした。

エラーメッセージ

次のメッセージが出力された場合，日立クラス別統計情報付き拡張スレッドダンプは出力されません。

エラーメッセージ	説明
usage: jheapprof [-f|-i] [-class classname] [-explicit|-noexplicit] [-fullgc|-copygc|-nogc] [-garbage|-nogarbage] [-rootobjectinfo|-norootobjectinfo] [-rootobjectinfost size] -p process-id jheapprof	コマンドへの引数の指定が間違っています。
jheapprof: illegal option -- <オプション>	jheapprof コマンドに指定した<オプション>が不正です。
<プロセス ID>: Now processing previous request, this request canceled	jheapprof コマンドの引数に指定した<プロセス ID>に該当するプロセスが，クラス別統計情報の出力中です。
0: Not owner	jheapprof コマンドの引数に指定した<プロセス ID>に 0 が指定されています。
jheapprof: can't create work file at temporary directory , this request canceled	一時ファイル用ディレクトリに参照・書き込み権限がない場合，日立クラス別統計情報付き拡張スレッドダンプを出力で

エラーメッセージ	説明
jheapprof: can't create work file at temporary directory , this request canceled	きません。日立クラス別統計情報付き拡張スレッドダンプの出力要求はキャンセルされます。
jheapprof: can't get temporary directory, this request canceled	一時ファイル用ディレクトリが取り出せない場合，日立クラス別統計情報付き拡張スレッドダンプを出力できません。日立クラス別統計情報付き拡張スレッドダンプの出力要求はキャンセルされます。
jheapprof: please delete <削除できなかったファイル名> in <削除できなかったファイルのフルパス>	jheapprof コマンドを終了したときに，内部ファイルを削除できませんでした。削除できなかったファイルのフルパスにある，削除できなかったファイルを削除してください。
jheapprof: unexpected error occurred: <エラー原因>	jheapprof コマンド実行中に予期しないエラーが発生しました。 <エラー原因>には，例えば次のような表示がされます。 • 作業用メモリ確保に失敗した場合 malloc systemcall fail (errno=Y) • オブジェクトのクローズに失敗した場合 close systemcall fail (errno=Y)
jheapprof: can't communicate with process <プロセス ID>	jheapprof コマンドの引数に指定した<プロセス ID>に該当するプロセスに問題があり，通信処理でエラーが発生しているため通信できません。または，jheapprof コマンドの引数に指定した<プロセス ID>に該当するプロセスがありません。
<プロセス ID>: Timeout occurred. Java process not responding	jheapprof コマンドの引数に指定した<プロセス ID>に該当するプロセスから，一定時間内に日立クラス別統計出力処理終了の応答がありませんでした。

使用例

プロセス ID が 8662 である Java プログラムの，日立クラス別統計情報付き拡張スレッドダンプを取得する場合のコマンド実行例を次に示します。

```
jheapprof -p 8662
```

このコマンドを実行すると，次のように，日立クラス別統計情報付き拡張スレッドダンプを出力してもよいかどうか，確認するメッセージが出力されます。

```
Force VM to output HitachiJavaHeapProfile: ? (y/n)
```

日立クラス別統計情報付き拡張スレッドダンプを出力する場合は，Y または y を入力します。Y または y 以外の文字を入力すると，日立クラス別統計情報付き拡張スレッドダンプを出力しないでコマンドを終了します。

```
Force VM to output HitachiJavaHeapProfile: ? (y/n)y
```

日立クラス別統計情報付き拡張スレッドダンプを出力すると，実行中の Java プログラムでは次のメッセージが出力されます。

```
Writing Java core to javacore8662.030806215140.txt... OK
```

また，実行中の Java プログラムのカレントディレクトリに，次のファイル名で拡張スレッドダンプが出力され，Java プログラムの実行が継続されます。

```
javacore＜プロセスID＞.＜日時＞.txt
```

6.4　主なトラブルへの対処方法

ここでは，Stream Data Platform - AF での主なトラブルへの対処方法について説明します。

6.4.1　システムの運用時のトラブル

システムの運用時に発生するトラブルについて，トラブル別に対処方法を説明します。

(1)　ストリームデータ処理エンジンとアダプター間のデータ送受信ができない

ストリームデータ処理エンジンと，アダプター間のデータ送受信ができない場合の対処手順を次に示します。

1. メッセージログにメッセージが出力されているかどうかを確認し，必要に応じてエラーの要因を取り除きます。

   エラーメッセージが出力されている場合は，マニュアル「uCosminexus Stream Data Platform - Application Framework メッセージ」を参照して対処し，エラーの要因を取り除きます。

2. 次の内容を確認します。

   - 入力元および出力先のパスが正しく指定されているかどうか。
   - フォーマット変換の指定方法は正しいかどうか。

     特に，アダプター構成定義ファイルのフォーマット変換定義内の unmatchedFormat タグに「IGNORE」を指定している場合は，フォーマットの不正を見落とすおそれがあります。いったん unmatchedFormat タグに「WARNING」を指定し動作確認をしてください。

(2)　ストリームデータ処理エンジンの入力ストリームキューあふれが発生した

ストリームデータ処理エンジンの入力ストリームキューあふれが発生した場合，入力アダプターの処理量を制御して対処します。一度に読み込むレコード数または SDP サーバに送信する間隔を変更することで，単位時間当たりに使用するメモリ使用量を軽減できます。

レコード数は，アダプター構成定義ファイルのファイル入力コネクター定義の input タグの readUnit 属性で変更できます。

送信間隔は，入力用 CB 定義の interval 属性で変更できます。

(3)　内部矛盾エラーのメッセージが出力された

クエリグループまたは SDP サーバで内部矛盾エラーのメッセージが出力された場合，「6.2　トラブル発生時に採取が必要な資料」を参照して必要な資料を採取し，保守員に連絡してください。

(4)　クエリグループが閉塞した

クエリグループの実行中に障害が発生した場合，Stream Data Platform - AF はクエリグループを閉塞します。クエリグループが閉塞すると，ストリームキュー内に滞留していたタプルは破棄されます。

クエリグループが閉塞した場合の対応手順を，要因別に説明します。

ストリームキューあふれが要因の場合

クエリグループのストリームキューがあふれた場合，メッセージログに次のメッセージが出力されます。

```
KFSP42005-E ストリームキューの上限値を超えました。ストリーム名＝…，要素数＝…，上限値＝…
KFSP82205-I クエリグループを閉塞しました。クエリグループ名＝…
```

このメッセージが出力された場合，SDP サーバ用定義ファイルで JavaVM のヒープサイズ，キューの最大値，保留タプルの上限数などを見直します。必要があれば再度見積もりを実施し，クエリグループを再開始してください。

SDP サーバ用定義ファイルの設定値の変更については，「5.3.1　プロパティファイルの設定値の変更」を参照してください。クエリグループの再開始については，「7.　コマンド」の「sdpcqlstart（クエリグループの開始)」を参照してください。

不正データの入力が要因の場合

クエリの実行中に，不正な送信データによって 0 による除算や不正な型変換（変換後の値が変換先の型で表現できない）が発生した場合，メッセージログに次のメッセージが出力されます。

```
KFSP420**-E ＝…
KFSP82205-I クエリグループを閉塞しました。クエリグループ名＝…
```

このメッセージが出力された場合，クエリ定義ファイルに誤りがないかどうかを確認してください。クエリ定義ファイルに誤りがある場合は，クエリ定義ファイルを修正してください。クエリ定義ファイルに誤りがなく，送信データに問題がある場合，入力アダプターの内容を修正したあとに，クエリグループを再開始してください。

クエリ定義ファイルの変更については，「5.3.2　クエリ定義ファイルの変更」を参照してください。クエリグループの再開始については，「7.　コマンド」の「sdpcqlstart（クエリグループの開始)」を参照してください。

タイムスタンプ調整機能使用時にタプル保留数が上限を超えたことが要因の場合

タイムスタンプ調整機能の保留タプル数が上限値を超えた場合，メッセージログに次のメッセージが出力されます。

```
KFSP42301-E タイムスタンプ調整機能の保留タプル数が上限値を超えました。…
KFSP82205-I クエリグループを閉塞しました。クエリグループ名＝…
```

このメッセージが出力された場合，「11.8.6　タプルの保留期限」を参照し，タプルを保留できる上限値，または調整する時刻の単位や範囲を見直してください。

(5)　クエリの処理でタイムアウトが発生した

クエリの処理で不正に時間が掛かってタイムアウトが発生した場合,「6.2　トラブル発生時に採取が必要な資料」を参照して必要な資料を採取し，原因を調査してください。また，必要に応じて，SDP サーバの強制停止などの対処をしてください。

SDP サーバの強制停止については，「7.　コマンド」の「sdpstop（SDP サーバの停止)」を参照してください。

(6)　アダプターの処理速度が遅い

アダプターの処理速度が遅い場合,フルガーベージコレクションが多発していないか確認してください。フルガーベージコレクションが発生している間は，アダプターの処理がすべて停止してしまうため，処理速度が遅くなります。フルガーベージコレクションの情報は，次に示すファイルから取得します。

インプロセス連携の場合

＜運用ディレクトリ＞¥logs¥SDPServerVM＜整数＞.log

RMI 連携の場合

＜運用ディレクトリ＞¥logs¥SDPClientVM＜整数＞.log

なお，ファイル名の＜整数＞には 01 から 04 の数字が入り，ファイルは最大 4 面できます。また，これらのファイルの出力先，およびファイル名は，デフォルトの状態です。

フルガーベージコレクションの出力例を示します。

```
[VGC]<Fri Jan 08 17:19:46.159 2010>[Full GC 54330K->45174K(519296K),
0.1150830 secs][DefNew::Eden: 9156K->0K(164608K)]・・・
```

下線で示している「Full GC」が頻繁に出力されている場合や，フルガーベージコレクションに掛かった時間が数秒単位の場合には，JavaVM が確保するヒープ領域のメモリ不足が考えられます。この場合は，次の対処を検討してください。

- JavaVM のヒープ領域の最大サイズを変更する。
  次のファイルで変更できます。詳細については，「8.　SDP サーバ用定義ファイル」を参照してください。
  - SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）
  - RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）
- 入力アダプターの処理量を制御する。
  入力アダプターの処理量の制御方法については，「(2)　ストリームデータ処理エンジンの入力ストリームキューあふれが発生した」を参照してください。

(7)　データ型の不一致でアダプター処理が停止した

アダプター構成定義ファイルのフォーマット変換定義で field タグの type 属性に指定したデータ型と，対応する CQL のデータ型が一致しない場合，タプル送信でエラーとなり，アダプターの処理が停止します。この場合は，データ型種別を見直してください。

フォーマット変換定義の field タグの type 属性については，「9.11.1　フォーマット変換定義」を参照してください。

(8)　タプルログの一部が欠落した

すべてのタプルログのバッファーがいっぱいになりタプルログのバッファリングができなくなると，タプルログの一部が欠落します。

タプルログが欠落すると，sdptplput コマンドを使用してクエリの再実行をした結果が，アダプターを使用してクエリを実行した結果と異なる場合があります。このため，十分な数のタプルを取得できるように，バッファーサイズおよび面数を指定する必要があります。

タプルログのバッファーの見積もり式を次に示します。この式を基に，格納するタプルの数に対して必要なタプルログバッファーのサイズおよび面数を見積もってください。

タプルログのバッファー一つ当たりのサイズ（単位：バイト）＝
タプル一つ当たりのサイズ×格納数

タプル一つ当たりのサイズについては，「2.7.1(2)　タプル一つ当たりに必要なメモリ使用量」を参照してください。

(9)　ダッシュボードで分析結果が表示できない

ダッシュボードで分析結果が表示できない場合，次のことを確認して対処してください。

- Hitachi Web Server が起動しているかどうかを確認してください。
  Hitachi Web Server が起動していない場合は，起動してください。
- インストールされている Flash Player のバージョンを確認してください。

Flash Player のバージョンが 9.0.124.0 より古い場合は，最新バージョンをインストールしてください。

- 次のファイルの内容が正しいかどうかを確認してください。
  - アダプター構成定義ファイルのダッシュボード出力コネクター定義
  - Dashboard Server の画面編集用ファイル

  ファイルの内容に誤りがある場合，ファイルを修正してください。
- ダッシュボード出力コネクターが起動しているかを確認してください。
  ダッシュボード出力コネクターが起動していない場合，受信したいデータを持つダッシュボード出力コネクターを起動してください。

なお，ダッシュボードでエラーが発生すると，ダッシュボード画面にエラーダイアログが出力されます。ダイアログには次の情報が出力されます。

表示名	内容
“ErrID”	エラーメッセージ ID が表示されます。
“ErrMsg”	エラーメッセージの内容が表示されます。

メッセージについては，マニュアル「uCosminexus Stream Data Platform - Application Framework メッセージ」を参照してください。

7 コマンド

この章では, ストリームデータ処理システムを運用するためのコマンドについて説明します。

コマンドの記述形式

ここでは，コマンドの記述形式について説明します。

各コマンドで説明する項目は次のとおりです。ただし，コマンドによっては説明しない項目もあります。

形式

コマンドの形式を説明しています。

機能

コマンドの機能について説明しています。

実行権限

コマンドの実行に必要なユーザーの権限について説明しています。

コマンド実行の前提条件

コマンドの実行に必要な前提条件について説明しています。

格納先ディレクトリ

コマンドの格納先ディレクトリについて説明しています。

引数

コマンドの引数について説明しています。

注意事項

注意事項を説明しています。

戻り値

コマンドの戻り値について説明しています。

出力形式

コマンドの出力形式について説明しています。

コマンド一覧

システムを構築・運用するためのコマンドの一覧を次に示します。

表 7-1　システムを構築・運用するためのコマンド一覧

項番	コマンド名	機能
1	sdpcql	クエリグループを SDP サーバに登録します。
2	sdpcqldel	SDP サーバに登録されている停止中または閉塞中のクエリグループを削除します。
3	sdpcqlstart	SDP サーバに登録されている停止状態または閉塞状態のクエリグループを開始します。
4	sdpcqlstop	SDP サーバで実行中のクエリグループを停止します。
5	sdpls	SDP サーバに登録されているクエリグループの情報を表示します。
6	sdpsetup	運用環境をセットアップします。
7	sdpstart	SDP サーバを起動します。
8	sdpstartap	指定した RMI 連携の標準提供アダプター，または RMI 連携のカスタムアダプターを起動します。
9	sdpstartinpro	指定したインプロセス連携の標準提供アダプター，またはインプロセス連携のカスタムアダプターを起動します。
10	sdpstop	SDP サーバを停止します。
11	sdpstopap	指定した RMI 連携の標準提供アダプターを停止します。
12	sdpstopinpro	指定したインプロセス連携の標準提供アダプター，または SDP サーバに登録されているインプロセス連携のカスタムアダプターを停止します。
13	sdptplls	タプルログファイルに取得したタプルの情報を表示します。
14	sdptplput	タプルログファイルに取得したタプルを入力ストリームキューに再投入します。
15	sdptrced	API トレース情報を編集して出力します。

sdpcql（クエリグループの登録）

形式

```
sdpcql ｛[-autostart] ＜クエリグループ用プロパティファイル名＞|-help｝
```

機能

クエリグループを SDP サーバに登録します。

クエリグループ用プロパティファイルに指定したクエリ定義ファイルの内容に構文エラーがある場合，構文エラーメッセージをログファイルとコンソールに出力します。ファイル内のすべてのクエリに対する構文解析は続行します。

なお，登録したクエリグループは，SDP サーバ停止後は保持されません。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）
- クエリグループ用プロパティファイル
- ストリーム用プロパティファイル
- クエリ定義ファイル

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

-autostart

クエリグループを登録したあと，クエリグループの実行および入力タプルの受け付けを開始します。

このオプションを指定しない場合，登録後のクエリグループは停止状態となります。

＜クエリグループ用プロパティファイル名＞

登録するクエリグループのクエリグループ用プロパティファイル名を指定します。

クエリグループ用プロパティファイルについては，「8.7　クエリグループ用プロパティファイル」を参照してください。

-help

コマンドの使用方法を表示します。

このオプションを指定した場合，クエリグループの登録およびオプションの不正チェックはされないでコマンドが終了します。

注意事項

- クエリ定義ファイルに記述されている CQL 文すべてに文法上のエラーがなく，正常に登録できたとき，戻り値 0 が返されます。
- クエリグループ用プロパティファイル名をクエリグループ名として登録します。そのため，登録済みのクエリグループと同じ名称のクエリグループ用プロパティファイルは登録できません。
- クエリグループ用プロパティファイル，ストリーム用プロパティファイルおよびクエリ定義ファイルの解析中にエラーが発生した場合，定義したストリームおよびクエリは登録されません。
- 同一名称のストリームまたはクエリがシステムに登録されている場合，登録できません。
- -autostart オプションを指定してこのコマンドを実行した結果，「クエリグループを開始できません。」というメッセージが出力された場合でも，クエリグループは登録されています。この場合，エラーの原因を取り除いたあと，sdpcqlstart コマンドを実行してクエリグループを開始してください。
- コマンド実行中に Ctrl+C で中断した場合，次回 sdpcql コマンド実行時にメッセージ KFSP52205-E（内部矛盾）が出力されることがあります。

戻り値

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。

sdpcqldel（クエリグループの削除）

形式

```
sdpcqldel {<クエリグループ名> | -help}
```

機能

SDP サーバに登録されている停止中または閉塞中のクエリグループを削除します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

<運用ディレクトリ>¥bin¥

引数

<クエリグループ名>

削除するクエリグループ名を指定します。

-help

コマンドの使用方法を表示します。

このオプションを指定した場合，クエリグループの削除およびオプションの不正チェックはされないでコマンドが終了します。

注意事項

なし

戻り値

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。

sdpcqlstart（クエリグループの開始）

形式

```
sdpcqlstart　{ [-clear]　[-reload]　＜クエリグループ名＞ | -help}
```

機能

SDP サーバに登録されている停止状態または閉塞状態のクエリグループを開始します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

-clear

クエリグループを開始する前に，出力ストリームキュー内に格納されたタプルを破棄します。

このオプションを指定しない場合は，前回実行時に出力ストリームキューに格納されたタプルを破棄しないで，クエリグループを開始します。

-reload

クエリグループ用プロパティファイルおよびストリーム用プロパティファイルの内容を再度読み込み，設定を反映してからクエリグループを開始します。

このオプションは，登録時に設定したクエリグループのプロパティ定義値を変更する場合に指定します。

クエリグループ用プロパティファイルと開始時に変更できるプロパティ定義値については，「8.7　クエリグループ用プロパティファイル」を参照してください。ストリーム用プロパティファイルと開始時に変更できるプロパティ定義値については，「8.8　ストリーム用プロパティファイル」を参照してください。

＜クエリグループ名＞

開始するクエリグループ名を指定します。

-help

コマンドの使用方法を表示します。

このオプションを指定した場合，クエリグループの開始およびオプションの不正チェックはされないでコマンドが終了します。

注意事項

- -reload オプションを指定すると，クエリグループ用プロパティファイルの定義解析でエラーになった場合，クエリグループは開始されません。
- -reload オプションを指定してクエリグループを開始し，開始処理でエラーになった場合，クエリグループ用プロパティファイルおよびストリーム用プロパティファイルの設定は反映されません。
- 入力ストリームに対して入力アダプターから接続している場合，-clear オプションを指定してこのコマンドを実行すると，入力ストリーム内のデータ受信 AP 接続キューに格納されたタプルも破棄対象になります。

戻り値

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。

sdpcqlstop（クエリグループの停止）

形式

```
sdpcqlstop ｛ [-force] ＜クエリグループ名＞ | -help}
```

機能

SDP サーバで実行中のクエリグループを停止します。-force オプション指定時以外は，コマンドを実行した時点で入力ストリームキューに格納されていたタプルを処理してから停止します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

-force

入力ストリームキューに格納されていたタプルを処理しないで，クエリグループを停止します。入力ストリームキューに格納されていたタプルは破棄されます。

＜クエリグループ名＞

停止するクエリグループ名を指定します。

-help

コマンドの使用方法を表示します。

このオプションを指定した場合，クエリグループの停止およびオプションの不正チェックはされないでコマンドが終了します。

注意事項

- このコマンドで正常停止処理中にタプルの送信処理（putEnd メソッドも含む）がされた場合，実行中の停止処理が完了するまで送信処理がされないことがあります。
- このコマンドを複数回実行した場合，実行中の停止処理が完了するまでコマンドがリターンしないことがあります。
- タプルの送信処理中に，-force オプションを指定しないでこのコマンドを実行した場合，タプルの送信処理完了後に正常停止処理が実行されます。また，タプルの送信処理途中に，-force オプションを指定してこのコマンドを実行した場合，送信処理中のタプルが入力ストリームキューに残ることがあります。入力ストリームキューに残ったタプルは次のクエリグループの開始時に破棄されます。

戻り値

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。

sdpls（クエリグループの状態表示）

形式

```
sdpls { {-all | ＜クエリグループ名＞ [ ＜クエリグループ名＞...] } | -help}
```

機能

SDP サーバに登録されているクエリグループの情報を表示します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

-all

SDP サーバに登録されているすべてのクエリグループに関する情報を表示します。

＜クエリグループ名＞ [＜クエリグループ名＞...]

表示するクエリグループ名を指定します。複数のクエリグループ名を指定する場合は，クエリグループ名とクエリグループ名の間を半角スペースで区切って指定します。

-help

コマンドの使用方法を表示します。

このオプションが指定された場合，ストリームおよびクエリグループの情報表示，ならびにオプションの不正チェックはされないでコマンドが終了します。

注意事項

- 引数に指定したすべてのクエリグループの表示が成功した場合，戻り値 0 が返されます。
- 複数のクエリグループを指定してこのコマンドを実行し，特定のクエリグループについての情報の取得処理でエラーが発生した場合，ほかのクエリグループの表示は続行されます。この場合，戻り値 1 が返されます。
- クエリグループの登録時は，開始時刻に「----/--/-- --:--:--」が表示されます。また，クエリグループの実行時は，停止時刻に「----/--/-- --:--:--」が表示されます。
- ストリームキューに格納されたタプルの総数が上限値（9,223,372,036,854,775,807）を超えた場合は，値が更新されなくなります。

- 引数に同一名称のクエリグループを複数指定した場合，該当するクエリグループの情報は 1 度だけ出力します。
- ストリームキューに格納されたタプルの総数およびストリームキューの最大滞留数はクエリグループの開始から停止までの値とし，クエリグループを一度停止し再度開始したときにリセットします。また，クエリグループのすべての入力ストリームに対して putEnd メソッドを発行した場合も，値をリセットします。リセットする内容は次のとおりです。
  - ストリームキューに格納されたタプルの総数：0
  - ストリームキューの最大滞留数：開始時のキュー滞留数
  - タイムスタンプ調整機能で保留していた最大タプル数：0
  - タイムスタンプ調整機能で破棄したタプル数：0
- ストリームキューに格納されたタプルの総数として，内部で生成する制御用タプルも総数に含みます。そのため，次の場合は，アダプターから送信されたタプルの総数より多く表示されることがあります。
  - カスタムアダプターから putEnd メソッドを発行した場合。
  - クエリグループを正常停止した場合。
- 入力ストリームに対して入力アダプターから接続した場合，ストリームキューの最大滞留数および現在の滞留数には，入力アダプターからの接続ストリーム内の滞留数が表示されることがあります。そのため，入力アダプターからのタプルの取り出しが遅延した場合などは，SDP サーバ内の滞留数より多く表示されることがあります。

戻り値

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。

出力形式

```
*** Total Information ***                                    ┐
Total Query Group Count  : aaaaaaaaaa                        ├ 1
Total Stream Count       : bbbbbbbbbb                        ┘
Total Query Count        : cccccccccc
---------+---------+---------+---------+---------+---------+                    ┐
*** 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 *****                               ┐                  ├ 2
Stream Name              : ll...ll                           │                  │
Data Count               : mm...mm                           │                  │
Queue Count              : nnnnnnnnnn                        ├ 3                │
Max   Count              : ooooooooo                         │                  │
Accuracy Adjust          : pp...pp                           ┐                  │
Tuple Count              : qqqqqqqqqq                        ├ 4                │
Max Tuple Count          : rrrrrrrrrr                        │                  │
Reject Tuple Count       : sssssssss                         ┘                  │
                                                             │                  │
***** Query Information *****                                ┐                  │
Query Name               : tt...tt                           │                  │
Data Count               : uu...uu                           ├ 5                │
Queue Count              : vvvvvvvvvv                        │                  │
Max   Count              : wwwwwwwww                         ┘                  ┘
```

(凡例)

1：-all オプション指定時にだけ表示されます。

2：表示されるクエリグループの数だけ繰り返し表示されます。

3：クエリグループに登録されたストリームの数だけ繰り返し表示されます。

4：クエリグループのタイムスタンプモードが，DATASOURCE の場合にだけ表示されます。

5：クエリグループに登録されたクエリ数分繰り返し表示されます。

出力形式中の各変数の意味を次に示します。

情報の分類	変数	意味
-all オプション指定時に出力される情報	aaaaaaaaaa	クエリグループの総数（10 けたの 10 進数）
	bbbbbbbbbb	ストリームの総数（10 けたの 10 進数）
	cccccccccc	クエリの総数（10 けたの 10 進数）
クエリグループごとの情報	dd...dd	クエリグループ名称
	ee...ee	タイムスタンプモード 次のどちらかが表示されます。 • DataSource：データソースモード • Server：サーバモード
	ff...ff	サーバモードでの sdptplput コマンドの実行可否 次のどちらかが表示されます。

情報の分類	変数	意味
クエリグループごとの情報	ff...ff	• true：実行できる • false：実行できない
	gggggggggg	グループに属するストリーム数（10 けたの 10 進数）
	hhhhhhhhhh	グループに属するクエリ数（10 けたの 10 進数）
	ii...ii	クエリグループのステータス 次のどれかが表示されます。 • Stop：クエリグループの停止中 • Running：クエリグループの実行中 • Hold：実行時エラーやキューあふれによる，クエリグループの閉塞中 • FatalHold：クエリグループの続行不可 • Stopping：クエリグループの停止処理中 • PreparingForStop：クエリグループの停止要求による，入力キューおよび仕掛かり中のタプルの処理中 • Starting：クエリグループの開始処理中 • Deleting：クエリグループの削除処理中 • Holding：クエリグループの閉塞処理中
	jjjj/jj/jj△jj:jj:jj	クエリグループの開始時刻
	kkkk/kk/kk△kk:kk:kk	クエリグループの停止時刻
ストリームごとの情報	ll...ll	ストリーム名称
	mm...mm	ストリームキューに格納されたタプルの総数（19 けたの 10 進数）
	nnnnnnnnnn	現在のストリームキューの滞留数（10 けたの 10 進数）
	oooooooooo	ストリームキューの最大滞留数（10 けたの 10 進数）
	pp...pp	タイムスタンプ調整機能のステータス 次のどちらかが表示されます。 • Stop：タイムスタンプ調整機能が停止中 • Running：タイムスタンプ調整機能が実行中
	qqqqqqqqqq	タイムスタンプ調整機能で保留しているタプル数（10 けたの 10 進数）
	rrrrrrrrr	タイムスタンプ調整機能で保留していた最大タプル数（10 けたの 10 進数）
	sssssssss	タイムスタンプ調整機能で破棄したタプル数（10 けたの 10 進数）
クエリごとの情報	tt...tt	クエリ名称
	uu...uu	出力ストリームキューに格納されたタプルの総数（19 けたの 10 進数）
	vvvvvvvvv	現在の出力ストリームキューの滞留数（10 けたの 10 進数）
	wwwwwwwwww	出力ストリームキューの最大滞留数（10 けたの 10 進数）

（凡例）

△：半角スペース

sdpsetup（運用環境のセットアップ）

形式

sdpsetup ＜運用ディレクトリ＞

機能

運用環境をセットアップします。

実行権限

運用ユーザー

コマンド実行の前提条件

なし

格納先ディレクトリ

＜インストールディレクトリ＞¥bin¥

引数

＜運用ディレクトリ＞

システムを運用するための運用ディレクトリを絶対パスで指定します。指定するディレクトリは，運用ユーザーが参照できるパス上にしてください。

- 指定したディレクトリがない場合
  新規に作成します。
- 指定したディレクトリがすでにある場合
  指定したディレクトリ下にある bin ディレクトリは上書きされます。
  bin 以外のディレクトリがある場合，上書きされません。bin 以外のディレクトリがない場合だけ新規に作成されます。

作成される運用ディレクトリの構成については，「3.2.2　運用ディレクトリの構成」を参照してください。

運用ディレクトリ，およびその配下にあるディレクトリ，ならびにファイルのアクセス権限は，このコマンドを実行したときに設定されます。セットアップ時に存在するディレクトリとファイルについては，アクセス権限を変更しません。

引数を省略した場合，または引数にエラーがあった場合，Usage メッセージを表示してコマンドを終了します。

正しい引数が指定された場合，次のメッセージが表示されます。

```
KFSP91030-I === Setup information:
KFSP91032-I Setup directory: ＜運用ディレクトリ＞
KFSP91033-Q Do you want to continue? (y/n)
```

「y」を選択すると，セットアップを開始します。

「n」を選択すると，セットアップを中断します。

注意事項

- 引数を二つ以上指定した場合，二つ目以降の引数を無視してセットアップを続行します。
- インストールディレクトリ下は，運用ディレクトリに指定できません。指定した場合，エラーとなります。
- 運用ディレクトリのパス名には，次の文字以外は指定できません。
  半角英数字，円記号（¥），ピリオド（.），アンダーライン（_），コロン（:）

戻り値

値	意味
0	セットアップが正常に完了しました。
1	セットアップ中にエラーが発生しました。
2	セットアップがユーザーによって中断されました。

sdpstart（SDP サーバの起動）

形式

```
sdpstart
```

機能

SDP サーバを起動します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要となります。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

なし

注意事項

- このコマンドを実行すると，カレントディレクトリを運用ディレクトリに移動してサーバを起動します。そのため，SDP サーバ用 JavaVM オプションファイルに定義するクラスパスを相対パスで設定する場合は，運用ディレクトリからの相対パスを設定する必要があります。
- 同一ポート番号が使用されている場合，エラーとなります。その場合，system_config.properties でポート番号を変更する必要があります。
- 一つの運用ディレクトリで複数の SDP サーバを起動できません。
- SDP サーバが起動してから停止するまで，このコマンドには制御が戻りません。
- JavaVM が異常終了した場合，戻り値は JavaVM の戻り値となります。
- インプロセス連携のカスタムアダプターでユーザーが実装する処理で exit メソッドを実行した場合，このコマンドの戻り値は exit メソッドが指定された値となります。

戻り値

値	意味
0	正常にサーバが停止しました。
1	システムがエラーを検知したか，ユーザーが強制停止を実行したため，サーバが停止しました。

sdpstartap（RMI 連携アダプターの起動）

形式

RMI 連携の標準提供アダプターを起動する場合

```
sdpstartap jp.co.Hitachi.soft.sdp.adaptor.AdaptorManager ＜アダプターグループ名＞
```

RMI 連携のカスタムアダプターを起動する場合

```
sdpstartap  [-clientcfg ＜jvm_client_options.cfgのパス＞]
          ＜Javaアプリケーションクラス名＞
           [ ＜mainメソッドに渡される引数＞…]
```

機能

指定した RMI 連携の標準提供アダプター，または RMI 連携のカスタムアダプターを起動します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

＜アダプターグループ名＞

起動する RMI グループ名を指定します。アダプター構成定義ファイルの RMI グループ定義（<RMIGroupDefinition>タグ）の name 属性で指定したアダプターグループ名を指定してください。

-clientcfg ＜ jvm_client_options.cfg のパス＞

送受信を行うための Java アプリケーションに個別の RMI 連携用 JavaVM オプションファイルを使用する場合，そのファイルのパス名を絶対パスまたは相対パスで指定します。相対パスの場合，コマンドを投入したディレクトリからの相対パスを指定します。

このオプションの指定を省略した場合，＜運用ディレクトリ＞¥conf¥jvm_client_options.cfg が使用されます。

＜ Java アプリケーションクラス名＞

起動する Java アプリケーションクラス名を指定します。ハイフン（-）で始まる Java アプリケーションクラス名は指定できません。

＜ main メソッドに渡される引数＞

Java アプリケーションクラスの main メソッドに渡される引数を指定します。

注意事項

- 引数を省略した場合，または引数にエラーがあった場合，コマンドの使用方法を示すメッセージが表示されコマンドが終了します。RMI 連携の標準提供アダプターを起動する場合は，表示メッセージの「Class」を「jp.co.Hitachi.soft.sdp.adaptor.AdaptorManager」に，「[Args...]」を「アダプターグループ名」にそれぞれ読み替えてください。
- カスタムアダプターの場合，このコマンドは，Java アプリケーションのメインクラスの main メソッドを実行します。main メソッドは，public static void main(String[])と宣言されている必要があります。

  RMI 連携のカスタムアダプターの作成については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。
- カスタムアダプターの終了コードとして 1 を使用してはいけません。

  RMI 連携のカスタムアダプターの作成については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。
- カスタムアダプターの場合，メインクラスを jar ファイルに格納しているときは，jvm_client_options.cfg に次のように jar ファイルのクラスパスを設定する必要があります。

  ```
  SDP_CLASS_PATH=.¥¥home¥¥abc¥¥sdpclient.jar
  ```

  また，デフォルトのクラスパスとして，カレントディレクトリを使用しません。使用するクラスパスにカレントディレクトリを追加するには，jvm_client_options.cfg に，次のように設定してください。

  ```
  SDP_CLASS_PATH=.
  ```

  設定方法については，「8.5　RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）」を参照してください。
- 複数のカスタムアダプターを起動する場合，RMI 連携用 JavaVM オプションファイルを複数用意し，Java のログファイルのプレフィックス名が重複しないように設定する必要があります。

  設定方法については，「8.5　RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）」を参照してください。

戻り値

値	意味
1	引数の指定に誤りがあります。
1 以外	Java アプリケーションの終了コードが返されます。

sdpstartinpro（インプロセス連携アダプターの起動）

形式

インプロセス連携の標準提供アダプターを起動する場合

```
sdpstartinpro {<アダプターグループ名> | -help}
```

インプロセス連携のカスタムアダプターを起動する場合

```
sdpstartinpro {<インプロセス連携のカスタムアダプター名>
              [ <インプロセス連携のカスタムアダプターに渡す引数>…] | -help}
```

機能

指定したインプロセス連携の標準提供アダプター，またはインプロセス連携のカスタムアダプターを起動します。

インプロセス連携のカスタムアダプターを指定してこのコマンドを実行すると，インプロセス連携用プロパティファイルに指定されたメインクラスをロードして SDP サーバに登録したあと，StreamInprocessUP インタフェース実装クラスの execute メソッドを実行します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

＜アダプターグループ名＞

起動するインプロセスグループ名を指定します。アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）のインプロセスグループ定義（<InprocessGroupDefinition>タグ）の name 属性で指定したアダプターグループ名を指定してください。

＜インプロセス連携のカスタムアダプター名＞

起動するインプロセス連携のカスタムアダプター名を指定します。user_app.＜インプロセス連携のカスタムアダプター名＞.properties で設定した名称を指定してください。

＜インプロセス連携のカスタムアダプターに渡す引数＞

起動するインプロセス連携のカスタムアダプターに渡す引数を指定します。指定した引数は，インプロセス連携のカスタムアダプターの String 型の可変長引数を持つコンストラクターに渡されます。

-help

コマンドの使用方法を表示します。

このオプションが指定された場合，インプロセス連携の標準提供アダプターの起動，インプロセス連携のカスタムアダプターの起動，およびオプションの不正チェックはされないでコマンドが終了します。

注意事項

- 引数を省略した場合，または引数にエラーがあった場合，コマンドの使用方法を示すメッセージが表示されコマンドが終了します。インプロセス連携の標準提供アダプターを起動する場合は，表示メッセージの「InprocessApName」を「アダプターグループ名」に読み替えてください。
- SDP サーバが稼働中のときだけ実行できます。
- インプロセス連携のカスタムアダプターが実行中の場合はエラーになります。
- user_app.＜アダプター名＞.properties がない場合，または user_app.＜アダプター名＞.properties の解析でエラーがあった場合，インプロセス連携のカスタムアダプターは起動されません。
- インプロセス連携のカスタムアダプターを指定してこのコマンドを実行すると，StreamInprocessUP インタフェースの execute メソッドを起動したあと，メソッドの完了を待たないでリターンします。そのため，execute メソッド内の処理でエラーが発生した場合でも戻り値 0 が返されます。
- このコマンドで起動したインプロセス連携のカスタムアダプターは，アプリケーションで実装した execute メソッドの処理が終了しても sdpstopinpro コマンドで停止するまでは再起動できません。
- インプロセス連携のカスタムアダプターに任意の引数を渡す場合は，StreamInprocessUP インタフェースを実装するカスタムアダプターに，String 型の可変長引数を持つコンストラクターを実装してください。なお，String 型の可変長引数を持つコンストラクターを実装していない場合は，デフォルトコンストラクターが呼ばれます（ただし，引数は渡されません）。
- インプロセス連携のカスタムアダプターに任意の引数を渡さない場合は，String 型の可変長引数を持つコンストラクターまたはデフォルトコンストラクターのどちらかを実装してください。どちらも実装している場合は，String 型の可変長引数を持つコンストラクターが呼ばれます。
- String 型の可変長引数を持つコンストラクターまたはデフォルトコンストラクターのどちらも実装していない場合は，エラーとなり，インプロセス連携のカスタムアダプターは起動しません。
- インプロセス連携のカスタムアダプターに渡す引数に「-help」は指定できません。指定した場合は Usage が表示されます。

戻り値

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。

sdpstop（SDP サーバの停止）

形式

```
sdpstop { [-force] | -help}
```

機能

SDP サーバを停止します。引数を省略してこのコマンドを実行した場合は，登録されているすべてのクエリグループおよびインプロセス連携のアダプターを正常停止してから SDP サーバを停止します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

-force

SDP サーバを強制停止します。引数の指定なしでコマンドを実行しても SDP サーバを停止できない場合に，このオプションを指定します。

-help

コマンドの使用方法を表示します。

このオプションが指定された場合，SDP サーバの停止およびオプションの不正チェックはされないでコマンドが終了します。

注意事項

- -force 指定時，各クエリの実行スレッドの停止処理のタイムアウト時間は 30 秒です。
  -force 指定時にクエリ実行スレッドの停止処理でタイムアウトした場合，タイムアウトしたクエリ実行スレッドのトレース情報の一部が採取されないことがあります。
- 引数を省略してこのコマンドを実行した場合に，操作中のクエリグループまたはインプロセス連携アダプターがあるときは，正常停止できません。この場合，戻り値 1 が返されます。
- 特定のクエリグループの停止中にエラーが発生した場合，ほかのクエリグループの停止およびシステムの停止は続行します。システムが停止した場合，コマンドの戻り値は 0 になります。

戻り値

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。

sdpstopap（RMI 連携アダプターの停止）

形式

```
sdpstopap　{ [-force]　＜アダプターグループ名＞　| -help}
```

機能

指定した RMI 連携の標準提供アダプターを停止します。

実行権限

運用ユーザー

コマンド実行の前提条件

なし

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

-force

RMI グループに属するアダプターを強制停止します。処理中のデータをすべて破棄し，コールバックの停止処理を実行しないで直ちにアダプターグループを停止します。

＜アダプターグループ名＞

停止する RMI グループの名称を指定します。アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）の RMI グループ定義（<RMIGroupDefinition>タグ）の name 属性で指定したアダプターグループ名を指定してください。

-help

コマンドの使用方法を表示します。

このオプションを指定した場合，RMI 連携アダプターの停止およびオプションの不正チェックはされないでコマンドが終了します。

注意事項

- RMI 連携の標準提供アダプターが実行中の時だけ実行できます。RMI 連携の標準提供アダプターが実行中でない場合はエラーになります。
- -force を指定して実行すると，タイミングによってエラーメッセージが出力される場合がありますが，動作に支障はありません。

戻り値

値	意味
0	コマンドが正常に終了しました。

値	意味
1	コマンドでエラーが発生しました。

sdpstopinpro（インプロセス連携アダプターの停止）

形式

インプロセス連携の標準提供アダプターを停止する場合

```
sdpstopinpro　{＜アダプターグループ名＞ | -help}
```

インプロセス連携のカスタムアダプターを停止する場合

```
sdpstopinpro　{＜インプロセス連携のカスタムアダプター名＞ | -help}
```

機能

指定したインプロセス連携の標準提供アダプター，または SDP サーバに登録されているインプロセス連携のカスタムアダプターを停止します。

インプロセス連携のカスタムアダプター名を指定してこのコマンドを実行すると，SDP サーバに登録された StreamInprocessUP インタフェースの stop メソッドを実行します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

```
＜運用ディレクトリ＞¥bin¥
```

引数

＜アダプターグループ名＞

停止するインプロセスグループ名を指定します。アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）のインプロセスグループ定義（<InprocessGroupDefinition>タグ）の name 属性で指定したアダプターグループ名を指定してください。

＜インプロセス連携のカスタムアダプター名＞

停止するインプロセス連携のカスタムアダプター名を指定します。user_app.＜アダプター名＞.properties で設定した名称を指定してください。

-help

コマンドの使用方法を表示します。

このオプションが指定された場合，インプロセス連携の標準提供アダプターの停止，インプロセス連携のカスタムアダプターの停止，およびオプションの不正チェックをしないでコマンドを終了します。

注意事項

- カスタムアダプターの場合，StreamInprocessUP インタフェースの execute メソッドで起動したユーザースレッドは，stop メソッドの中で必ず停止してください。stop メソッドに停止処理を記述しなかった場合，または stop メソッドで停止できない無限ループなどの処理がユーザースレッドに含まれる場合，ユーザースレッドが停止されないで残ることがあります。
- カスタムアダプターの場合，StreamInprocessUP インタフェースの execute メソッドに無限ループなどの処理を記述したときに，execute メソッド自身の実行スレッドが残ってしまうことがあります。
- このコマンドは stop メソッドが終了するまでリターンしません。そのため，stop メソッドで時間の掛かる処理を記述した場合，コマンドがリターンしないおそれがあります。
- stop メソッドがエラーになった場合でも，停止処理は続行します。

戻り値

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。

sdptplls（タプル情報の表示）

形式

```
sdptplls  {-file ＜タプルログファイル＞［ ＜タプルログファイル＞...］
          ［-time ［＜開始時刻＞］［,＜終了時刻＞］］ ［-info {file｜tuple} ］
          ［-csv］ ［-data］ ｜-file ＜タプルログファイル＞
          ［ ＜タプルログファイル＞...］ -check｜-help}
```

機能

タプルログファイルに取得したタプルの情報を表示します。

実行権限

運用ユーザー

コマンド実行の前提条件

なし

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

-file ＜タプルログファイル＞［ ＜タプルログファイル＞...］

表示するタプルログファイルのパスを指定します。

複数のタプルログファイルを指定する場合は，半角スペースで区切って指定します。この場合，指定したファイルの順番に情報が表示されます。

-time ［＜開始時刻＞］［,＜終了時刻＞］

表示するタプルの開始時刻と終了時刻を，次の形式で指定します。

hhmmsslll［MMDD［YYYY］］

hh：時（00≦hh≦23）

mm：分（00≦mm≦59）

ss：秒（00≦ss≦59）

lll：ミリ秒（000≦lll≦999）

MM：月（01≦MM≦12）

DD：日（01≦DD≦31）

YYYY：年（4 けたの西暦）

- 半角数字で指定してください。
- 開始，または終了の「年」の指定を省略した場合は，当年の指定月日時刻と見なされます。
- 「年，月，日」の指定を省略した場合，当年当月当日の指定時刻と見なされます。
- 「月，日」，「月」，または「日」だけを省略できません。省略した場合はオプションエラーになります。
- 「月」または「日」を省略したい場合は，「年」，「月」，「日」のすべてを省略してください。

開始時刻，終了時刻はタプルの取得時刻で指定します。タプルの取得時刻は，このコマンドで確認できます。

開始時刻と終了時刻は，スペースを挿入しないでコンマで区切って指定します。

時刻の指定方法によって，それぞれ，次のように表示されます。

- 開始時刻を省略した場合
  終了時刻以前のタプルが表示されます。
- 終了時刻を省略した場合
  開始時刻以降のタプルが表示されます。
- 開始時刻，および終了時刻の両方を省略した場合
  すべてのタプルが表示されます。
- 開始時刻と終了時刻に同じ値を指定した場合
  指定した時刻のタプルだけが表示されます。
- 終了時刻よりもあとの時間を開始時刻に指定した場合
  オプションエラーになります。

なお，-info オプションで file を指定した場合，この引数の指定は無効になります。

-info {file | tuple}

表示する情報の種類を指定します。

- file：ファイルの情報を表示します。
- tuple：タプルの情報を表示します。

指定を省略した場合は，タプルの情報が表示されます。

-csv

表示内容を，CSV ファイル形式で表示します。

このオプションを指定した場合，タイムスタンプなどの時刻情報は，日付と時刻で一つのデータとして扱われます。

-data

タプルのデータオブジェクトの内容を表示します。

-info オプションで file を指定した場合，このオプションの指定は無効になります。

-csv オプションを指定した場合，このオプションで表示されるタプルのデータオブジェクトは一つのデータとして扱われます。

-check

タプルログファイルに，タプルの欠落があるかを確認します。

-help

コマンドの使用方法を表示します。

このオプションが指定された場合，タプルの情報の表示，およびオプションの不正チェックをしないでコマンドを終了します。

注意事項

デフォルトと異なる文字コードの環境で取得したタプルログファイルを指定した場合，タプルの文字列情報が正しく復元されないことがあります。

戻り値

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。

出力形式

-info file オプション指定時

-csv オプションを指定しない場合

```
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

...（ファイルごとに繰り返し）
```

-csv オプションを指定する場合

```
File Name,Version,Query Group Name,Stream Name,Time Stamp Mode,Stream Type,Base
Time,Update Timeaa...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...（ファイルごとに繰り返し）
```

出力形式中の，各変数の意味を次に示します。

変数	意味
aa...aa	タプルログファイルのファイルパス（絶対パス）
bb-bb-bb	タプルログを取得した Stream Data Platform - AF のバージョン情報
cc...cc	タプルログを取得したクエリグループ名（1～64 文字の文字列）
dd...dd	タプルログを取得したストリーム名（1～100 文字の文字列）
ee...ee	タプルログを取得したクエリグループのタイムスタンプモード 次のどちらかが表示されます。 • DataSource：データソースモード • Server：サーバモード
ff...ff	タプルログを取得したストリームの種別 次のどちらかが表示されます。 • Input：入力ストリーム • Output：出力ストリーム

変数	意味
gggg/gg/gg gg:gg:gg ggg	タプルログ取得基準時刻（年/月/日 時:分:秒 ミリ秒） タプルログ取得基準時刻とは，クエリグループを初期化する契機でタプルログファイルに出力される時刻情報です。この時刻が同じタプルログファイルが，クエリの再実行に必要な一連のタプルログとなります。
hhhh/hh/hh hh:hh:hh hhh	タプルログファイルの更新開始時刻（年/月/日 時:分:秒 ミリ秒）

-info tuple オプション指定時

-csv オプション，および-data オプションを指定しない場合

```
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
...（タプルごとに繰り返し）

...（ファイルごとに繰り返し）
```

-csv オプションを指定しないで，-data オプションを指定する場合

```
File Name : ii...ii
Record No Record Time             Time Stamp              Status   Data
jj...jj   kkkk/kk/kk kk:kk:kk kkk llll/ll/ll ll:ll:ll lll mm...mm  nn...nn
...（タプルごとに繰り返し）

...（ファイルごとに繰り返し）
```

-csv オプションを指定し，-data オプションを指定しない場合

```
File Name,Record No,Record Time,Time Stamp,Status
ii...ii,jj...jj,kkkk/kk/kk kk:kk:kk kkk, llll/ll/ll ll:ll:ll lll,mm...mm
...（タプルごとに繰り返し）

...（ファイルごとに繰り返し）
```

-csv オプション，および-data オプションを指定する場合

```
File Name,Record No,Record Time,Time Stamp,Status,Data
ii...ii,jj...jj,kkkk/kk/kk kk:kk:kk kkk,llll/ll/ll ll:ll:ll lll,mm...mm,nn...nn
...（タプルごとに繰り返し）

...（ファイルごとに繰り返し）
```

出力形式中の，各変数の意味を次に示します。

変数	意味
ii...ii	タプルログファイルのファイルパス（絶対パス）
jj...jj	タプルログに割り当てられる通番（19 けたの 10 進数）
kkkk/kk/kk kk:kk:kk kkk	タプルログを取得した時刻（年/月/日 時:分:秒 ミリ秒）
llll/ll/ll ll:ll:ll lll	タプルのタイムスタンプ（年/月/日 時:分:秒 ミリ秒）
mm...mm	タプルの状態 次のどちらかが表示されます。 • Put：正常 • TimeBack：時刻の逆転による破棄
nn...nn	データオブジェクトの内容（[カラム 1 | カラム 2 | ...]） -data オプションを指定した場合にだけ表示します。

-check オプション指定時

```
File Name : oo...oo
First No Last No
pp...pp  qq...qq
Lost Norr...rr-rr...rr
...（欠落ごとに繰り返し）

...（ファイルごとに繰り返し）
```

出力形式中の，各変数の意味を次に示します。

変数	意味
oo...oo	タプルログファイルのファイルパス（絶対パス）
pp...pp	タプルログファイルの先頭の通番（19 けたの 10 進数）
qq...qq	タプルログファイルの末尾の通番（19 けたの 10 進数）
rr...rr-rr...rr	タプルが欠落している範囲（欠落範囲の先頭の通番-欠落範囲の末尾の通番）

sdptplput（タプルの再投入）

形式

```
sdptplput  {-file ＜タプルログファイル＞［ ＜タプルログファイル＞...］
           ［-time ［＜開始時刻＞］［,＜終了時刻＞］］ ［-interval ＜投入間隔＞］
           ［-count ＜投入個数＞］｜-help}
```

機能

タプルログファイルに取得したタプルを入力ストリームキューに再投入します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

-file ＜タプルログファイル＞［ ＜タプルログファイル＞...］

再投入するタプルログファイルのパスを指定します。

複数のタプルログファイルを指定する場合は，半角スペースで区切って指定します。

ファイルを複数指定する場合は，次のことに注意してください。

- タプルログ取得基準時刻が異なるタプルログファイルを一度に指定すると，ファイル指定エラーになります。

  タプルログ取得基準時刻とは，クエリグループを初期化する契機で出力される時刻情報です。この時刻が同じタプルログファイルがクエリの再実行に必要な一連のタプルログとなります。

  各タプルログファイルのタプルログ取得基準時刻は，sdptplls コマンドで確認できます。
- タプルログが複数ファイルに出力された場合や，複数の入力ストリームを持つクエリグループに再投入する場合，取得したすべてのタプルログファイルを指定しなくてもエラーになりません。指定されたファイルで再投入します。
- 同一のタプルログファイルを複数指定した場合，ファイル指定エラーになります。
- 同一ストリームキューのタプルログファイルを複数指定した場合，ファイル更新時刻が古いファイルから再投入します。

  各タプルログファイルのファイル更新時刻は，sdptplls コマンドで確認できます。

-time ［＜開始時刻＞］［,＜終了時刻＞］

再投入するタプルの開始時刻と終了時刻を，次の形式で指定します。

hhmmsslll［MMDD［YYYY]]

hh：時（00≦hh≦23）

mm：分（00≦mm≦59）

ss：秒（00≦ss≦59）

lll：ミリ秒（000≦lll≦999）

MM：月（01≦MM≦12）

DD：日（01≦DD≦31）

YYYY：年（4 けたの西暦）

- 半角数字で指定してください。
- 開始，または終了の「年」の指定を省略した場合は，当年の指定月日時刻と見なされます。
- 「年，月，日」の指定を省略した場合，当年当月当日の指定時刻と見なされます。
- 「月，日」,「月」，または「日」だけを省略できません。省略した場合はオプションエラーになります。
- 「月」または「日」を省略したい場合は,「年」,「月」,「日」のすべてを省略してください。

開始時刻，終了時刻はタプルの取得時刻で指定します。タプルの取得時刻は，sdptplls コマンドで確認できます。

開始時刻と終了時刻は，スペースを挿入しないでコンマで区切って指定します。

時刻の指定方法によって，それぞれ次のように再投入されます。

- 開始時刻を省略した場合
  終了時刻以前のタプルが再投入されます。
- 終了時刻を省略した場合
  開始時刻以降のタプルが再投入されます。
- 開始時刻，および終了時刻の両方を省略した場合
  すべてのタプルが再投入されます。
- 開始時刻と終了時刻に同じ値を指定した場合
  指定した時刻のタプルだけが再投入されます。
- 終了時刻よりもあとの時間を開始時刻に指定した場合
  オプションエラーになります。

-interval ＜投入間隔＞（1～600000）

タプルを再投入する間隔をミリ秒単位で指定します。指定を省略した場合，100 ミリ秒が設定されます。

-count オプションで指定した個数のタプルを再投入後，指定した間隔を空けて再投入を開始します。また，入力ストリームキューのキューあふれが発生した場合，再投入に失敗したタプルを指定した間隔を空けて，再度投入します。

-count オプションを指定していない場合，-interval オプションによる投入間隔は，入力ストリームキューのキューあふれが発生したときだけ有効になります。

-count ＜投入個数＞（1～1048576）

一度に再投入するタプルの個数を指定します。

指定した個数のタプルを再投入したあと，-interval オプションで指定した間隔を空けて再投入を開始します。

指定を省略した場合，間隔を空けないですべてのタプルを再投入します。

-help

コマンドの使用方法を表示します。

このオプションを指定した場合，タプルの再投入，およびオプションの不正チェックはされないでコマンドを終了します。

注意事項

- このコマンドは，複数の入力ストリームキューのタプル入力待ち合わせによるタイムアウトを再現できません。
- デフォルトと異なる文字コードの環境で取得したタプルログファイルを指定した場合，タプルの文字列情報が正しく復元されないことがあります。
- このコマンドを実行した場合，入力アダプターでのタプルの投入と同様にタプルログが取得されます。再投入するときは，使用するタプルログファイルを運用ディレクトリから退避し，退避したファイルを使用して実行してください。
- このコマンドは，最後の入力タプルを再投入してから putEnd メソッドを実行するまでの間隔を再現できません。そのため，サーバモードで取得したタプルログファイルを使用したクエリの再実行では，最後の入力タプルを投入してから putEnd メソッドを実行するまでの間の結果が，入力アダプターを使用してクエリを実行した結果と異なる場合があります。
- このコマンドは，SDP サーバ内でタプルを再投入します。そのため，コマンドプロセスを強制終了した場合，タプルの再投入を継続します。継続中のタプルの再投入を中止するには，sdpcqlstop コマンドでクエリグループを停止してください。
- このコマンドを，-interval オプション，および-count オプションを指定して実行した場合，クエリグループの状態の変更は，-interval に指定した間隔で検知されます。そのため，このコマンドの実行中にクエリグループを停止させても，すぐに再投入を中止しないことがあります。

戻り値

値	意味
0	コマンドが正常に終了しました。
1	コマンドでエラーが発生しました。

sdptrced（トレース情報の編集）

形式

```
sdptrced {-file ＜トレースファイル名＞［ ＜トレースファイル名＞…］
          [-time ［＜開始時刻＞］ [,＜終了時刻＞］]
          [-threadid ＜スレッドID＞［ ＜スレッドID＞…］］ [-csv] |-help}
```

機能

API トレース情報を編集して出力します。

実行権限

運用ユーザー

コマンド実行の前提条件

このコマンドの実行には，次のファイルが必要です。

- システムコンフィグプロパティファイル（system_config.properties）
- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

格納先ディレクトリ

＜運用ディレクトリ＞¥bin¥

引数

-file ＜トレースファイル名＞［ ＜トレースファイル名＞…］

編集するトレースファイルのパス名を絶対パスで指定します。複数のトレースファイル名を指定する場合は，トレースファイル名とトレースファイル名の間を半角スペースで区切って指定します。

-time ［＜開始時刻＞］[,＜終了時刻＞]

出力したいトレース情報が取得された開始時刻と終了時刻を，次の形式で指定します。

hhmmss［MMDD［YYYY］］

hh：時（00≦hh≦23）

mm：分（00≦mm≦59）

ss：秒（00≦ss≦59）

MM：月（01≦MM≦12）

DD：日（01≦DD≦31）

YYYY：年（4 けたの西暦）

- 半角英数字で指定してください。
- 開始，または終了の「年」の指定を省略した場合は，当年の指定月日時刻と見なされます。
- 「年，月，日」の指定を省略した場合，当年当月当日の指定時刻と見なされます。
- 「月，日」，「月」，または「日」だけを省略できません。省略した場合はオプションエラーになります。
- 「月」または「日」を省略したい場合は，「年」，「月」，「日」のすべてを省略してください。

開始時刻と終了時刻は，スペースを挿入しないでコンマで区切って指定します。

時刻の指定方法によって，それぞれ，次のように出力されます。

- 開始時刻を省略した場合
  終了時刻以前のトレース情報が出力されます。
- 終了時刻を省略した場合
  開始時刻以降のトレース情報が出力されます。
- 開始時刻，および終了時刻の両方を省略した場合
  すべてのトレース情報が出力されます。
- 開始時刻と終了時刻に同じ値を指定した場合
  指定した時刻のトレース情報が出力されます。
- 終了時刻よりもあとの時間を開始時刻に指定した場合
  オプションエラーになります。

-threadid ＜スレッド ID＞［＜スレッド ID＞…］

指定されたスレッド ID のスレッドによって出力されたトレース情報だけ出力します。

スレッド ID は，半角数字で指定します。指定できる値は，0～9223372036854775807 です。複数のスレッド ID を指定する場合は，スレッド ID とスレッド ID の間を半角スペースで区切って指定します。

-csv

トレース情報の編集結果を CSV 形式で表示します。

-help

コマンドの使用方法を表示します。

このオプションが指定された場合，トレース情報の出力およびオプションの不正チェックはしないでコマンドを終了します。

注意事項

- API トレースに出力されるトレース情報は，時系列でソートされていません。
- -file オプションで指定したトレースファイルのサイズが大きい場合，編集に時間が掛かることがあります。

戻り値

値	意味
0	コマンドが正常終了しました。
1	コマンドが異常終了しました。

出力形式

出力形式を次に示します。出力例については，「6.3.2　API トレースの詳細」を参照してください。

-csv オプションを指定しない場合

```
Ver       Kind  File
aa-aa-aa  bbb   cc...cc
Date       Time     nSec      Current         Parent          Root            EventID
dddd/dd/dd ee:ee:ee fffffffff gg...gg-hh...hh ii...ii-jj...jj kk...kk-ll...ll 0xmmmmmmmm
Rc        Data
noo...oo  pp...pp
```

-csv オプションを指定する場合

```
Ver,Kind,File
aa-aa-aa,bbb,cc...cc
Date,Time,nSec,Current,Parent,Root,EventID,Rc,Data
dddd/dd/dd,ee:ee:ee,fffffffff,gg...gg-hh...hh,ii...ii-jj...jj,kk...kk-ll...ll,0xmmmmmmmm,
noo...oo,pp...pp
```

出力形式中の，各変数の意味を次に示します。

変数	意味
aa-aa-aa	トレースファイルを出力した Stream Data Platform - AF のバージョン情報
bbb	トレースファイルの種別情報 次のどちらかが表示されます。 • API：API トレース • None：Stream Data Platform - AF のトレースファイル以外
cc...cc	トレースファイルのパス（絶対パス）
dddd/dd/dd	トレースの出力年月日（年/月/日）
ee:ee:ee	トレースの出力時刻（時:分:秒）
fffffffff	トレース出力時刻のナノ秒の部分（9 けたの 10 進数）
gg...gg-hh...hh	このトレースに割り当てられた通番（スレッド ID（19 けたの 10 進数）-処理通番（10 けたの 10 進数)）
ii...ii-jj...jj※	予備領域 19 けたの 10 進数-10 けたの 10 進数で出力されます。
kk...kk-ll...ll	トレースの起点に割り当てた通番（スレッド ID（19 けたの 10 進数）-処理通番（10 けたの 10 進数)） トレースの起点とは，このトレースを取得した処理で扱ったタプルがクエリに入力（またはクエリ内で生成）された時点です。この通番が同じトレースは，同じタプルに対する一連の処理であることを示します。
mmmmmmmm	イベント ID（8 けたの 16 進数） 次のどちらかが表示されます。 • 0x00020000：クエリ処理（タプルの入力） • 0x00020001：クエリ処理（タプルの出力）
noo...oo※	予備領域 n は符号部です。符号部を含めて 11 けたの 10 進数で出力されます。n には半角スペースが表示されます。
pp...pp	トレース情報を取得したクエリ名称（クエリ名称の先頭から最大 32 バイト）

注※

Stream Data Platform - AF 01-00 では，半角スペースと 10 けたの 0 が出力されます。

8 SDP サーバ用定義ファイル

この章では，SDP サーバ用定義ファイルについて説明します。

8.1　SDP サーバ用定義ファイルの説明の記述形式

この章では，SDP サーバ用定義ファイルの説明を次の形式で記述します。ただし，ファイルによっては説明しない項目もあります。

記述形式

ファイルの記述形式を示します。

ファイル名

ファイル名を示します。

ファイルの格納先

ファイルの格納先を示します。

説明

ファイルの用途について説明します。

指定できるパラメーター

ファイルの中で指定できるパラメーターについて説明します。※

注※

システムコンフィグプロパティファイル（system_config.properties）などの一部のファイルについては，パラメーターの詳細を別の項で説明します。

注意事項

ファイル作成時の注意事項について説明します。

8.2　SDP サーバ用定義ファイル作成上の注意事項

SDP サーバ用定義ファイルを作成する際の注意事項を次に示します。

- #で始まる行は，コメント行として扱われます。
- ファイルパスを記述する際のファイルセパレーターは「¥¥」と記述してください。
  記述例を次に示します。
  ```
  SDP_CLASS_PATH=C:¥¥sdp¥¥AP
  ```
- システム起動後は，次のファイルの内容を変更しないでください。
  - SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）
  - システムコンフィグプロパティファイル（system_config.properties）
  - ログファイル出力用プロパティファイル（logger.properties）

  ファイルの内容を変更する場合は，「5.3.1　プロパティファイルの設定値の変更」を参照してください。
- SDP サーバ用定義ファイルは，java.util.Properties クラスの仕様に従って記述してください。

8.3　SDP サーバ用定義ファイルの一覧

SDP サーバ用定義ファイルの一覧を次の表に示します。ファイルの詳細については，表中の参照先を参照してください。

表 8-1　SDP サーバ用定義ファイルの一覧

項番	分類	定義ファイル	概要	参照先
1	JavaVM の起動オプションの設定	SDP サーバ用 JavaVM オプションファイル (jvm_options.cfg)	運用ディレクトリごとに必ず作成します。 このファイルには，SDP サーバを実行する JavaVM オプションを設定します。なお，インプロセス連携アダプターは，このファイルで設定したオプションで動作します。	8.4
2		RMI 連携用 JavaVM オプションファイル (jvm_client_options.cfg)	RMI 連携をする場合だけ，運用ディレクトリごと，またはアダプターごとに作成します。 このファイルには，RMI 連携アダプターを実行する JavaVM オプションを設定します。	8.5
3	動作環境プロパティの設定	システムコンフィグプロパティファイル (system_config.properties)	運用ディレクトリごとに必ず作成します。 このファイルには，SDP サーバで使用するポート番号，採取する API トレースとタプルログの詳細などを設定します。	8.6
4		クエリグループ用プロパティファイル	クエリグループごとに必ず作成します。 このファイルには，クエリ定義ファイルのパスやクエリグループを実行するときのチューニングパラメーターを設定します。	8.7
5		ストリーム用プロパティファイル	クエリグループ内のストリーム単位でチューニングパラメーターを設定したい場合だけ，ストリームごとに作成します。 このファイルを作成しない場合には，クエリグループ用プロパティファイルで設定した内容で動作します。	8.8
6		インプロセス連携用プロパティファイル	インプロセス連携をする場合だけ，標準提供アダプターはアダプターグループごと，カスタムアダプターはアダプターごとに作成します。 このファイルには，インプロセス連携アダプターのクラス名や jar ファイルのパスを設定します。	8.9
7		ログファイル出力用プロパティファイル (logger.properties)	運用ディレクトリごとに作成します。作成しない場合には，デフォルト値で動作します。 このファイルには，メッセージログやトレースログの面数やサイズを設定します。	8.10

8.4　SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）

(1)　記述形式

パラメーターは次の形式で記述します。

＜パラメーター名＞=＜JavaVMオプション＞

- コメント以外に空白は指定できません。ただし，JavaVM オプションとして空白を含むパスを指定する場合には，パスをダブルクォーテーション（"）で囲んで記述してください。記述例を次に示します。

  （例）

  ```
  SDP_JVM_LOG=-XX:HitachiJavaLog:"D:¥¥a b c¥¥SDPServerVM"
  ```

- 一つのパラメーター名に指定できる JavaVM オプションは一つです。

(2)　ファイル名

jvm_options.cfg

(3)　ファイルの格納先

このファイルは，必ず次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥

(4)　説明

sdpstart コマンドで SDP サーバを起動するとき，または Stream Data Platform - AF のコマンドを実行するときの JavaVM オプションを指定します。インプロセス連携アダプターは，SDP サーバと同じ JavaVM 上で動作するため，このファイルの SDP_USER_OPT パラメーターで指定したオプションで動作します。このファイルは，運用ディレクトリごとに必ず作成します。

なお，このファイルの各パラメーターで指定する JavaVM オプションは，次に示すように java コマンドのコマンドラインに指定されます。

SDP_CLASS_PATH パラメーターの場合

JavaVM オプションは，セミコロン区切りで，java コマンドの-classpath オプションに指定されます。

ほかのパラメーターの場合

JavaVM オプションは，半角スペース区切りで，java コマンドの引数として指定されます。

(5)　指定できるパラメーター

指定できるパラメーターとデフォルト値を次の表に示します。なお，JavaVM オプションについては，「8.11　JavaVM オプションの一覧」を参照してください。

表 8-2　指定できるパラメーターとデフォルト値（jvm_options.cfg）

項番	パラメーター名	内容	デフォルト値
1	SDP_CLASS_PATH	インプロセス連携アダプターが使用する jar ファイルを指定します。指定するファイルは，システム構成によって異なります。※	なし

項番	パラメーター名	内容	デフォルト値
1	SDP_CLASS_PATH	複数の jar ファイルを指定する場合，jar ファイルごとに指定します。指定例を次に示します。 （例） SDP_CLASS_PATH=. SDP_CLASS_PATH=C:¥¥sdp¥¥AP なお，jar ファイルを相対パスで指定する場合は，運用ディレクトリからの相対パスを記述してください。	なし
2	SDP_CLASSLIB_TRACE	クラスライブラリのスタックトレースを出力するかどうかを指定します。	-XX:-HitachiJavaClassLibTrace
3	SDP_CLASSLIB_TRACE_LINESIZE	クラスライブラリのスタックトレースの 1 行の文字数を指定します。	-XX:HitachiJavaClassLibTraceLineSize=1024
4	SDP_GC	ガーベージコレクションの発生時に拡張 verbosegc 情報を出力するかどうかを指定します。	-XX:-HitachiVerboseGC
5	SDP_GC_PRINT_CAUSE	ガーベージコレクションの要因内容を出力するかどうかを指定します。	-XX:+HitachiVerboseGCPrintCause
6	SDP_INITIAL_MEM_SIZE	Java ヒープの初期サイズを指定します。 なお，このパラメーターには，「2.7.1　ストリームデータ処理エンジンに関するメモリ使用量の見積もり」，および「2.7.2　標準提供アダプターに関するメモリ使用量の見積もり」で計算した値を指定します。	-Xms2048k
7	SDP_JVM_LOG	ログファイル名のプレフィックスを指定します。	-XX:HitachiJavaLog:javalog
8	SDP_JVM_LOG_FILE_SIZE	ログファイルの 1 ファイルの最大ファイルサイズを指定します。	-XX:HitachiJavaLogFileSize=256k
9	SDP_LOCALS_IN_STACK_TRACE	スレッドダンプ出力時のスタックトレースに，ローカル変数情報を出力するかどうかを指定します。	-XX:-HitachiLocalsInStackTrace
10	SDP_LOCALS_SIMPLE_FORMAT	ローカル変数情報出力を簡易フォーマットにするかどうかを指定します。	-XX:-HitachiLocalsSimpleFormat
11	SDP_MAX_MEM_SIZE	Java ヒープの最大サイズを指定します。 なお，このパラメーターには，「2.7.1　ストリームデータ処理エンジンに関するメモリ使用量の見積もり」，および「2.7.2　標準提供アダプターに関するメモリ使用量の見積もり」で計算した値を指定します。	-Xmx64m
12	SDP_MAX_PERM_SIZE	Permanent 領域の最大サイズを指定します。	-XX:MaxPermSize=64m

項番	パラメーター名	内容	デフォルト値
13	SDP_NEW_RATIO	DefNew 領域に対する Tenured 領域の割合を指定します。	-XX:NewRatio=2
14	SDP_OOM_STACK_TRACE	OutOfMemoryError 発生時のスタックトレースを出力するかどうかを指定します。	-XX:-HitachiOutOfMemoryStackTrace
15	SDP_OUTPUT_MILLI_TIME	ミリ秒までの時間を出力するかどうかを指定します。	-XX:-HitachiOutputMilliTime
16	SDP_PERM_SIZE	Permanent 領域の初期サイズを指定します。	-XX:PermSize=16m
17	SDP_SYS_OPT	Stream Data Platform - AF のコマンドの実行に必要な JavaVM オプションを指定します。 システム環境に依存して，コマンドでも JavaVM オプションを指定する必要がある場合に，このパラメーターを指定します。 このパラメーターの指定は，Stream Data Platform - AF のコマンドを実行する場合だけ有効です。	なし
18	SDP_THRD_DUMP	標準出力にスレッドダンプを出力するかどうかを指定します。	-XX:+HitachiThreadDumpToStdout
19	SDP_TRUE_TYPE_IN_LOCALS	ローカル変数情報出力時に，ローカル変数オブジェクトの実際の型名を文字列として出力するかどうかを指定します。	-XX:-HitachiTrueTypeInLocals
20	SDP_USER_OPT	次のどちらかの場合に，このパラメーターで JavaVM オプションを指定します。 • 運用ユーザーが JavaVM オプションを追加したい場合 • インプロセス連携アダプターを実行するときの JavaVM オプションを指定したい場合 複数の JavaVM オプションを指定する場合には，オプションごとに指定します。指定例を次に示します。 （例） SDP_USER_OPT=-Dxxx=www SDP_USER_OPT=-Dyyy=zzz なお，同一のオプションを複数回指定した場合は，あとで指定したオプション（ファイルの末尾に近い方のオプション）が有効になります。	なし

注※

標準提供アダプターの場合は，指定値は固定です。次の値を指定してください。

SDP_CLASS_PATH=＜インストールディレクトリ＞¥¥psb¥¥jaxp¥¥lib¥¥csmjaxb.jar

SDP_CLASS_PATH=＜インストールディレクトリ＞¥¥psb¥¥jaxp¥¥lib¥¥csmjaxp.jar

SDP_CLASS_PATH=＜インストールディレクトリ＞¥¥psb¥¥jaxp¥¥lib¥¥csmstax.jar

SDP_CLASS_PATH=＜インストールディレクトリ＞¥¥psb¥¥CC¥¥lib¥¥hitj2ee.jar

(6)　注意事項

- リモートオブジェクトの参照に対して定期的にガーベージコレクションが発生することがあります。**ガーベージコレクションの発生間隔**は，RMI 連携時に指定する Java の sun.rmi.dgc.client.gcInterval プロパティ，および sun.rmi.dgc.server.gcInterval プロパティにミリ秒単位で指定できます。デフォルトは 60,000 ミリ秒（60 秒）です。ガーベージコレクションの発生間隔を変更する場合は，SDP_USER_OPT パラメーターに sun.rmi.dgc.client.gcInterval プロパティ，および sun.rmi.dgc.server.gcInterval プロパティを指定し，任意の間隔を指定してください。指定できる値の範囲は，1～9,223,372,036,854,775,806 です。指定例を次に示します。

  （例）

  ```
  SDP_USER_OPT=-Dsun.rmi.dgc.client.gcInterval=1000000000
  SDP_USER_OPT=-Dsun.rmi.dgc.server.gcInterval=1000000000
  ```

8.5　RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）

(1)　記述形式

パラメーターは次の形式で記述します。

＜パラメーター名＞=＜JavaVMオプション＞

- コメント以外に空白は指定できません。ただし，JavaVM オプションとして空白を含むパスを指定する場合には，パスをダブルクォーテーション（"）で囲んで記述してください。記述例を次に示します。

  （例）

  ```
  SDP_JVM_LOG=-XX:HitachiJavaLog:"D:¥¥a b c¥¥SDPServerVM"
  ```

- 一つのパラメーター名に指定できる JavaVM オプションは一つです。

(2)　ファイル名

jvm_client_options.cfg

(3)　ファイルの格納先

このファイルの格納先は任意です。このファイルの格納先は，RMI 連携アダプターを起動するときの sdpstartap コマンドの引数で指定します。引数の指定を省略した場合には，「＜運用ディレクトリ＞¥conf¥」になります。

sdpstartap コマンドについては，「7.　コマンド」の「sdpstartap（RMI 連携アダプターの起動）」を参照してください。

(4)　説明

RMI 連携アダプターを起動するときの JavaVM オプションを指定します。このファイルは，RMI 連携をする場合だけ，運用ディレクトリごと，またはアダプターごとに作成します。

なお，このファイルの各パラメーターで指定する JavaVM オプションは，次に示すように java コマンドのコマンドラインに指定されます。

SDP_CLASS_PATH パラメーターの場合

JavaVM オプションは，セミコロン区切りで，java コマンドの-classpath オプションに指定されます。

ほかのパラメーターの場合

JavaVM オプションは，半角スペース区切りで，java コマンドの引数として指定されます。

(5)　指定できるパラメーター

指定できるパラメーターとデフォルト値を次の表に示します。なお，JavaVM オプションについては，「8.11　JavaVM オプションの一覧」を参照してください。

表 8-3　指定できるパラメーターとデフォルト値（jvm_client_options.cfg）

項番	パラメーター名	内容	デフォルト値
1	SDP_CLASS_PATH	RMI 連携アダプターが使用する jar ファイルを指定します。指定するファイルは，システム構成によって異なります。※ 複数の jar ファイルを指定する場合，jar ファイルごとに指定します。指定例を次に示します。 （例） SDP_CLASS_PATH=. SDP_CLASS_PATH=C:¥¥sdp¥¥AP なお，jar ファイルを相対パスで指定する場合は，運用ディレクトリからの相対パスを記述してください。	なし
2	SDP_CLASSLIB_TRACE	クラスライブラリのスタックトレースを出力するかどうかを指定します。	-XX:-HitachiJavaClassLibTrace
3	SDP_CLASSLIB_TRACE_LINESIZE	クラスライブラリのスタックトレースの 1 行の文字数を指定します。	-XX:HitachiJavaClassLibTraceLineSize=1024
4	SDP_GC	ガーベージコレクションの発生時に拡張 verbosegc 情報を出力するかどうかを指定します。	-XX:-HitachiVerboseGC
5	SDP_GC_PRINT_CAUSE	ガーベージコレクションの要因内容を出力するかどうかを指定します。	-XX:+HitachiVerboseGCPrintCause
6	SDP_INITIAL_MEM_SIZE	Java ヒープの初期サイズを指定します。 なお，このパラメーターには，「2.7.1　ストリームデータ処理エンジンに関するメモリ使用量の見積もり」，および「2.7.2　標準提供アダプターに関するメモリ使用量の見積もり」で計算した値を指定します。	-Xms2048k
7	SDP_JVM_LOG	ログファイル名のプレフィックスを指定します。	-XX:HitachiJavaLog:javalog
8	SDP_JVM_LOG_FILE_SIZE	ログファイルの 1 ファイルの最大ファイルサイズを指定します。	-XX:HitachiJavaLogFileSize=256k
9	SDP_LOCALS_IN_STACK_TRACE	スレッドダンプ出力時のスタックトレースに，ローカル変数情報を出力するかどうかを指定します。	-XX:-HitachiLocalsInStackTrace
10	SDP_LOCALS_SIMPLE_FORMAT	ローカル変数情報出力を簡易フォーマットにするかどうかを指定します。	-XX:-HitachiLocalsSimpleFormat
11	SDP_MAX_MEM_SIZE	Java ヒープの最大サイズを指定します。 なお，このパラメーターには，「2.7.1　ストリームデータ処理エンジンに関するメモリ使用量の	-Xmx64m

項番	パラメーター名	内容	デフォルト値
11	SDP_MAX_MEM_SIZE	見積もり」, および「2.7.2　標準提供アダプターに関するメモリ使用量の見積もり」で計算した値を指定します。	-Xmx64m
12	SDP_MAX_PERM_SIZE	Permanent 領域の最大サイズを指定します。	-XX:MaxPermSize=64m
13	SDP_NEW_RATIO	DefNew 領域に対する Tenured 領域の割合を指定します。	-XX:NewRatio=2
14	SDP_OOM_STACK_TRACE	OutOfMemoryError 発生時のスタックトレースを出力するかどうかを指定します。	-XX:-HitachiOutOfMemoryStackTrace
15	SDP_OUTPUT_MILLI_TIME	ミリ秒までの時間を出力するかどうかを指定します。	-XX:-HitachiOutputMilliTime
16	SDP_PERM_SIZE	Permanent 領域の初期サイズを指定します。	-XX:PermSize=16m
17	SDP_THRD_DUMP	標準出力にスレッドダンプを出力するかどうかを指定します。	-XX:+HitachiThreadDumpToStdout
18	SDP_TRUE_TYPE_IN_LOCALS	ローカル変数情報出力時に，ローカル変数オブジェクトの実際の型名を文字列として出力するかどうかを指定します。	-XX:-HitachiTrueTypeInLocals
19	SDP_USER_OPT	運用ユーザーが JavaVM オプションを追加したい場合に, このパラメーターで JavaVM オプションを指定します。 複数の JavaVM オプションを指定する場合には，オプションごとに指定します。記述例を次に示します。 (例) SDP_USER_OPT=-Dxxx=www SDP_USER_OPT=-Dyyy=zzz なお，同一のオプションを複数回指定した場合は，あとで指定したオプション（ファイルの末尾に近い方のオプション）が有効になります。	なし

注※

標準提供アダプターの場合は，指定値は固定です。次の値を指定してください。

SDP_CLASS_PATH=＜インストールディレクトリ＞¥¥psb¥¥jaxp¥¥lib¥¥csmjaxb.jar

SDP_CLASS_PATH=＜インストールディレクトリ＞¥¥psb¥¥jaxp¥¥lib¥¥csmjaxp.jar

SDP_CLASS_PATH=＜インストールディレクトリ＞¥¥psb¥¥jaxp¥¥lib¥¥csmstax.jar

SDP_CLASS_PATH=＜インストールディレクトリ＞¥¥psb¥¥CC¥¥lib¥¥hitj2ee.jar

(6)　注意事項

- リモートオブジェクトの参照に対して定期的にガーベージコレクションが発生することがあります。**ガーベージコレクションの発生間隔**は，RMI 連携時に指定する Java の sun.rmi.dgc.client.gcInterval プロパティ, および sun.rmi.dgc.server.gcInterval プロパティにミリ秒単位で指定できます。デフォルトは 60,000 ミリ秒（60 秒）です。ガーベージコレクションの発生間隔を変更する場合は，SDP_USER_OPT パラメーターに sun.rmi.dgc.client.gcInterval プロパティ，および

sun.rmi.dgc.server.gcInterval プロパティを指定し，任意の間隔をで指定してください。指定できる値の範囲は，1～9,223,372,036,854,775,806 です。指定例を次に示します。

（例）

```
SDP_USER_OPT=-Dsun.rmi.dgc.client.gcInterval=1000000000
SDP_USER_OPT=-Dsun.rmi.dgc.server.gcInterval=1000000000
```

- JavaVM のログ出力先は，入力アダプターと出力アダプターで変えることをお勧めします。JavaVM のログ出力先や Java ヒープのサイズなどをアダプターごとに指定したい場合は，次の手順でアダプターごとに RMI 連携用 JavaVM オプションファイルを作成してください。

  1. RMI 連携用 JavaVM オプションファイルを用意します。

     RMI 連携用 JavaVM オプションファイルのサンプルファイルを任意のディレクトリにコピーしてください。サンプルファイルの格納ディレクトリについては，「3.2.1　インストールディレクトリの構成」を参照してください。

  2. コピーした RMI 連携用 JavaVM オプションファイルを編集します。

     なお，JavaVM のログファイル名には，SDP_JVM_LOG パラメーターで指定したプレフィックスの後ろに，2 けたの数（01～ログファイル数）が付加されます。このため，入力アダプターと出力アダプターでログ出力先を変える場合は，プレフィックス名が重複しないようにしてください。

     （例）

     入力アダプターの JavaVM のログ出力先の指定

     ```
     SDP_JVM_LOG=-XX:HitachiJavaLog:C:¥¥sdp¥¥logs¥¥SDPReceiverClientVM
     ```

     出力アダプターの JavaVM のログ出力先の指定

     ```
     SDP_JVM_LOG=-XX:HitachiJavaLog:C:¥¥sdp¥¥logs¥¥SDPSenderClientVM
     ```

  3. アダプターを起動する際に，sdpstartap コマンドの-clientcfg オプションで，個別に作成した RMI 連携用 JavaVM オプションファイルを指定します。

8.6　システムコンフィグプロパティファイル（system_config.properties）

ここでは，システムコンフィグプロパティファイル（system_config.properties）について，ファイルの詳細と，ファイルに指定するパラメーターの詳細に分けて説明します。

8.6.1　システムコンフィグプロパティファイル（system_config.properties）の詳細

(1)　記述形式

パラメーターは次の形式で記述します。

＜パラメーター名＞=＜値＞

- 値の後ろに空白やコメントなどの文字列を追加できません。
  追加した場合は値の一部として解釈されます。
- パラメーター名だけを指定して，値を指定していない行は，無視されます。

(2)　ファイル名

system_config.properties

(3)　ファイルの格納先

このファイルは，必ず次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥

(4)　説明

SDP サーバで使用するポート番号，採取する API トレースとタプルログの詳細などを設定します。このファイルは，運用ディレクトリごとに必ず作成します。

(5)　指定できるパラメーター

指定できるパラメーターとデフォルト値を次の表に示します。なお，パラメーターの詳細については，「8.6.2　システムコンフィグプロパティファイル（system_config.properties）のパラメーターの詳細」で説明します。

表 8-4　指定できるパラメーターとデフォルト値（system_config.properties）

項番	パラメーター名	内容	デフォルト値	指定できる値の範囲
1	engine.externalStreamFuncVerifyMode	ストリーム間演算の戻り値の型を検証するかどうかを指定します。	true	true または false
2	engine.initialQueueSize	ストリームキューで使用する要素数の初期値を指定します。	1024	1～1048576 の整数

項番	パラメーター名	内容	デフォルト値	指定できる値の範囲
3	engine.maxQueueSize	ストリームキューで使用する要素数の上限値を指定します。	4096	1～1048576 の整数
4	engine.watchQueueSize.threshold	ストリームキューで使用する要素数を監視するしきい値（単位：%）を指定します。	なし	1～99 の整数
5	logger.console.abnormal.enabled	ログファイルの初期化で障害が発生した場合に，SDP サーバの起動を続行させるかどうかを指定します。 • true：メッセージをコンソールに出力して SDP サーバを起動します。 • false：SDP サーバを停止します。	false	true または false
6	logger.console.enabled	SDP サーバが出力するメッセージをコンソールに出力するかどうかを指定します。 • true：メッセージをログファイルとコンソールの両方に出力します。 • false：メッセージをコンソールに出力しません。	false	true または false
7	mon.process_exp_time	タイムアウトと判定する処理時間（単位：ミリ秒）を指定します。	30000	1～2147483647 の整数
8	query.decimalMaxPrecision	クエリの算術演算の結果が DECIMAL 型となる場合の，算術演算の最大精度を指定します。	38	1～38 の整数
9	query.decimalRoundingMode	クエリの演算結果が DECIMAL 型となる場合の，query.decimalMaxPrecision パラメーターで指定した精度を超えたときの丸めの動作を指定します。	HALF_UP	HALF_UP または DOWN
10	querygroup.sleepOnOverStore	SDP サーバが出力ストリームキューに空きがあるかどうかをチェックして空きがなかった場合に，クエリグループの実行をスリープさせる時間を指定します。	100	1～2147483647
11	querygroup.sleepOnOverStoreRetryCount	SDP サーバがクエリ実行後のタプルを出力ストリームキューへ投入する前に，出力ストリームキューに空きがあるかどうかをチェックする回数を指定します。	0	0～2147483647
12	rmi.serverPort	SDP サーバのポート番号を指定します。	20400	1～65535 の整数
13	stream.freeInputQueueSizeThreshold	入力ストリームキューで使用する要素数の上限値に対する空きサイズのしきい値（単位：%）を指定します。	なし	1～99 の整数
14	stream.freeInputQueueSizeThresholdOutputMessage	SDP サーバのメッセージログに，警告メッセージを出力するかどうかを指定します。 • true：警告メッセージを出力します。	false	true または false

項番	パラメーター名	内容	デフォルト値	指定できる値の範囲
14	stream.freeInputQueueSizeThresholdOutputMessage	• false：警告メッセージを出力しません。	false	true または false
15	stream.maxKeepTupleCount	タイムスタンプ調整機能で保留できるタプル数の上限値を指定します。	125828	1～1048576 の整数
16	stream.timestampAccuracy	タイムスタンプ調整機能の時刻単位と時刻調整範囲を指定します。	なし	「8.6.2(16) stream.timestampAccuracy={{sec ｜ msec ｜ usec}，時刻調整範囲｜unuse}」を参照してください。
17	stream.timestampMode	タプルに時刻を付与するためのタイムスタンプモードを指定します。 • Server：サーバモードを使用します。 • DataSource：データソースモードを使用します。	Server	Server または DataSource
18	stream.timestampPosition	タプル内の時刻データ列名を指定します。	なし	「8.6.2(18) stream.timestampPosition=時刻データ列名」を参照してください。
19	stream.tupleLogMode	sdptplput コマンドを実行するかどうかを指定します。 • true：sdptplput コマンドを実行できます。 • false：sdptplput コマンドを実行できません。	false	true または false
20	tpl.backupFileCount	タプルログファイルのバックアップを残す最大世代数を指定します。	1	0～10 の整数
21	tpl.bufferCount	タプルログのバッファーの面数を指定します。	5	3～512 の整数
22	tpl.bufferSize	タプルログのバッファーの最大サイズ（単位：キロバイト）を指定します。	1024	1～2048000 の整数
23	tpl.fileCount	タプルログファイルの最大ファイル数を指定します。	3	3～512 の整数
24	tpl.fileSize	タプルログファイルの最大サイズ（単位：メガバイト）を指定します。	100	1～2048 の整数
25	tpl.outputLevel	タプルログの出力レベルを指定します。	3	1～3 の整数

項番	パラメーター名	内容	デフォルト値	指定できる値の範囲
25	tpl.outputLevel	• 1:ストリームキューへ格納するタプルについて，タプルログを出力します。 • 2:時刻の逆転によって破棄するタプルについて，タプルログを出力します。 • 3：レベル 1 とレベル 2 のタプルについて，タプルログを出力します。	3	1～3 の整数
26	tpl.outputTrigger	タプルログのファイルへの出力契機を指定します。 • BUFFER：対象のストリームで現在のタプルログを取得しているバッファーがいっぱいになったら，タプルログをファイルへ出力します。 • NONE：タプルログをファイルに出力しません。バッファリングもしません。	• 入力ストリームキュー：BUFFER • 出力ストリームキュー：NONE	BUFFER または NONE
27	tpl.useOverwrite	タプルログのバッファーの枯渇時にバッファーを上書きするかどうかを指定します。 • true：タプルログのバッファーを上書きします。 • false：タプルログのバッファーを上書きしません。	true	true または false
28	trc.api.bufferCount	API トレースのバッファーの面数を指定します。	3	3～512 の整数
29	trc.api.bufferSize	API トレースのバッファーの最大サイズ（単位：キロバイト）を指定します。	1024	1～2048000 の整数
30	trc.api.fileCount	API トレースを出力するファイルの最大ファイル数を指定します。	3	3～512 の整数
31	trc.api.fileSize	API トレースを出力するファイルの最大サイズ（単位：メガバイト）を指定します。	1024	1～2048 の整数
32	trc.api.ioBufferSize	API トレースの I/O バッファーの最大サイズ（単位：キロバイト）を指定します。	2048	1～2048000 の整数
33	trc.api.outputTrigger	API トレースのファイルへの出力契機を指定します。 • BUFFER：I/O スレッドのバッファーがいっぱいになったら，API トレースをファイルに出力します。 • NONE：API トレースをファイルに出力しません。バッファリングもしません。	BUFFER	BUFFER または NONE
34	trc.api.useOverwrite	API トレースのバッファーの枯渇時にバッファーを上書きするかどうかを指定します。 • true：API トレースのバッファーを上書きします。	true	true または false

項番	パラメーター名	内容	デフォルト値	指定できる値の範囲
34	trc.api.useOverwrite	• false：API トレースのバッファーを上書きしません。	true	true または false

8.6.2　システムコンフィグプロパティファイル（system_config.properties）のパラメーターの詳細

ここでは，「8.6.1(5)　指定できるパラメーター」で示したシステムコンフィグプロパティファイル（system_config.properties）のパラメーターの詳細について説明します。

(1)　engine.externalStreamFuncVerifyMode=ストリーム間演算の戻り値の型の検証

ストリーム間演算の戻り値の型を検証するかどうかを true または false で指定します。大文字，小文字の区別はされません。デフォルトは true です。

true
　検証します。

false
　検証しません。

false を指定することで，ストリーム間演算関数を頻繁に呼び出すシステムで性能が改善する場合があります。ただし，検証をしない場合は，トラブルシュートが困難になるので注意してください。

このパラメーターは，システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.）。

1. クエリグループ用プロパティファイル
2. システムコンフィグプロパティファイル

(2)　engine.initialQueueSize=ストリームキューで使用する要素数の初期値

ストリームキューで使用する要素数の初期値を 1～1048576 の整数で指定します。デフォルトは 1024 です。

ここで指定した要素数を超える要素数をストリームキューに登録しようとした場合，engine.maxQueueSize パラメーターで設定する上限値までストリームキューが拡張されます。

なお，ここで指定した要素数の初期値は，個々のストリームキューに対する初期値になります。

(3)　engine.maxQueueSize=ストリームキューで使用する要素数の上限値

ストリームキューで使用する要素数の上限値を 1～1048576 の整数で指定します。デフォルトは 4096 です。

ストリームキューで使用する要素数の上限値は，engine.initialQueueSize パラメーターで指定した要素数の初期値以下となるように指定してください。

なお，ここで指定した要素数の上限値は，個々のストリームキューに対する上限値になります。

(4) engine.watchQueueSize.threshold=ストリームキューで使用する要素数を監視するしきい値

ストリームキューで使用する要素数を監視するしきい値（単位：%）を 1～99 の整数で指定します。このパラメーターを省略した場合，要素数は監視されません。

次の計算式で算出される値が，このパラメーターの指定値を超えると，警告メッセージが出力されます。

要素数÷engine.maxQueueSizeパラメーターの指定値×100

警告メッセージは，いったん出力されると，要素数がしきい値を下回ったあとに再びしきい値を上回るまで出力されません。

(5) logger.console.abnormal.enabled= {true | false}

ログファイルの初期化で障害が発生した場合に，SDP サーバの起動を続行させるかどうかを true または false で指定します。大文字，小文字の区別はされません。デフォルトは false です。

true
メッセージをコンソールに出力して SDP サーバを起動します。

false
SDP サーバを停止します。

(6) logger.console.enabled= {true | false}

SDP サーバが出力するメッセージをコンソールに出力するかどうかを true または false で指定します。大文字，小文字の区別はされません。デフォルトは false です。

true
メッセージをログファイルとコンソールの両方に出力します。

false
メッセージをコンソールに出力しません。

(7) mon.process_exp_time=タイムアウト処理時間

タイムアウトと判定する処理時間（単位：ミリ秒）を 1～2147483647 の整数で指定します。デフォルトは 30000 です。

デフォルト値でタイムアウト検知のメッセージが頻発する場合は，API トレースから実測した処理時間を基に，十分な余裕を持った値を指定してください。

(8) query.decimalMaxPrecision=クエリの算術演算の最大精度

クエリの算術演算の結果が DECIMAL 型（NUMERIC 型を含めます）となる場合の，算術演算の最大精度を 1～38 の整数で指定します。デフォルトは 38 です。

演算結果が DECIMAL 型となる算術演算とは，次の算術演算です。

- 演算項に DECIMAL 型を含む四則演算（2 項演算）
- 引数が DECIMAL 型となる集合関数 AVG/SUM
- DECIMAL 型以外のデータ型の DECIMAL 型へのキャスト

算術演算の結果が，このパラメーターで指定した精度を超えない場合は，算術演算の結果の精度は加工されません。超える場合は，このパラメーターで指定した精度に丸められます。丸めの動作は，query.decimalRoundingMode パラメーターで指定します。

(9)　query.decimalRoundingMode= {HALF_UP | DOWN}

クエリの演算結果が DECIMAL 型（NUMERIC 型を含めます）となる場合の，query.decimalMaxPrecision パラメーターで指定した精度を超えたときの丸めの動作を指定します。デフォルトは HALF_UP です。

HALF_UP

破棄される小数部の最上位のけたを四捨五入して丸めます。

DOWN

破棄される小数部の最上位のけたより一つ上のけたの値が減るように丸めます。

(10)　querygroup.sleepOnOverStore=クエリグループの実行をスリープさせる時間

SDP サーバが出力ストリームキューに空きがあるかどうかをチェックして空きがなかった場合に，クエリグループの実行をスリープさせる時間（単位：ミリ秒）を 1～2147483647 の整数で指定します。デフォルトは 100 です。

このパラメーターの指定は，querygroup.sleepOnOverStoreRetryCount パラメーターで 1 以上を指定した場合に有効です。

このパラメーターによってスリープしたクエリグループに対しては，次のことが実行できます。

- 入力アダプターによる入力ストリームキューへのタプルの投入
- 出力アダプターによる出力ストリームキューからのタプルの取得

このパラメーターは，システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.）。

1. クエリグループ用プロパティファイル
2. システムコンフィグプロパティファイル

(11)　querygroup.sleepOnOverStoreRetryCount=出力ストリームキューの空きをチェックする回数

SDP サーバがクエリ実行後のタプルを出力ストリームキューへ投入する前に，出力ストリームキューに空きがあるかどうかをチェックする回数を 0～2147483647 の整数で指定します。デフォルトは 0 です。

0 を指定した場合，SDP サーバは，出力ストリームキューに空きがあるかをチェックしないで，クエリ実行後のタプルを出力ストリームキューへ投入します。

1 以上を指定した場合，SDP サーバは，指定された回数だけ次の処理を実行します。

1. 出力ストリームキューに空きがあるかどうかをチェックします。
2. 出力ストリームキューに空きがなければ，出力ストリームキューのあるクエリグループの実行をスリープさせます。スリープさせる時間は，querygroup.sleepOnOverStore パラメーターで指定します。
   スリープ後，このパラメーターで指定した回数分の処理を実行していない場合は，1.の処理に戻ります。
3. 出力ストリームキューに空きがあれば，出力ストリームキューへクエリ実行後のタプルを投入します。

このパラメーターは，システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.）。

1. クエリグループ用プロパティファイル
2. システムコンフィグプロパティファイル

(12) rmi.serverPort=SDP サーバのポート番号

SDP サーバのポート番号を 1～65535 の整数で指定します。デフォルトは 20400 です。

デフォルトのポート番号（20400）が，ほかのシステムのポート番号と重複する場合は，別のポート番号を指定してください。ただし，1～1023（Well Known ポート番号）は使用しないことをお勧めします。

(13) stream.freeInputQueueSizeThreshold=入力ストリームキューの最大サイズに対する空きサイズのしきい値

入力ストリームキューで使用する要素数の上限値（engine.maxQueueSize パラメーターの指定値）に対する空きサイズのしきい値（単位：%）を 1～99 の整数で指定します。

次に示す条件を満たした場合に，put(StreamTuple tuple)メソッドまたは put(ArrayList<StreamTuple> tuple_list)メソッドから SDPClientFreeInputQueueSizeThresholdOverException の例外がスローされます。このとき，入力ストリームキューへのタプルの投入は，成功しています。

このパラメーターの指定値 ≧（（入力ストリームキューの空きサイズ÷入力ストリームキューの最大サイズ）×100）

このパラメーターを省略した場合は，しきい値のチェックによる例外は発生しません。

なお，engine.watchQueueSize.threshold パラメーターは，入力および出力ストリームキューの最大サイズに対する使用サイズのしきい値を設定するものであり，stream.freeInputQueueSizeThreshold とは異なるパラメーターなので注意してください。詳細については，engine.watchQueueSize.threshold パラメーターを参照してください。

sdptplput コマンドで入力ストリームキューにタプルを投入する場合，このパラメーターの指定は無効となります。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(14) stream.freeInputQueueSizeThresholdOutputMessage= {true | false}

SDP サーバのメッセージログに，警告メッセージ（メッセージ KFSP42032-W）を出力するかどうかを true または false で指定します。大文字，小文字の区別はされません。デフォルトは false です。

true

警告メッセージを出力します。

false

警告メッセージを出力しません。

このパラメーターの指定は，stream.freeInputQueueSizeThreshold パラメーターが指定された場合だけ有効です。

なお，警告メッセージが出力されるのは，このパラメーターで true を指定し，次の条件を満たした場合です。

stream.freeInputQueueSizeThresholdパラメーターの指定値 ≧（（入力ストリームキューの空きサイズ÷入力ストリームキューの最大サイズ）×100）

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(15)　stream.maxKeepTupleCount=タイムスタンプ調整機能で保留できるタプル数の上限値

タイムスタンプ調整機能で保留できるタプル数の上限値を 1～1048576 の整数で指定します。デフォルトは 125828 です。

このパラメーターで指定したタプル数の上限値は，個々の入力ストリームのタイムスタンプ調整機能に対する上限値となります。

タプル数がこのパラメーターで指定した上限値を超えると，クエリグループが閉塞されます。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(16)　stream.timestampAccuracy= {{sec | msec | usec} ,時刻調整範囲 | unuse}

タイムスタンプ調整機能の時刻単位と時刻調整範囲を指定します。大文字，小文字の区別はされません。

stream.timestampMode パラメーターで DataSource を指定した場合，このパラメーターを必ず指定してください。stream.timestampMode パラメーターで Server を指定した場合，このパラメーターの指定は無視されますが，パラメーターの形式チェックは実施されます。

{sec | msec | usec} ,時刻調整範囲

時刻単位と時刻調整範囲を指定します。時刻単位（sec，msec，または usec）と時刻調整範囲を区切る半角コンマ（,）の前後には，空白やタブなどを記述しないでください。記述した場合はエラーとなります。時刻単位に sec，msec，または usec のどれかを指定し，時刻調整範囲に 0 を指定した場合は基準時刻だけが時刻調整範囲となります。

それぞれの値の意味を次に示します。

sec

時刻単位として秒を使用することを指定します。

msec

時刻単位としてミリ秒を使用することを指定します。

usec

時刻単位としてマイクロ秒を使用することを指定します。

時刻調整範囲

タイムスタンプ調整での時刻の調整範囲を整数で指定します。時刻単位に指定した単位によって指定範囲が異なります。時刻単位ごとの指定範囲を次の表に示します。

時刻単位	指定範囲
sec（秒）	0～59 の整数
msec（ミリ秒）	0～999 の整数
usec（マイクロ秒）	0～999 の整数

unuse

時刻調整を行いません。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(17)　stream.timestampMode= {Server | DataSource}

タプルに時刻を付与するためのタイムスタンプモードを Server または DataSource で指定します。大文字，小文字の区別はされません。デフォルトは Server です。

Server

サーバモードを使用します。

DataSource

データソースモードを使用します。

DataSource を指定した場合は，stream.timestampAccuracy パラメーター，および stream.timestampPosition パラメーターを必ず指定してください。

このパラメーターは，システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.）。

1. クエリグループ用プロパティファイル
2. システムコンフィグプロパティファイル

(18)　stream.timestampPosition=時刻データ列名

タプル内の時刻データ列名を指定します。大文字，小文字の区別はされません。

stream.timestampMode パラメーターで DataSource を指定した場合，このパラメーターを必ず指定してください。stream.timestampMode パラメーターで Server を指定した場合，このパラメーターの指定は無視されますが，パラメーターの形式チェックは実施されます。

時刻データとして指定できるデータ型は，TIMESTAMP 型だけです。

時刻データの範囲は，GMT（グリニッジ標準時）の 1970/01/01 00:00:00.000000000 から 2261/12/31 23:59:59.999999999 までになります。それ以外の時刻を指定した場合は，ストリームデータ送信時に例外が発生します。日本標準時は GMT より 9 時間あとであることに注意してください。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(19)　stream.tupleLogMode=｛true ｜ false｝

sdptplput コマンドを実行するかどうかを true または false で指定します。大文字，小文字の区別はされません。デフォルトは false です。

true
: sdptplput コマンドを実行できます。

false
: sdptplput コマンドを実行できません。

stream.timestampMode パラメーターで Server を，このパラメーターで true を指定した場合は，StreamInput インタフェースの put メソッドでストリームデータを送信すると，SDPClientException の例外が発生します。

このパラメーターは，システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.）。

1. クエリグループ用プロパティファイル
2. システムコンフィグプロパティファイル

(20)　tpl.backupFileCount=タプルログファイルのバックアップを残す最大世代数

タプルログファイルのバックアップを残す最大世代数を 0～10 の整数で指定します。デフォルトは 1 です。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(21)　tpl.bufferCount=タプルログのバッファーの面数

タプルログのバッファーの面数を 3～512 の整数で指定します。デフォルトは 5 です。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(22)　tpl.bufferSize=タプルログのバッファーの最大サイズ

タプルログのバッファーの最大サイズ（単位：キロバイト）を 1～2048000 の整数で指定します。デフォルトは 1024 です。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(23)　tpl.fileCount=タプルログファイルの最大ファイル数

タプルログファイルの最大ファイル数を 3～512 の整数で指定します。デフォルトは 3 です。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(24)　tpl.fileSize=タプルログファイルの最大サイズ

タプルログファイルの最大サイズ（単位：メガバイト）を 1～2048 の整数で指定します。デフォルトは 100 です。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(25)　tpl.outputLevel=タプルログの出力レベル

タプルログの出力レベルを 1～3 の整数で指定します。デフォルトは 3 です。

1

ストリームキューへ格納するタプルについて，タプルログを出力します。

2

時刻の逆転によって破棄するタプルについて，タプルログを出力します。

3

レベル 1 とレベル 2 のタプルについて，タプルログを出力します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(26)　tpl.outputTrigger=｛BUFFER ｜ NONE｝

タプルログのファイルへの出力契機を BUFFER または NONE で指定します。大文字，小文字の区別はされません。入力ストリームキューの場合，デフォルトは BUFFER です。出力ストリームキューの場合，デフォルトは NONE です。

BUFFER

対象のストリームキューで現在のタプルログを取得しているバッファーがいっぱいになったら，タプルログをファイルへ出力します。

NONE

タプルログをファイルに出力しません。バッファリングもしません。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(27)　tpl.useOverwrite=｛true ｜ false｝

タプルログのバッファーの枯渇時にバッファーを上書きするかどうかを true または false で指定します。大文字，小文字の区別はされません。デフォルトは true です。

true

タプルログのバッファーを上書きします。

false

タプルログのバッファーを上書きしません。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の順位で指定値が有効になります（1.＞ 2.＞ 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(28) trc.api.bufferCount=API トレースのバッファーの面数

API トレースのバッファーの面数を 3～512 の整数で指定します。デフォルトは 3 です。

(29) trc.api.bufferSize=API トレースのバッファーの最大サイズ

API トレースのバッファーの最大サイズ（単位：キロバイト）を 1～2048000 の整数で指定します。デフォルトは 1024 です。

(30) trc.api.fileCount=API トレースファイルの最大ファイル数

API トレースを出力するファイルの最大ファイル数を 3～512 の整数で指定します。デフォルトは 3 です。

(31) trc.api.fileSize=API トレースファイルの最大サイズ

API トレースを出力するファイルの最大サイズ（単位：メガバイト）を 1～2048 の整数で指定します。デフォルトは 1024 です。

(32) trc.api.ioBufferSize=API トレース I/O バッファーの最大サイズ

API トレースの I/O バッファーの最大サイズ（単位：キロバイト）を 1～2048000 の整数で指定します。デフォルトは 2048 です。

(33) trc.api.outputTrigger= {BUFFER | NONE}

API トレースのファイルへの出力契機を BUFFER または NONE で指定します。大文字，小文字の区別はされません。デフォルトは BUFFER です。

BUFFER
I/O スレッドのバッファーがいっぱいになったら，API トレースをファイルに出力します。

NONE
API トレースをファイルに出力しません。バッファリングもしません。

(34) trc.api.useOverwrite= {true | false}

API トレースのバッファーの枯渇時にバッファーを上書きするかどうかを true または false で指定します。大文字，小文字の区別はされません。デフォルトは true です。

true
API トレースのバッファーを上書きします。

false
API トレースのバッファーを上書きしません。

8.7　クエリグループ用プロパティファイル

ここでは，クエリグループ用プロパティファイルについて，ファイルの詳細と，ファイルに指定するパラメーターの詳細に分けて説明します。

8.7.1　クエリグループ用プロパティファイルの詳細

(1)　記述形式

パラメーターは次の形式で記述します。

＜パラメーター名＞=＜値＞

- 値の後ろに空白やコメントなどの文字列を追加できません。
  追加した場合は値の一部として解釈されます。
- パラメーター名だけを指定して，値を指定していない行は，無視されます。

(2)　ファイル名

＜クエリグループ名＞

クエリグループ名は，英数字（0～9，a～z，A～Z）とアンダーライン（_）で，1～64 文字で指定してください。なお，クエリグループ名の先頭に使用できる文字は半角英文字だけです。

(3)　ファイルの格納先

このファイルは，必ず次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥

(4)　説明

クエリ定義ファイルのパスやクエリグループを実行するときのチューニングパラメーターを設定します。このファイルは，クエリグループごとに必ず作成します。

(5)　指定できるパラメーター

指定できるパラメーターと省略時の仮定値を次の表に示します。なお，パラメーターの詳細については，「8.7.2　クエリグループ用プロパティファイルのパラメーターの詳細」を参照してください。

表 8-5　指定できるパラメーターと省略時の仮定値（クエリグループ用プロパティファイル）

項番	パラメーター名	内容	省略時の仮定値	指定できる値の範囲	再開始時の変更可否 ※1
1	engine.externalStreamFuncVerifyMode	ストリーム間演算の戻り値の型を検証するかどうかを指定します。	true	true または false	×
2	querygroup.cqlFilePath	クエリグループを定義するクエリ定義ファイルのパスを指定します。	なし	絶対パスまたは運用ディレクト	×

項番	パラメーター名	内容	省略時の仮定値	指定できる値の範囲	再開始時の変更可否※1
2	querygroup.cqlFilePath	クエリグループを定義するクエリ定義ファイルのパスを指定します。	なし	リからの相対パス	×
3	querygroup.sleepOnOverStore	SDP サーバが出力ストリームキューに空きがあるかどうかをチェックして空きがなかった場合に, クエリグループの実行をスリープさせる時間を指定します。	system_config.properties で指定した値※2	1～2147483647	○
4	querygroup.sleepOnOverStoreRetryCount	SDP サーバがクエリ実行後のタプルを出力ストリームキューへ投入する前に，出力ストリームキューに空きがあるかどうかをチェックする回数を指定します。	system_config.properties で指定した値※2	0～2147483647	○
5	stream.filterCondition	タイムスタンプ調整機能でタプルのフィルタリングをする場合に, フィルタリングの条件演算式を指定します。	なし	「8.7.2(5) stream.filterCondition=条件演算式」を参照してください。	○
6	stream.filterMode	タイムスタンプ調整機能でタプルのフィルタリングをするかどうかを指定します。 • unuse：タプルのフィルタリングをしません。 • condition：タプルのフィルタリングをします。	unuse	unuse または condition	○
7	stream.freeInputQueueSizeThreshold	入力ストリームキューで使用する要素数の上限値に対する空きサイズのしきい値（単位：%）を指定します。	system_config.properties で指定した値※2	1～99 の整数	○
8	stream.freeInputQueueSizeThresholdOutputMessage	SDP サーバのメッセージログに, 警告メッセージを出力するかどうかを指定します。 • true：警告メッセージを出力します。 • false：警告メッセージを出力しません。	system_config.properties で指定した値※2	true または false	○
9	stream.maxKeepTupleCount	タイムスタンプ調整機能で保留できるタプル数の上限値を指定します。	system_config.properties で指定した値※2	1～1048576 の整数	○
10	stream.propertyFiles	ストリームごとにプロパティを設定する場合に, ストリーム用プロパティファイルのファイル名を指定します。	なし	－	○

項番	パラメーター名	内容	省略時の仮定値	指定できる値の範囲	再開始時の変更可否※1
11	stream.timestampAccuracy	タイムスタンプ調整機能の時刻単位と時刻調整範囲を指定します。	system_config.properties で指定した値※2	「8.7.2(11) stream.timestampAccuracy={{sec | msec | usec} ,時刻調整範囲 | unuse}」を参照してください。	○
12	stream.timestampMode	タプルに時刻を付与するためのタイムスタンプモードを指定します。 • Server：サーバモードを使用します。 • DataSource：データソースモードを使用します。	system_config.properties で指定した値※2	Server または DataSource	×
13	stream.timestampPosition	タプル内の時刻データ列名を指定します。	system_config.properties で指定した値※2	−	○
14	stream.tupleLogMode	sdptplput コマンドを実行するかどうかを指定します。 • true：sdptplput コマンドを実行できます。 • false：sdptplput コマンドを実行できません。	system_config.properties で指定した値※2	true または false	×
15	tpl.backupFileCount	タプルログファイルのバックアップを残す最大世代数を指定します。	system_config.properties で指定した値※2	0～10 の整数	○
16	tpl.bufferCount	タプルログのバッファーの面数を指定します。	system_config.properties で指定した値※2	3～512 の整数	○
17	tpl.bufferSize	タプルログのバッファーの最大サイズ（単位：キロバイト）を指定します。	system_config.properties で指定した値※2	1～2048000 の整数	○
18	tpl.fileCount	タプルログファイルの最大ファイル数を指定します。	system_config.properties で指定した値※2	3～512 の整数	○
19	tpl.fileSize	タプルログファイルの最大サイズ（単位：メガバイト）を指定します。	system_config.properties で指定した値※2	1～2048 の整数	○

項番	パラメーター名	内容	省略時の仮定値	指定できる値の範囲	再開始時の変更可否※1
20	tpl.outputLevel	タプルログの出力レベルを指定します。 • 1：ストリームキューへ格納するタプルについて，タプルログを出力します。 • 2：時刻の逆転によって破棄するタプルについて，タプルログを出力します。 • 3：レベル 1 とレベル 2 のタプルについて，タプルログを出力します。	system_config.properties で指定した値※2	1～3 の整数	○
21	tpl.outputTrigger	タプルログのファイルへの出力契機を指定します。 • BUFFER：対象のストリームキューで現在のタプルログを取得しているバッファーがいっぱいになったら，タプルログをファイルへ出力します。 • NONE：タプルログをファイルに出力しません。バッファリングもしません。	system_config.properties で指定した値※2	BUFFER または NONE	○
22	tpl.useOverwrite	タプルログのバッファーの枯渇時にバッファーを上書きするかどうかを指定します。 • true：タプルログのバッファーを上書きします。 • false：タプルログのバッファーを上書きしません。	system_config.properties で指定した値※2	true または false	○

（凡例）

○：変更できます。

×：変更できません。

－：該当しません。

注※1

sdpcqlstart コマンドで-reload オプションを指定してクエリグループを開始する際に，パラメーターの指定内容を変更できるかどうかを示します。

注※2

システムコンフィグプロパティファイル（system_config.properties）で値が指定されていない場合は，システムコンフィグプロパティファイルのデフォルト値が適用されます。

(6)　注意事項

• クエリグループの登録後にクエリグループ用プロパティファイルの内容を変更する場合，「(5)　指定できるパラメーター」の一覧で「再開始時の変更可否」が「○」になっているパラメーターの値を変更す

る場合，クエリグループを削除する必要はありません。「×」になっているパラメーターの値を変更する場合，クエリグループを削除する必要があります。変更の手順については，「5.3.1　プロパティファイルの設定値の変更」を参照してください。

- sdpcqlstart コマンドで-reload オプションを指定してクエリグループを再開始した場合に，「(5)　指定できるパラメーター」の一覧で「再開始時の変更可否」が「×」になっているパラメーターの値を変更していたときは，変更した内容を無視して処理を続行します。

8.7.2　クエリグループ用プロパティファイルのパラメーターの詳細

ここでは，「8.7.1(5)　指定できるパラメーター」で示したクエリグループ用プロパティファイルのパラメーターの詳細について説明します。

(1)　engine.externalStreamFuncVerifyMode=ストリーム間演算の戻り値の型の検証

ストリーム間演算の戻り値の型を検証するかどうかを true または false で指定します。大文字，小文字の区別はされません。デフォルトは true です。

true
　検証します。

false
　検証しません。

false を指定することで，ストリーム間演算関数を頻繁に呼び出すシステムで性能が改善する場合があります。ただし，検証をしない場合は，トラブルシュートが困難になるので注意してください。

このパラメーターは，システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.）。

1. クエリグループ用プロパティファイル
2. システムコンフィグプロパティファイル

(2)　querygroup.cqlFilePath=クエリ定義ファイルのパス

クエリグループを定義するクエリ定義ファイルのパスを絶対パスまたは運用ディレクトリからの相対パスで指定します。このパラメーターの指定がない場合，クエリグループを登録できません。

(3)　querygroup.sleepOnOverStore=クエリグループの実行をスリープさせる時間

SDP サーバが出力ストリームキューに空きがあるかどうかをチェックして空きがなかった場合に，クエリグループの実行をスリープさせる時間（単位：ミリ秒）を 1～2147483647 の整数で指定します。

このパラメーターの指定は，querygroup.sleepOnOverStoreRetryCount パラメーターで 1 以上を指定した場合に有効です。

このパラメーターによってスリープしたクエリグループに対しては，次のことが実行できます。

- 入力アダプターによる入力ストリームキューへのタプルの投入
- 出力アダプターによる出力ストリームキューからのタプルの取得

このパラメーターは，システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.)。

1. クエリグループ用プロパティファイル
2. システムコンフィグプロパティファイル

(4)　querygroup.sleepOnOverStoreRetryCount=出力ストリームキューの空きをチェックする回数

SDP サーバがクエリ実行後のタプルを出力ストリームキューへ投入する前に，出力ストリームキューに空きがあるかどうかをチェックする回数を 0～2147483647 の整数で指定します。

0 を指定した場合，SDP サーバは，出力ストリームキューに空きがあるかをチェックしないで，クエリ実行後のタプルを出力ストリームキューへ投入します。

1 以上を指定した場合，SDP サーバは，指定された回数だけ次の処理を実行します。

1. 出力ストリームキューに空きがあるかどうかをチェックします。
2. 出力ストリームキューに空きがなければ，出力ストリームキューのあるクエリグループの実行をスリープさせます。スリープさせる時間は，querygroup.sleepOnOverStore パラメーターで指定します。
   スリープ後，このパラメーターで指定した回数分の処理を実行していない場合は，1.の処理に戻ります。
3. 出力ストリームキューに空きがあれば，出力ストリームキューへクエリ実行後のタプルを投入します。

このパラメーターは，システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.)。

1. クエリグループ用プロパティファイル
2. システムコンフィグプロパティファイル

(5)　stream.filterCondition=条件演算式

タイムスタンプ調整機能でタプルのフィルタリングをする場合に，フィルタリングの条件演算式を指定します。タプルのフィルタリングでは，条件演算式の列名に指定した列の内容と，定数に指定した内容に対して条件演算を行います。条件演算の結果，一致する内容を持つタプルはタイムスタンプ調整機能で保留し，一致しないタプルは破棄します。

条件演算式の記述方法については，「8.7.3　条件演算式の記述規則」を参照してください。

stream.filterMode パラメーターで condition を指定した場合，このパラメーターを必ず指定してください。stream.filterMode パラメーターで unuse を指定した場合，このパラメーターの指定は無視されますが，パラメーターの形式チェックは実施されます。

このパラメーターは，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります(1.> 2.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル

(6)　stream.filterMode=｛unuse ｜ condition｝

タイムスタンプ調整機能でタプルのフィルタリングをするかどうかを unuse または condition で指定します。大文字，小文字の区別はされません。デフォルトは unuse です。

unuse

タプルのフィルタリングをしません。

condition

タプルのフィルタリングをします。

condition を指定した場合は，stream.filterCondition パラメーターを必ず指定してください。

stream.timestampAccuracy パラメーターで unuse を指定した場合，このパラメーターの指定は無視されますが，パラメーターの形式チェックは実施されます。

このパラメーターは，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.＞ 2.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル

(7)　stream.freeInputQueueSizeThreshold=入力ストリームキューの最大サイズに対する空きサイズのしきい値

入力ストリームキューで使用する要素数の上限値（システムコンフィグプロパティファイルの engine.maxQueueSize パラメーターの指定値）に対する空きサイズのしきい値（単位：%）を 1～99 の整数で指定します。

次に示す条件を満たした場合に，put(StreamTuple tuple)メソッドまたは put(ArrayList<StreamTuple> tuple_list)メソッドから SDPClientFreeInputQueueSizeThresholdOverException の例外がスローされます。このとき，入力ストリームキューへのタプルの投入は，成功しています。

このパラメーターの指定値 ≧（（入力ストリームキューの空きサイズ÷入力ストリームキューの最大サイズ）×100）

このパラメーターを省略した場合は，しきい値のチェックによる例外は発生しません。

sdptplput コマンドで入力ストリームキューにタプルを投入する場合，このパラメーターの指定は無効となります。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.＞ 2.＞ 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(8)　stream.freeInputQueueSizeThresholdOutputMessage=｛true ｜ false｝

SDP サーバのメッセージログに，警告メッセージ（メッセージ KFSP42032-W）を出力するかどうかを true または false で指定します。大文字，小文字の区別はされません。

true

警告メッセージを出力します。

false

警告メッセージを出力しません。

このパラメーターの指定は，stream.freeInputQueueSizeThreshold パラメーターが指定された場合だけ有効です。

なお，警告メッセージが出力されるのは，このパラメーターで true を指定し，次の条件を満たした場合です。

stream.freeInputQueueSizeThresholdパラメーターの指定値 ≧（（入力ストリームキューの空きサイズ÷入力ストリームキューの最大サイズ）×100）

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(9)　stream.maxKeepTupleCount=タイムスタンプ調整機能で保留できるタプル数の上限値

タイムスタンプ調整機能で保留できるタプル数の上限値を 1～1048576 の整数で指定します。

このパラメーターで指定したタプル数の上限値は，個々の入力ストリームのタイムスタンプ調整機能に対する上限値となります。

タプル数がこのパラメーターで指定した上限値を超えると，クエリグループが閉塞されます。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(10)　stream.propertyFiles=ストリーム用プロパティファイル名

ストリームごとにプロパティを設定する場合に，ストリーム用プロパティファイルのファイル名を指定します。

このパラメーターで指定するファイル名には，パス名を含めないでください。パス名を含めて指定した場合，パス名を含めて一つのファイル名として認識されます。

複数のファイル名を指定する場合は，半角コンマ（,）で区切って指定します。次の場合はエラーになります。

- 半角コンマの前後にファイル名が指定されていない場合
- 区切り文字の前後に空白，タブなどを記述した場合

- 同じストリームのストリーム用プロパティファイルが複数個指定された場合

なお，クエリグループに存在しないストリームのストリーム用プロパティファイルが指定された場合，指定されたファイルは解析対象となりますが，ファイルの定義内容は無視されます。

(11)　stream.timestampAccuracy= {{sec | msec | usec} ,時刻調整範囲 | unuse}

タイムスタンプ調整機能の時刻単位と時刻調整範囲を指定します。大文字，小文字の区別はされません。

stream.timestampMode パラメーターで DataSource を指定した場合，このパラメーターを必ず指定してください。stream.timestampMode パラメーターで Server を指定した場合，このパラメーターの指定は無視されますが，パラメーターの形式チェックは実施されます。

{sec | msec | usec} ,時刻調整範囲

時刻単位と時刻調整範囲を指定します。時刻単位（sec，msec，または usec）と時刻調整範囲を区切る半角コンマ（,）の前後には，空白やタブなどを記述しないでください。記述した場合はエラーとなります。時刻単位に sec，msec，または usec のどれかを指定し，時刻調整範囲に 0 を指定した場合は基準時刻だけが時刻調整範囲となります。

それぞれの値の意味を次に示します。

sec

時刻単位として秒を使用することを指定します。

msec

時刻単位としてミリ秒を使用することを指定します。

usec

時刻単位としてマイクロ秒を使用することを指定します。

時刻調整範囲

タイムスタンプ調整での時刻の調整範囲を整数で指定します。時刻単位に指定した単位によって指定範囲が異なります。時刻単位ごとの指定範囲を次の表に示します。

時刻単位	指定範囲
sec（秒）	0～59 の整数
msec（ミリ秒）	0～999 の整数
usec（マイクロ秒）	0～999 の整数

unuse

時刻調整を行いません。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(12) stream.timestampMode= {Server | DataSource}

タプルに時刻を付与するためのタイムスタンプモードを Server または DataSource で指定します。大文字，小文字の区別はされません。

Server

サーバモードを使用します。

DataSource

データソースモードを使用します。

DataSource を指定した場合は，stream.timestampAccuracy パラメーター，および stream.timestampPosition パラメーターを必ず指定してください。

このパラメーターは，システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.）。

1. クエリグループ用プロパティファイル
2. システムコンフィグプロパティファイル

(13) stream.timestampPosition=時刻データ列名

タプル内の時刻データ列名を指定します。大文字，小文字の区別はされません。

stream.timestampMode パラメーターで DataSource を指定した場合，このパラメーターを必ず指定してください。stream.timestampMode パラメーターで Server を指定した場合，このパラメーターの指定は無視されますが，パラメーターの形式チェックは実施されます。

時刻データとして指定できるデータ型は，TIMESTAMP 型だけです。

時刻データの範囲は，GMT（グリニッジ標準時）の 1970/01/01 00:00:00.000000000 から 2261/12/31 23:59:59.999999999 までになります。それ以外の時刻を指定した場合は，ストリームデータ送信時に例外が発生します。日本標準時は GMT より 9 時間あとであることに注意してください。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(14) stream.tupleLogMode= {true | false}

sdptplput コマンドを実行するかどうかを true または false で指定します。大文字，小文字の区別はされません。

true

sdptplput コマンドを実行できます。

false

sdptplput コマンドを実行できません。

stream.timestampMode パラメーターで Server を，このパラメーターで true を指定した場合は，StreamInput インタフェースの put メソッドでストリームデータを送信すると，SDPClientException の例外が発生します。

このパラメーターは，システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.)。

1. クエリグループ用プロパティファイル
2. システムコンフィグプロパティファイル

(15)　tpl.backupFileCount=タプルログファイルのバックアップを残す最大世代数

タプルログファイルのバックアップを残す最大世代数を 0～10 の整数で指定します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(16)　tpl.bufferCount=タプルログのバッファーの面数

タプルログのバッファーの面数を 3～512 の整数で指定します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(17)　tpl.bufferSize=タプルログのバッファーの最大サイズ

タプルログのバッファーの最大サイズ（単位：キロバイト）を 1～2048000 の整数で指定します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(18)　tpl.fileCount=タプルログファイルの最大ファイル数

タプルログファイルの最大ファイル数を 3～512 の整数で指定します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(19)　tpl.fileSize=タプルログファイルの最大サイズ

タプルログファイルの最大サイズ（単位：メガバイト）を 1～2048 の整数で指定します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(20)　tpl.outputLevel=タプルログの出力レベル

タプルログの出力レベルを 1～3 の整数で指定します。

1
　ストリームキューへ格納するタプルについて，タプルログを出力します。

2
　時刻の逆転によって破棄するタプルについて，タプルログを出力します。

3
　レベル 1 とレベル 2 のタプルについて，タプルログを出力します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(21)　tpl.outputTrigger= {BUFFER | NONE}

タプルログのファイルへの出力契機を BUFFER または NONE で指定します。大文字，小文字の区別はされません。

BUFFER
　対象のストリームキューで現在のタプルログを取得しているバッファーがいっぱいになったら，タプルログをファイルへ出力します。

NONE
　タプルログをファイルに出力しません。バッファリングもしません。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(22)　tpl.useOverwrite= {true | false}

タプルログのバッファーの枯渇時にバッファーを上書きするかどうかを true または false で指定します。大文字，小文字の区別はされません。

true

タプルログのバッファーを上書きします。

false

タプルログのバッファーを上書きしません。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の順位で指定値が有効になります（1.> 2.> 3.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

8.7.3　条件演算式の記述規則

ここでは，stream.filterCondition パラメーターで指定する条件演算式の記述規則について説明します。

(1)　条件演算式の形式

条件演算式は，次の形式で記述します。

```
'('列名 比較演算子 定数')'[ {AND | OR} '('列名 比較演算子 定数')'
                          {AND | OR} '('列名 比較演算子 定数')'…]
```

複数の比較演算を組み合わせる場合は，演算式を AND または OR でつないで指定します。指定できる演算式の数は 1～10 個です。

条件演算式に指定できる要素について説明します。

列名

比較演算対象の列名を指定します。列名はストリームの定義（register stream 句）のスキーマ指定文字列で指定した列名を指定します。

比較演算子

条件の判定をするための演算子を指定します。指定できる比較演算子の種類およびその意味を次の表に示します。

比較演算子	比較演算子の使用例	使用例の意味
<=	A <= B	A は B 以下

比較演算子	比較演算子の使用例	使用例の意味
>=	A >= B	A は B 以上
<	A < B	A は B より小さい
>	A > B	A は B より大きい
=	A = B	A は B と等しい
!=	A != B	A は B と等しくない

定数

比較演算を実施する対象の値を整定数または文字列定数で指定します。

- 整定数

  比較演算を数値で実施する場合に指定します。数値として指定できる範囲は，-9223372036854775808～9223372036854775807 の整数になります。

- 文字列定数

  比較演算を文字列で実施する場合に指定します。文字列を指定する場合はアポストロフィ（'）で囲んで記述してください。文字列として指定できるのは半角文字，および全角文字です。指定できる文字数は，100 文字以内です。

(2)　条件演算式の記述例

条件演算式の記述例を次に示します。

```
stream.filterCondition=(xxx='abc')AND(zzz>18)AND(zzz<60)
```

この例では，列名 xxx が abc で，列名 zzz が 18 より大きく 60 より小さいタプルだけを保留します。

(3)　条件演算式の注意事項

- 演算式内に AND と OR を混在させることはできません。
- AND と OR，および列名で，大文字，小文字の区別はされません。
- 列名に指定したデータ型が文字データ（CHAR，VARCHAR）の場合，比較演算子に「=」と「!=」以外は指定できません。
- 整定数を指定する場合，列名に指定できるデータ型は CQL で使用できる次のデータ型になります。
  - INT
  - SMALLINT
  - TINYINT
  - BIGINT
  - DEC
  - NUMERIC
  - REAL
  - FLOAT
  - DOUBLE

  CQL で使用できるデータ型については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

- 整定数を使用する場合，列名に指定したデータ型と整定数に指定した数値の範囲はチェックされません。
- 文字列定数では，文字列をアポストロフィ（'）で囲んで記述しますが，アポストロフィ（'）を文字列として使用したい場合は，1 個のアポストロフィを表すために 2 個のアポストロフィを記述してください。例えば，abc'abc と指定したい場合には，'abc''abc' と記述してください。
- 文字列定数に指定した文字列より大きい文字列が，該当の列に設定されていた場合は，先頭から指定した文字数までを演算の対象とします。例えば，文字列定数に指定した文字列が 'abc'（文字数が 3）で，該当の列に設定されていた文字列が 'abcde' の場合，先頭からの 'abc' までが演算の対象となることから，文字列定数 'abc' と合致します。
- 文字列定数では，空文字（文字列を表す先頭と最終のアポストロフィ（'）の間に何も指定しない）の指定はできません。条件式を囲む括弧，比較演算子，AND，および OR の前後には，空白またはタブを記述できますが，空白またはタブは無視されます。ただし，最後の演算式の閉じ括弧の後ろには，空白およびタブは記述できません。
- 文字列定数を指定する場合, 列名に指定できるデータ型は CQL で使用できる次のデータ型になります。
  - CHAR
  - VARCHAR

  CQL で使用できるデータ型については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

8.8　ストリーム用プロパティファイル

ここでは，ストリーム用プロパティファイルについて，ファイルの詳細と，ファイルに指定するパラメーターの詳細に分けて説明します。

8.8.1　ストリーム用プロパティファイルの詳細

(1)　記述形式

パラメーターは次の形式で記述します。

＜パラメーター名＞=＜値＞

- 値の後ろに空白やコメントなどの文字列を追加できません。
  追加した場合は値の一部として解釈されます。
- パラメーター名だけを指定して，値を指定していない行は，無視されます。

(2)　ファイル名

ファイル名は任意です。このファイル名は，クエリグループ用プロパティファイルの stream.propertyFiles パラメーターで指定します。

(3)　ファイルの格納先

このファイルは，必ず次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥

(4)　説明

ストリーム単位でチューニングパラメーターを設定します。このファイルは，クエリグループ内のストリーム単位でチューニングパラメーターを設定したい場合だけ，ストリームごとに作成します。

(5)　指定できるパラメーター

指定できるパラメーターと省略時の仮定値を次の表に示します。なお，パラメーターの詳細については，「8.8.2　ストリーム用プロパティファイルのパラメーターの詳細」を参照してください。

表 8-6　指定できるパラメーターと省略時の仮定値（ストリーム用プロパティファイル）

項番	パラメーター名	内容	省略時の仮定値	指定できる値の範囲	再開始時の変更可否※1
1	stream.filterCondition	タイムスタンプ調整機能でタプルのフィルタリングをする場合に，フィルタリングの条件演算式を指定します。	クエリグループ用プロパティファイルで指定した値※2	－	○
2	stream.filterMode	タイムスタンプ調整機能でタプルのフィルタリングをするかどうかを指定します。 unuse：タプルのフィルタリングをしません。	クエリグループ用プロパティファイルで指定した値※2	unuse または condition	○

項番	パラメーター名	内容	省略時の仮定値	指定できる値の範囲	再開始時の変更可否※1
2	stream.filterMode	condition：タプルのフィルタリングをします。	クエリグループ用プロパティファイルで指定した値※2	unuse または condition	○
3	stream.freeInputQueueSizeThreshold	入力ストリームキューで使用する要素数の上限値に対する空きサイズのしきい値（単位：%）を指定します。	クエリグループ用プロパティファイルで指定した値※2	1～99 の整数	○
4	stream.freeInputQueueSizeThresholdOutputMessage	SDP サーバのメッセージログに，警告メッセージを出力するかどうかを指定します。 true：警告メッセージを出力します。 false：警告メッセージを出力しません。	クエリグループ用プロパティファイルで指定した値※2	true または false	○
5	stream.maxKeepTupleCount	タイムスタンプ調整機能で保留できるタプル数の上限値を指定します。	クエリグループ用プロパティファイルで指定した値※2	1～1048576 の整数	○
6	stream.streamName	ストリームの名称を指定します。	なし	－	○
7	stream.timestampAccuracy	タイムスタンプ調整機能の時刻単位と時刻調整範囲を指定します。	クエリグループ用プロパティファイルで指定した値※2	「8.8.2(7) stream.timestampAccuracy={{sec ｜ msec ｜ usec} ,時刻調整範囲 ｜ unuse}」を参照してください。	○
8	stream.timestampPosition	タプル内の時刻データ列名を指定します。	クエリグループ用プロパティファイルで指定した値※2	「8.8.2(8) stream.timestampPosition=時刻データ列名」を参照してください。	○
9	tpl.backupFileCount	タプルログファイルのバックアップを残す最大世代数を指定します。	クエリグループ用プロパティファイルで指定した値※2	0～10 の整数	○
10	tpl.bufferCount	タプルログのバッファーの面数を指定します。	クエリグループ用プロパティファイルで指定した値※2	3～512 の整数	○
11	tpl.bufferSize	タプルログのバッファーの最大サイズ（単位：キロバイト）を指定します。	クエリグループ用プロパティファイルで指定した値※2	1～2048000 の整数	○

項番	パラメーター名	内容	省略時の仮定値	指定できる値の範囲	再開始時の変更可否※1
12	tpl.fileCount	タプルログファイルの最大ファイル数を指定します。	クエリグループ用プロパティファイルで指定した値※2	3～512 の整数	○
13	tpl.fileSize	タプルログファイルの最大サイズ（単位：メガバイト）を指定します。	クエリグループ用プロパティファイルで指定した値※2	1～2048 の整数	○
14	tpl.outputLevel	タプルログの出力レベルを指定します。 • 1：ストリームキューへ格納するタプルについて，タプルログを出力します。 • 2：時刻の逆転によって破棄するタプルについて，タプルログを出力します。 • 3：レベル 1 とレベル 2 のタプルについて，タプルログを出力します。	クエリグループ用プロパティファイルで指定した値※2	1～3 の整数	○
15	tpl.outputTrigger	タプルログのファイルへの出力契機を指定します。 BUFFER：対象のストリームキューで現在のタプルログを取得しているバッファーがいっぱいになったら，タプルログをファイルへ出力します。 NONE：タプルログをファイルに出力しません。バッファリングもしません。	クエリグループ用プロパティファイルで指定した値※2	BUFFER または NONE	○
16	tpl.useOverwrite	タプルログのバッファーの枯渇時にバッファーを上書きするかどうかを指定します。 • true：タプルログのバッファーを上書きします。 • false：タプルログのバッファーを上書きしません。	クエリグループ用プロパティファイルで指定した値※2	true または false	○

（凡例）

－：該当しません。

○：変更できます。

注※1

sdpcqlstart コマンドで-reload オプションを指定してクエリグループを開始する際に，パラメーターの指定内容を変更できるかどうかを示します。

注※2

クエリグループ用プロパティファイルで値が指定されていない場合は，クエリグループ用プロパティファイルのデフォルト値が適用されます。

(6)　注意事項

- クエリグループの登録後にストリーム用プロパティファイルの内容を変更する場合，「(5)　指定できるパラメーター」の一覧で「再開始時の変更可否」が「○」になっているパラメーターの値を変更する場合，クエリグループを削除する必要はありません。変更の手順については，「5.3.1　プロパティファイルの設定値の変更」を参照してください。

8.8.2　ストリーム用プロパティファイルのパラメーターの詳細

ここでは，「8.8.1(5)　指定できるパラメーター」で示したストリーム用プロパティファイルのパラメーターの詳細について説明します。

(1)　stream.filterCondition=条件演算式

タイムスタンプ調整機能でタプルのフィルタリングをする場合に，フィルタリングの条件演算式を指定します。タプルのフィルタリングでは，条件演算式の列名に指定した列の内容と，定数に指定した内容に対して条件演算を行います。条件演算の結果，一致する内容を持つタプルはタイムスタンプ調整機能で保留し，一致しないタプルは破棄します。

条件演算式の記述方法については，「8.7.3　条件演算式の記述規則」を参照してください。

stream.filterMode パラメーターで condition を指定した場合，このパラメーターを必ず指定してください。stream.filterMode パラメーターで unuse を指定した場合，このパラメーターの指定は無視されますが，パラメーターの形式チェックは実施されます。

このパラメーターは，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります(1.> 2.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル

(2)　stream.filterMode= {unuse | condition}

タイムスタンプ調整機能でタプルのフィルタリングをするかどうかを unuse または condition で指定します。大文字，小文字の区別はされません。

unuse
　タプルのフィルタリングをしません。

condition
　タプルのフィルタリングをします。
　condition を指定した場合は，stream.filterCondition パラメーターを必ず指定してください。

stream.timestampAccuracy パラメーターで unuse を指定した場合，このパラメーターの指定は無視されますが，パラメーターの形式チェックは実施されます。

このパラメーターは，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります(1.> 2.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル

(3)　stream.freeInputQueueSizeThreshold=入力ストリームキューの最大サイズに対する空きサイズのしきい値

入力ストリームキューで使用する要素数の上限値（システムコンフィグプロパティファイルの engine.maxQueueSize パラメーターの指定値）に対する空きサイズのしきい値（単位：%）を 1～99 の整数で指定します。

次に示す条件を満たした場合に，put(StreamTuple tuple)メソッドまたは put(ArrayList<StreamTuple> tuple_list)メソッドから SDPClientFreeInputQueueSizeThresholdOverException の例外がスローされます。このとき，入力ストリームキューへのタプルの投入は，成功しています。

このパラメーターの指定値 ≧（（入力ストリームキューの空きサイズ÷入力ストリームキューの最大サイズ）×100）

このパラメーターを省略した場合は，しきい値のチェックによる例外は発生しません。

sdptplput コマンドで入力ストリームキューにタプルを投入する場合，このパラメーターの指定は無効となります。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(4)　stream.freeInputQueueSizeThresholdOutputMessage= {true | false}

SDP サーバのメッセージログに，警告メッセージ（メッセージ KFSP42032-W）を出力するかどうかを true または false で指定します。大文字，小文字の区別はされません。

true

警告メッセージを出力します。

false

警告メッセージを出力しません。

このパラメーターの指定は，stream.freeInputQueueSizeThreshold パラメーターが指定された場合だけ有効です。

なお，警告メッセージが出力されるのは，このパラメーターで true を指定し，次の条件を満たした場合です。

stream.freeInputQueueSizeThresholdパラメーターの指定値 ≧（（入力ストリームキューの空きサイズ÷入力ストリームキューの最大サイズ）×100）

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(5)　stream.maxKeepTupleCount=タイムスタンプ調整機能で保留できるタプル数の上限値

タイムスタンプ調整機能で保留できるタプル数の上限値を 1～1048576 の整数で指定します。

このパラメーターで指定したタプル数の上限値は，個々の入力ストリームのタイムスタンプ調整機能に対する上限値となります。

タプル数がこのパラメーターで指定した上限値を超えると，クエリグループが閉塞されます。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.＞ 2.＞ 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(6)　stream.streamName=ストリームの名称

このプロパティファイルの定義内容を適用するストリームの名称を指定します。

指定したストリームの名称は，すべて大文字として扱われます。このパラメーターの指定がない場合は，エラーになります。

(7)　stream.timestampAccuracy= {{sec | msec | usec} ,時刻調整範囲 | unuse}

タイムスタンプ調整機能の時刻単位と時刻調整範囲を指定します。大文字，小文字の区別はされません。

{sec | msec | usec} ,時刻調整範囲

時刻単位と時刻調整範囲を指定します。時刻単位（sec，msec，または usec）と時刻調整範囲を区切る半角コンマ（,）の前後には，空白やタブなどを記述しないでください。記述した場合はエラーとなります。時刻単位に sec，msec，または usec のどれかを指定し，時刻調整範囲に 0 を指定した場合は基準時刻だけが時刻調整範囲となります。

それぞれの値の意味を次に示します。

sec

時刻単位として秒を使用することを指定します。

msec

時刻単位としてミリ秒を使用することを指定します。

usec

時刻単位としてマイクロ秒を使用することを指定します。

時刻調整範囲

タイムスタンプ調整での時刻の調整範囲を整数で指定します。時刻単位に指定した単位によって指定範囲が異なります。時刻単位ごとの指定範囲を次の表に示します。

時刻単位	指定範囲
sec（秒）	0～59 の整数
msec（ミリ秒）	0～999 の整数

時刻単位	指定範囲
usec（マイクロ秒）	0～999 の整数

unuse

時刻調整を行いません。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(8)　stream.timestampPosition=時刻データ列名

タプル内の時刻データ列名を指定します。大文字，小文字の区別はされません。

時刻データとして指定できるデータ型は，TIMESTAMP 型だけです。

時刻データの範囲は，GMT（グリニッジ標準時）の 1970/01/01 00:00:00.000000000 から 2261/12/31 23:59:59.999999999 までになります。それ以外の時刻を指定した場合は，ストリームデータ送信時に例外が発生します。日本標準時は GMT より 9 時間あとであることに注意してください。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(9)　tpl.backupFileCount=タプルログファイルのバックアップを残す最大世代数

タプルログファイルのバックアップを残す最大世代数を 0～10 の整数で指定します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(10)　tpl.bufferCount=タプルログのバッファーの面数

タプルログのバッファーの面数を 3～512 の整数で指定します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(11)　tpl.bufferSize=タプルログのバッファーの最大サイズ

タプルログのバッファーの最大サイズ（単位：キロバイト）を 1～2048000 の整数で指定します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(12)　tpl.fileCount=タプルログファイルの最大ファイル数

タプルログファイルの最大ファイル数を 3～512 の整数で指定します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(13)　tpl.fileSize=タプルログファイルの最大サイズ

タプルログファイルの最大サイズ（単位：メガバイト）を 1～2048 の整数で指定します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.)。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(14)　tpl.outputLevel=タプルログの出力レベル

タプルログの出力レベルを 1～3 の整数で指定します。

1
　ストリームキューへ格納するタプルについて，タプルログを出力します。

2
　時刻の逆転によって破棄するタプルについて，タプルログを出力します。

3
　レベル 1 とレベル 2 のタプルについて，タプルログを出力します。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(15) tpl.outputTrigger= {BUFFER | NONE}

タプルログのファイルへの出力契機を BUFFER または NONE で指定します。大文字，小文字の区別はされません。

BUFFER
: 対象のストリームキューで現在のタプルログを取得しているバッファーがいっぱいになったら，タプルログをファイルへ出力します。

NONE
: タプルログをファイルに出力しません。バッファリングもしません。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の優先順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

(16) tpl.useOverwrite= {true | false}

タプルログのバッファーの枯渇時にバッファーを上書きするかどうかを true または false で指定します。大文字，小文字の区別はされません。

true
: タプルログのバッファーを上書きします。

false
: タプルログのバッファーを上書きしません。

このパラメーターは，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，およびストリーム用プロパティファイルで指定できます。指定が重複した場合，または指定を省略した場合，次の順位で指定値が有効になります（1.> 2.> 3.）。

1. ストリーム用プロパティファイル
2. クエリグループ用プロパティファイル
3. システムコンフィグプロパティファイル

8.9　インプロセス連携用プロパティファイル

(1)　記述形式

パラメーターは次の形式で記述します。

＜パラメーター名＞=＜値＞

- 値の後ろに空白やコメントなどの文字列を追加できません。
  追加した場合は値の一部として解釈されます。
- パラメーター名だけを指定して，値を指定していない行は，無視されます。

(2)　ファイル名

user_app.＜アダプターグループ名またはアダプター名＞.properties

標準提供アダプターの場合はアダプターグループ名を，カスタムアダプターの場合はアダプター名を，英数字（0～9，a～z，A～Z）とアンダーライン（_）で指定します。指定できる文字数は，1～32 文字です。なお，インプロセス連携をするアダプターグループ名またはアダプター名の先頭に使用できる文字は半角英文字だけです。

(3)　ファイルの格納先

このファイルは，必ず次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥

(4)　説明

インプロセス連携をするアダプターのクラス名や jar ファイルのパスを設定します。インプロセス連携をする場合だけ，標準提供アダプターはアダプターグループごと，カスタムアダプターはアダプターごとに作成します。

(5)　指定できるパラメーター

指定できるパラメーターとデフォルト値を次の表に示します。

表 8-7　指定できるパラメーターとデフォルト値（インプロセス連携用プロパティファイル）

項番	パラメーター名	内容	デフォルト値
1	user_app.classname	インプロセスで連携するアダプターグループまたはアダプターのメインクラス名を指定します。※1 このクラスは，システムが提供する interface クラスを実装したクラスとなります。 インプロセスで連携する場合，このパラメーターの指定がないと，エラーとなります。	なし
2	user_app.classpath_dir	user_app.classname パラメーターで指定されたクラスに対するクラスファイルの格納ディレクトリのパス，または jar ファイルのパスを指定します。※1 ※2 パスは，絶対パスまたは運用ディレクトリからの相対パスで指定します。 このパラメーターに指定できるパスは一つだけです。	なし

項番	パラメーター名	内容	デフォルト値
2	user_app.classpath_dir	インプロセスで連携する場合，このパラメーターの指定がないと，エラーとなります。	なし

注※1

標準提供アダプターの場合は，指定値は固定です。次の値を指定してください。

user_app.classname=jp.co.Hitachi.soft.sdp.adaptor.AdaptorManager

user_app.classpath_dir=＜インストールディレクトリ＞¥¥lib¥¥sdp.jar

注※2

SDP サーバ用 JavaVM オプションファイルの SDP_CLASS_PATH パラメーターで指定したパスに jar ファイルが存在する場合，SDP_CLASS_PATH パラメーターで指定したパスにある jar ファイルが優先されます。

ユーザーが作成したインプロセス連携アダプターから，このパラメーターに指定したクラスパス以外のパスを参照する場合は，SDP サーバ用 JavaVM オプションファイルの SDP_CLASS_PATH パラメーターにパスを指定してください。なお，SDP サーバ用 JavaVM オプションファイルの SDP_CLASS_PATH パラメーターに指定したパスは，SDP サーバの起動時だけ読み込まれます。そのため，SDP サーバの起動後にパスの値や jar ファイルの内容を変更しても反映されません。

8.10　ログファイル出力用プロパティファイル（logger.properties）

(1)　記述形式

パラメーターは次の形式で記述します。

＜パラメーター名＞=＜値＞

- 値の後ろに空白やコメントなどの文字列を追加できません。
  追加した場合は値の一部として解釈されます。
- パラメーター名だけを指定して，値を指定していない行は，無視されます。

(2)　ファイル名

logger.properties

(3)　ファイルの格納先

このファイルは，必ず次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥

(4)　説明

メッセージログやトレースログの面数やサイズを設定します。このファイルは，運用ディレクトリごとに作成します。作成しない場合には，デフォルト値で動作します。

(5)　指定できるパラメーター

指定できるパラメーターとデフォルト値を次の表に示します。

表 8-8　指定できるパラメーターとデフォルト値（logger.properties）

項番	パラメーター名	内容	デフォルト値	指定できる値の範囲
1	logger.serverMessageFileCount	メッセージログファイルの最大面数を指定します。	3	2～16 の整数
2	logger.serverMessageMaxFileSize	メッセージログファイルの最大サイズ（単位：バイト）を指定します。	1048576	4096～2147483647 の整数
3	logger.serverTraceFileCount	トレースログファイルの最大面数を指定します。	3	2～16 の整数
4	logger.serverTraceMaxFileSize	トレースログファイルの最大サイズ（単位：バイト）を指定します。	1048576	4096～2147483647 の整数

8.11　JavaVM オプションの一覧

ここでは，日立 JavaVM のオプションについて説明します。日立 JavaVM のオプションは，次のファイルで指定できます。

- SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）
  このファイルへの指定方法，およびこのシステムでのデフォルト値については，「8.4　SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg）」を参照してください。
- RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）
  このファイルへの指定方法，およびこのシステムでのデフォルト値については，「8.5　RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）」を参照してください。

JavaVM オプションの一覧を次の表に示します。なお，ここで説明する JavaVM オプションのデフォルト値は，SDP サーバ用 JavaVM オプションファイル（jvm_options.cfg），または RMI 連携用 JavaVM オプションファイル（jvm_client_options.cfg）で JavaVM オプションを指定していない場合のデフォルト値を示します。

表 8-9　JavaVM オプションの一覧

項番	分類	オプション名	説明
1	JavaVM メモリ空間のサイズや割合指定オプション	-Xms<size>※1	Java ヒープの初期サイズを設定します。
		-Xmx<size>※1	Java ヒープの最大サイズを設定します。
		-XX:NewRatio=<value>	DefNew 領域に対する Tenured 領域の割合を設定します。<value>が 2 の場合は，DefNew 領域と Tenured 領域の割合が，1：2 になります。
		-XX:PermSize=<size>※1	Permanent 領域の初期サイズを設定します。
		-XX:MaxPermSize=<size>※1	Permanent 領域の最大サイズを設定します。
2	拡張スレッドダンプ機能オプション	-XX:[+|-]HitachiThreadDumpToStdout	標準出力にスレッドダンプを出力するかどうかを指定します。 • -XX:+HitachiThreadDumpToStdout：拡張スレッドダンプを標準出力，および日立 JavaVM ログファイルに出力します。 • -XX:-HitachiThreadDumpToStdout：拡張スレッドダンプを標準出力に出力しません。日立 JavaVM ログファイルだけに出力します。 デフォルト値は-XX:+HitachiThreadDumpToStdout です。
3	日立 JavaVM ログファイルオプション	-XX:HitachiJavaLog:<文字列>	ログファイル名のプレフィックスを指定します。ログファイル名は「文字列 xx.log」（xx は 01～99 の通し番号）で生成されます。 例えば，文字列に Samp と指定するとログファイル名は「Samp01.log」となります。 文字列にはパスの指定もできます。文字列にディレクトリを指定した場合，そのディレクトリにデフォルトの名称でログファイルを作成します。 デフォルト値は javalog です。

項番	分類	オプション名	説明
3	日立 JavaVM ログファイルオプション	-XX:HitachiJavaLogFileSize=<整数値>	1 ファイルの最大サイズ(単位:キロバイト)を 1024～INT_MAX バイトの整数で指定します。最大サイズを超えた場合は，そのファイルへの出力は行いません。 デフォルト値は 256 です。 範囲外の値が指定された場合は 1024 となります。
		-XX:HitachiJavaLogNumberOfFile=<整数値>	ログファイルの単純増加を防ぐため，作成する最大ファイル数を 1～99 の整数で指定します。最大ファイル数を超えた場合は，再度，最初に作成したファイルへの出力を開始します。 デフォルト値は 4 です。 100 以上の値が指定された場合は 99 となります。0 以下の値が指定された場合は 1 となります。
4	詳細時間出力オプション	-XX:[+|-]HitachiOutputMilliTime	ミリ秒までの時間を出力するかどうかを指定します。 • -XX:+HitachiOutputMilliTime:日立 JavaVM ログファイルに出力する日時に，ミリ秒まで出力します。 • -XX:-HitachiOutputMilliTime：日立 JavaVM ログファイルに出力する日時に，秒まで出力します。 デフォルト値は-XX:-HitachiOutputMilliTime です。
5	拡張 verbosegc 機能オプション	-XX:[+|-]HitachiVerboseGC※2	ガーベージコレクションが発生した場合，拡張 verbosegc 情報を出力するかどうかを指定します。 • -XX:+HitachiVerboseGC:ガーベージコレクションが発生した場合，拡張 verbosegc 情報を日立 JavaVM ログファイルに出力します。 ガーベージコレクションの内部領域である Eden，Survivor，Tenured，Perm 種別の情報を拡張 verbosegc 情報として出力します。 • -XX:-HitachiVerboseGC：ガーベージコレクションが発生した場合，拡張 verbosegc 情報を日立 JavaVM ログファイルに出力しません。 デフォルト値は-XX:-HitachiVerboseGC です。
		-XX:[+|-]HitachiVerboseGCPrintCause※3	ガーベージコレクションの要因内容を出力するかどうかを指定します。 • -XX:+HitachiVerboseGCPrintCause：ガーベージコレクションの要因内容を，拡張 verbosegc 情報の行末に出力します。 • -XX:-HitachiVerboseGCPrintCause：拡張 verbosegc 情報を通常形式で出力します。 デフォルト値は-XX:+HitachiVerboseGCPrintCause です。
6	OutOfMemoryError 発生時の拡張機能オプション	-XX:[+|-]HitachiOutOfMemoryStackTrace※2	OutOfMemoryError 発生時のスタックトレースを出力するかどうかを指定します。 • -XX:+HitachiOutOfMemoryStackTrace：OutOfMemoryError 発生時に，例外情報とスタックトレースを日立 JavaVM ログファイルに出力し

項番	分類	オプション名	説明
6	OutOfMemoryError 発生時の拡張機能オプション	-XX:[+|-]HitachiOutOfMemoryStackTrace※2	ます。スタックトレースは 1 スタックごとにバッファーに格納し，コード変換したあとに出力します。スタックトレースの出力は，OutOfMemoryError がスローされるたびに行われるため，OutOfMemoryError をキャッチして再スローした場合には複数回出力されます。なお，スレッド作成時に OutOfMemoryError となった場合は，スタックトレースは出力されません。 • -XX:-HitachiOutOfMemoryStackTrace：OutOfMemoryError 発生時に，スタックトレースを日立 JavaVM ログファイルに出力しません。 デフォルト値は-XX:-HitachiOutOfMemoryStackTrace です。
7	クラスライブラリトレース機能オプション	-XX:[+|-]HitachiJavaClassLibTrace※2	クラスライブラリのスタックトレースを出力するかどうかを指定します。 • -XX:+HitachiJavaClassLibTrace：クラスライブラリのスタックトレースを出力します。 • -XX:-HitachiJavaClassLibTrace：クラスライブラリのスタックトレースを出力しません。 デフォルト値は-XX:-HitachiJavaClassLibTrace です。
		-XX:HitachiJavaClassLibTraceLineSize=<整数値>	クラスライブラリのスタックトレースの 1 行の文字数（単位：バイト）を 1024～INT_MAX バイトの整数で指定します。1 行の文字数が指定した文字数を超えた場合は，at 以降の文字列の前半部分を削除して，指定された文字数分出力します。 デフォルト値は 1024 です。 範囲外の値が指定された場合は 1024 となります。
8	ローカル変数情報出力機能オプション	-XX:[+|-]HitachiLocalsInStackTrace	スレッドダンプ出力時のスタックトレースに，ローカル変数情報を出力するかどうかを指定します。 • -XX:+HitachiLocalsInStackTrace：スレッドダンプ出力時のスタックトレースに，ローカル変数情報を出力します。 • -XX:-HitachiLocalsInStackTrace：スレッドダンプ出力時のスタックトレースに，ローカル変数情報を出力しません。 デフォルト値は-XX:-HitachiLocalsInStackTrace です。
		-XX:[+|-]HitachiLocalsSimpleFormat	ローカル変数情報出力を簡易フォーマットにするかどうかを指定します。 • -XX:+HitachiLocalsSimpleFormat：ローカル変数情報出力を簡易フォーマットで出力します。 • -XX:-HitachiLocalsSimpleFormat：ローカル変数情報出力を通常フォーマットで出力します。 デフォルト値は-XX:-HitachiLocalsSimpleFormat です。

項番	分類	オプション名	説明
8	ローカル変数情報出力機能オプション	-XX: [+|-]HitachiTrueTypeInLocals	ローカル変数情報出力時に，ローカル変数オブジェクトの実際の型名を文字列として出力するかどうかを指定します。 • -XX:+HitachiTrueTypeInLocals：ローカル変数情報に，実際のオブジェクト型名を出力します。 • -XX:-HitachiTrueTypeInLocals：ローカル変数情報に，実際のオブジェクト型名を出力しません。 デフォルト値は-XX:-HitachiTrueTypeInLocals です。

注※1

<size>の単位はバイトです。

k または K を付記するとキロバイトで指定できます。また，m または M を付記するとメガバイトで指定できます。

（例）

-Xms6291456：6,291,456 バイト

-Xms6144k：6,144 キロバイト

-Xms6m：6 メガバイト

注※2

これらのオプションを指定した場合，日立 JavaVM ログファイルが出力されます。

注※3

-XX:+HitachiVerboseGC オプションが指定されている場合は，このオプションも指定されます。

9 アダプター用定義ファイル

この章では，標準提供アダプターを使用する場合に作成する，アダプター用定義ファイルについて説明します。

カスタムアダプターを使用する場合は，アダプター用定義ファイルを作成する必要はありません。

9.1　アダプター用定義ファイルの説明の記述形式

この章では，アダプター用定義ファイルの説明を次の形式で記述します。ただし，ファイルによっては説明しない項目もあります。また，各ファイルの固有情報を記載している場合があります。

記述形式

定義の記述形式を示します。

ファイル名

ファイル名を示します。

ファイルの格納先

ファイルの格納先を示します。

定義の詳細

定義するタグの詳細について説明します。

記述例

各定義の記述例を示します。

9.2　アダプター用定義ファイル作成上の注意事項

アダプター用定義ファイルを作成する際の注意事項を次に示します。

- アダプター用定義ファイルは XML1.0 形式で記述します。記述形式の仕様については，W3C による XML の仕様書（Extensible Markup Language (XML) 1.0）を参照してください。
- アダプター用定義ファイルの中で特殊文字（記号）を使用する場合は，サニタイジング（特殊文字の無効化）が必要です。次の表に示す対応に従って，特殊文字を置換して記述してください。

サニタイジングが必要な特殊文字	置換後の文字
<	<
>	>
&	&
"	"
'	'または'

9.3　アダプター用定義ファイルの一覧

アダプター用定義ファイルの一覧を次の表に示します。ファイルの詳細については，表中の参照先を参照してください。

表 9-1　アダプター用定義ファイルの一覧

項番	定義ファイル	概要	参照先
1	アダプターコマンド定義ファイル（AgentManagerDefinition.xml）	RMI 連携をする場合だけ，運用ディレクトリごとに作成します。 このファイルには，RMI 連携で使用するポート番号を設定します。	9.4
2	アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）	運用ディレクトリごとに必ず作成します。 このファイルには，アダプターグループの構成や，標準提供アダプターで実行するコールバックの処理などを設定します。	9.5 以降

9.4　アダプターコマンド定義ファイル (AgentManagerDefinition.xml)

アダプターコマンド定義ファイルでは，RMI 連携時に使用するポート番号を指定します。このファイルは，RMI 連携をする場合だけ作成してください。インプロセス連携をする場合には，このファイルを作成する必要はありません。

(1)　記述形式

```
<AgentManagerDefinition port="＜ポート番号＞"/>
```

(2)　ファイル名

AgentManagerDefinition.xml

(3)　ファイルの格納先

このファイルは，必ず次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥xml¥

(4)　定義の詳細

AgentManagerDefinition タグ（RMI 連携用のポート番号の定義）

この定義は必ず 1 個だけ記述します。

port="＜ポート番号＞"

標準提供アダプターを RMI 連携するときに使用するポート番号を 1～65535 の整数で指定します。省略した場合，20420 が仮定されます。

1～1023（Well Known ポート番号）の番号は指定しないようにしてください。

(5)　名前空間 URI

アダプターコマンド定義ファイルの名前空間 URI は，次の形式で表現されます。

```
http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/agentmanager
```

(6)　記述例

```
<?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　アダプター構成定義ファイル (AdaptorCompositionDefinition.xml)

アダプター構成定義ファイルでは，アダプターグループやコネクターなど，標準提供アダプターの構成について定義します。

ここでは，アダプター構成定義ファイルの概要と，アダプター構成定義ファイルの名前空間 URI について説明します。定義ファイルの各定義の詳細は，次の節以降で説明します。また，定義ファイルの記述例については，「9.13　アダプター構成定義ファイルの記述例」で説明します。

9.5.1　アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）の概要

(1)　記述形式

記述形式は，次の節以降で，定義ごとに説明します。

(2)　ファイル名

AdaptorCompositionDefinition.xml

(3)　ファイルの格納先

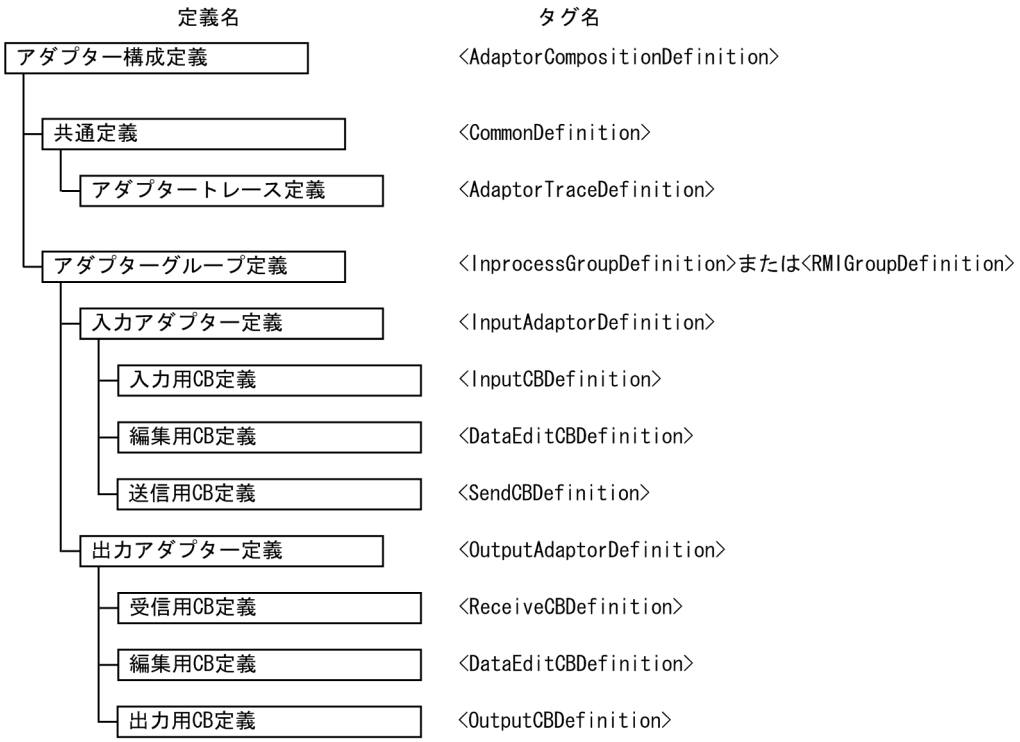
このファイルは，必ず次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥xml¥

(4)　定義の詳細

アダプター構成定義ファイルの定義は，階層構造になっています。アダプター構成定義ファイルの構造を次の図に示します。

図 9-1　アダプター構成定義ファイルの構造



階層構造になっている定義は，親子関係にあります。例えば，入力アダプター定義は，アダプターグループ定義の**子要素**として，アダプターグループ定義のタグ内に定義します。また，アダプターグループ定義は，入力アダプター定義の**親要素**となります。

アダプター構成定義ファイルの定義の一覧を次の表に示します。それぞれの定義の詳細は，表中の参照先で説明します。

表 9-2　アダプター構成定義ファイルの定義の一覧

項番	定義		説明	参照先
1	共通定義 (CommonDefinition)	アダプタートレース定義 (AdaptorTraceDefinition タグ)	すべての標準提供アダプターで共通の情報を定義します。	9.6
2	アダプターグループ定義	インプロセスグループ定義 (InprocessGroupDefinition タグ)	インプロセス連携で起動する入力アダプター，または出力アダプターを定義します。	9.7
3		RMI グループ定義 (RMIGroupDefinition タグ)	RMI 連携で起動する入力アダプター，または出力アダプターを定義します。	
4	アダプター定義	入力アダプター定義 (InputAdaptorDefinition タグ)	入力アダプターを構成するコールバックを定義します。 この定義は，入力アダプターごとに定義します。	9.8

項番	定義			説明	参照先
5	アダプター定義		出力アダプター定義 (OutputAdaptorDefinition タグ)	出力アダプターを構成するコールバックを定義します。 この定義は，出力アダプターごとに定義します。	9.8
6	CB 定義	入力用 CB 定義 (InputCBDefinition タグ)	ファイル入力コネクター定義 (FileInputConnectorDefinition タグ)	ファイルを入力する場合に使用する，ファイル入力コネクターの処理について定義します。	9.9, 9.10
7			HTTP パケット入力コネクター定義 (HttpPacketInputConnectorDefinition タグ)	HTTP パケットを入力する場合に使用する，HTTP パケット入力コネクターの処理について定義します。	
8		出力用 CB 定義 (OutputCBDefinition タグ)	ファイル出力コネクター定義 (FileOutputConnectorDefinition タグ)	ファイルに出力する場合に使用する，ファイル出力コネクターの処理について定義します。	
9			ダッシュボード出力コネクター定義 (DashboardOutputConnectorDefinition タグ)	ダッシュボードへ出力する場合に使用する，ダッシュボード出力コネクターの処理について定義します。	
10		編集用 CB 定義 (DataEditCBDefinition タグ)	フォーマット変換定義 (FormatDefinition タグ)	ファイルの入力，または出力を使用する場合に実施する，フォーマット変換の処理について定義します。	9.9, 9.11
11			マッピング定義 (MappingDefinition タグ)	マッピングの処理について定義します。	
12			フィルター定義 (FilterDefinition タグ)	レコードのフィルタリングの処理について定義します。	
13			レコード抽出定義 (RecordExtractionDefinition タグ)	レコードの抽出の処理について定義します。	
14		送信用 CB 定義 (SendCBDefinition タグ)	入力ストリーム定義 (streamInfo タグ)	入力アダプターが接続する入力ストリームについて定義します。	9.9, 9.12
15		受信用 CB 定義 (ReceiveCBDefinition タグ)	出力ストリーム定義 (streamInfo タグ)	出力アダプターが接続する出力ストリームについて定義します。	

(5)　CB 定義の定義の順序

CB 定義は，入力アダプター定義の場合と，出力アダプター定義の場合とで，定義の順序が異なります。それぞれの場合の定義の順序を次に示します。

入力アダプター定義の場合の定義の順序

1. 入力用 CB 定義（ファイル入力コネクター定義など）
2. 編集用 CB 定義（フォーマット変換定義など）
3. 送信用 CB 定義（入力ストリーム定義）

出力アダプター定義の場合の定義の順序

1. 受信用 CB 定義（出力ストリーム定義）
2. 編集用 CB 定義（フォーマット変換定義など）
3. 出力用 CB 定義（ファイル出力コネクター定義など）

また，編集用 CB 定義については，次に示すように，使用する機能に応じて定義します。

- 標準提供アダプターで入出力するデータの種類によって，編集用 CB 定義で定義する内容が異なります。例えば，フォーマット変換定義は，ファイルの入力，またはファイルへの出力の場合しか定義しません。
- フィルター定義とレコード抽出定義は，任意で実行する処理です。必要に応じて定義します。

CB 定義で定義するコールバックの処理については，「11.　定義ファイルでの定義内容の詳細」を参照してください。

9.5.2　アダプター構成定義ファイルの名前空間 URI

アダプター構成定義ファイルでは，**名前空間 URI** は，次の形式で表現されます。

```
http://www.hitachi.co.jp/soft/xml/sdp/adaptor/xxx～
```

形式の「xxx～」の部分は，アダプター構成定義ファイルの定義によって異なります。名前空間は，指定する定義に応じて宣言してください。定義と「xxx～」の部分に記述する名前空間の対応を次の表に示します。

表 9-3　定義と名前空間の対応

項番	定義	名前空間
1	アダプター構成定義	definition
2	共通定義	definition/common
3	アダプタートレース定義	
4	インプロセスグループ定義	definition/adaptor
5	RMI グループ定義	
6	入力アダプター定義	
7	出力アダプター定義	
8	入力用 CB 定義	definition/callback

項番	定義	名前空間
9	出力用 CB 定義	definition/callback
10	編集用 CB 定義	
11	送信用 CB 定義	
12	受信用 CB 定義	
13	ファイル入力コネクター定義	definition/callback/FileInputConnectorDefinition
14	HTTP パケット入力コネクター定義	definition/callback/HttpPacketInputConnectorDefinition
15	ファイル出力コネクター定義	definition/callback/FileOutputConnectorDefinition
16	ダッシュボード出力コネクター定義	definition/callback/DashboardOutputConnectorDefinition
17	フォーマット変換定義	definition/callback/FormatDefinition
18	マッピング定義	definition/callback/MappingDefinition
19	フィルター定義	definition/callback/FilterDefinition
20	レコード抽出定義	definition/callback/RecordExtractionDefinition

9.6　アダプター構成定義ファイルの共通定義

ここでは，アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）の共通定義について説明します。共通定義の一覧を次の表に示します。

表 9-4　共通定義の一覧

項番	共通定義	親要素	参照先
1	共通定義 （CommonDefinition タグ）	アダプター構成定義	9.6.1
2	アダプタートレース定義 （AdaptorTraceDefinition タグ）	共通定義	9.6.2

9.6.1　共通定義

この定義は 1 個だけ記述できます。この定義は省略できません。

(1)　記述形式

```
<CommonDefinition>
  ＜アダプタートレース定義＞
</CommonDefinition>
```

(2)　定義の詳細

CommonDefinition タグ（全体情報の定義）

共通定義の全体情報を定義します。

＜アダプタートレース定義＞

アダプタートレース定義については，「9.6.2　アダプタートレース定義」を参照してください。

9.6.2　アダプタートレース定義

アダプタートレース定義（AdaptorTraceDefinition タグ）は，「9.6.1　共通定義」で説明した共通定義（CommonDefinition タグ）の子要素として定義します。

(1)　記述形式

```
<AdaptorTraceDefinition trace=”{ON | OFF} ”/>
```

(2)　定義の詳細

AdaptorTraceDefinition タグ（全体情報の定義）

アダプタートレース定義の全体情報を定義します。

trace="｛ON ｜ OFF｝"

アダプタートレースを出力するかどうかを指定します。省略した場合，OFF が仮定されます。

指定できる値を次に示します。

- ON

  アダプタートレースを出力します。

- OFF

アダプタートレースを出力しません。

9.7　アダプター構成定義ファイルのアダプターグループ定義

ここでは，アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）のアダプターグループ定義（InprocessGroupDefinition タグ，または RMIGroupDefinition タグ）について説明します。アダプターグループ定義の一覧を次の表に示します。

表 9-5　アダプターグループ定義の一覧

項番	アダプターグループ定義	親要素	参照先
1	インプロセスグループ定義 （InprocessGroupDefinition タグ）	アダプター構成定義	9.7.1
2	RMI グループ定義 （RMIGroupDefinition タグ）		9.7.2

9.7.1　インプロセスグループ定義

インプロセスグループ定義（InprocessGroupDefinition タグ）では，インプロセス連携で起動する入力アダプター，または出力アダプターを定義します。

この定義は 1 個だけ記述できます。標準提供アダプターと SDP サーバをインプロセスで連携しない場合，この定義は省略できます。なお，この定義を省略する場合は，RMI グループ定義を記述してください。

(1)　記述形式

```
<InprocessGroupDefinition name="＜アダプターグループ名＞"
 dashboardPortNo="＜RMIサーバのポート番号＞">
  ＜アダプター定義＞
</InprocessGroupDefinition>
```

(2)　定義の詳細

InprocessGroupDefinition タグ（全体情報の定義）

インプロセスグループ定義の全体情報を定義します。

name="＜アダプターグループ名＞"

アダプターグループを識別するための名称を 1～32 文字の半角英数字，またはアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は省略できません。

ここで指定するアダプターグループ名は，インプロセス連携用プロパティファイルのファイル名に指定するアダプターグループ名と一致させてください。

dashboardPortNo="＜ RMI サーバのポート番号＞"

ダッシュボード出力コネクターが使用する RMI サーバのポート番号を 1～65535 の整数で指定します。省略した場合，20421 が仮定されます。

＜アダプター定義＞

アダプター定義については，「9.8　アダプター構成定義ファイルのアダプター定義」を参照してください。

9.7.2　RMI グループ定義

RMI グループ定義（RMIGroupDefinition タグ）では，RMI 連携で起動する入力アダプター，または出力アダプターを定義します。

この定義は 1 個だけ記述できます。標準提供アダプターと SDP サーバを RMI で連携しない場合，この定義は省略できます。なお，この定義を省略する場合は，インプロセスグループ定義を記述してください。

(1)　記述形式

```
<RMIGroupDefinition name=”＜アダプターグループ名＞”
 dashboardPortNo=”＜RMIサーバのポート番号＞”>
  ＜アダプター定義＞
</RMIGroupDefinition>
```

(2)　定義の詳細

RMIGroupDefinition タグ（全体情報の定義）

RMI グループ定義の全体情報を定義します。

name="＜アダプターグループ名＞"

アダプターグループを識別するための名称を 1～32 文字の半角英数字，またはアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は省略できません。

dashboardPortNo="＜ RMI サーバのポート番号＞"

ダッシュボード出力コネクターが使用する RMI サーバのポート番号を 1～65535 の整数で指定します。省略した場合，20421 が仮定されます。

＜アダプター定義＞

アダプター定義については，「9.8　アダプター構成定義ファイルのアダプター定義」を参照してください。

9.8　アダプター構成定義ファイルのアダプター定義

ここでは，アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）のアダプター定義について説明します。

アダプター定義は，「9.7　アダプター構成定義ファイルのアダプターグループ定義」で説明したアダプターグループ定義（InprocessGroupDefinition タグ，または RMIGroupDefinition タグ）の子要素として定義します。

アダプター定義の一覧を次の表に示します。

表 9-6　アダプター定義の一覧

項番	アダプター定義	親要素	参照先
1	入力アダプター定義 （InputAdaptorDefinition タグ）	アダプターグループ定義	9.8.1
2	出力アダプター定義 （OutputAdaptorDefinition タグ）		9.8.2

9.8.1　入力アダプター定義

入力アダプター定義は，「9.7　アダプター構成定義ファイルのアダプターグループ定義」で説明したアダプターグループ定義（InprocessGroupDefinition タグ，または RMIGroupDefinition タグ）の子要素として定義します。

入力アダプター定義は，64 個まで記述できます。この定義は省略できます。

(1)　記述形式

```
<InputAdaptorDefinition name="＜アダプター名＞"
  interval="＜アダプターの実行間隔＞"
  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定義＞
</InputAdaptorDefinition>
```

(2)　定義の詳細

InputAdaptorDefinition タグ（全体情報の定義）

入力アダプター定義の全体情報を定義します。

name="＜アダプター名＞"

入力アダプターを識別するための名称を 1～100 文字の半角英数字，またはアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は省略できません。アダプター名は，アダプターグループ定義内で一意となるように指定してください。

interval="＜アダプターの実行間隔＞"

入力アダプターを実行する間隔（単位：ミリ秒）を 0～60000 の整数で指定します。

入力アダプター内に定義されている最後のコールバックが終了してから最初のコールバックを起動するまでの間，指定した間隔で入力アダプターの処理が停止します。0 を指定した場合は停止しません。省略した場合，0 が仮定されます。

charCode="｛SJIS ｜ MS932 ｜ EUC-JP ｜ UTF-8 ｜ UTF-8-BOM ｜ UTF-16BE ｜ UTF-16BE-BOM ｜ UTF-16LE ｜ UTF-16LE-BOM｝"

入力元で扱われている文字コードを指定します。省略した場合，MS932 が仮定されます。

文字コードと charCode 属性に指定する値の対応を次の表に示します。

文字コード	charCode 属性の指定値
シフト JIS	SJIS
MS932	MS932
EUC	EUC-JP
UTF-8（BOM なし）	UTF-8
UTF-8（BOM あり）	UTF-8-BOM
UTF-16（ビッグエンディアン BOM なし）	UTF-16BE
UTF-16（ビッグエンディアン BOM あり）	UTF-16BE-BOM
UTF-16（リトルエンディアン BOM なし）	UTF-16LE
UTF-16（リトルエンディアン BOM あり）	UTF-16LE-BOM

lineFeed="｛CR_LF ｜ LF｝"

入力元の改行コードを指定します。省略した場合，CR_LF が仮定されます。

指定できる値を次に示します。

- CR_LF

  CR（Carriage Return）と LF（Line Feed）の組み合わせを改行として扱います。

- LF

  LF（Line Feed）を改行として扱います。

＜CB 定義＞

指定できる CB 定義を次に示します。

- 入力用 CB 定義
- 編集用 CB 定義
- 送信用 CB 定義

CB 定義については，「9.9　アダプター構成定義ファイルの CB 定義」を参照してください。

9.8.2　出力アダプター定義

出力アダプター定義は，「9.7　アダプター構成定義ファイルのアダプターグループ定義」で説明したアダプターグループ定義（InprocessGroupDefinition タグ，または RMIGroupDefinition タグ）の子要素として定義します。

出力アダプター定義は，64 個まで記述できます。この定義は省略できます。

(1)　記述形式

```
<OutputAdaptorDefinition name="＜アダプター名＞"
  interval="＜アダプターの実行間隔＞"
  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定義＞
</OutputAdaptorDefinition>
```

(2)　定義の詳細

OutputAdaptorDefinition タグ（全体情報の定義）

出力アダプター定義の全体情報を定義します。

name="＜アダプター名＞"

出力アダプターを識別するための名称を 1～100 文字の半角英数字，またはアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は省略できません。アダプター名は，アダプターグループ定義内で一意となるように指定してください。

interval="＜アダプターの実行間隔＞"

出力アダプターを実行する間隔（単位：ミリ秒）を 0～60000 の整数で指定します。

出力アダプター内に定義されている最後のコールバックが終了してから最初のコールバックを起動するまでの間，指定した間隔で出力アダプターの処理が停止します。0 を指定した場合は停止しません。省略した場合，0 が仮定されます。

charCode="｛SJIS ｜ MS932 ｜ EUC-JP ｜ UTF-8 ｜ UTF-8-BOM ｜ UTF-16BE ｜ UTF-16BE-BOM ｜ UTF-16LE ｜ UTF-16LE-BOM｝"

出力先で扱われている文字コードを指定します。省略した場合，MS932 が仮定されます。文字コードと charCode 属性に指定する値の対応については，「9.8.1　入力アダプター定義」の charCode 属性の説明を参照してください。

lineFeed="｛CR_LF ｜ LF｝"

出力先の改行コードを指定します。省略した場合，CR_LF が仮定されます。

指定できる値を次に示します。

- CR_LF

  CR（Carriage Return）と LF（Line Feed）の組み合わせを改行として扱います。

- LF

  LF（Line Feed）を改行として扱います。

＜ CB 定義＞

指定できる CB 定義を次に示します。

- 受信用 CB 定義
- 編集用 CB 定義
- 出力用 CB 定義

CB 定義については，「9.9　アダプター構成定義ファイルの CB 定義」を参照してください。

9.9　アダプター構成定義ファイルの CB 定義

ここでは，アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）の CB 定義について説明します。

CB 定義では，標準提供アダプターのデータの入出力，データの編集，およびタプルの送受信の各処理を実施する，コールバックについて定義します。

CB 定義は，次に示す定義の子要素として定義します。

- 「9.8.1　入力アダプター定義」で説明した入力アダプター定義（InputAdaptorDefinition タグ）
- 「9.8.2　出力アダプター定義」で説明した出力アダプター定義（OutputAdaptorDefinition タグ）

CB 定義の一覧を次の表に示します。

表 9-7　CB 定義の一覧

項番	CB 定義	親要素	参照先
1	入力用 CB 定義 （InputCBDefinition タグ）	入力アダプター定義	9.9.1
2	出力用 CB 定義 （OutputCBDefinition タグ）	出力アダプター定義	9.9.2
3	編集用 CB 定義 （DataEditCBDefinition タグ）	入力アダプター定義，または出力アダプター定義	9.9.3
4	送信用 CB 定義 （SendCBDefinition タグ）	入力アダプター定義	9.9.4
5	受信用 CB 定義 （ReceiveCBDefinition タグ）	出力アダプター定義	9.9.5

なお，CB 定義は，入力アダプター定義の場合と，出力アダプター定義の場合とで，定義の順序が異なります。CB 定義の定義の順序については，「9.5.1(5)　CB 定義の定義の順序」を参照してください。

9.9.1　入力用 CB 定義

入力用 CB 定義（InputCBDefinition タグ）は，「9.8.1　入力アダプター定義」で説明した入力アダプター定義（InputAdaptorDefinition タグ）の子要素として定義します。

(1)　記述形式

```
<InputCBDefinition class="＜クラス名＞"
  name="＜コールバック名＞"
  interval="＜コールバックの実行間隔＞">
  ＜入力コネクター定義＞
</InputCBDefinition>
```

(2)　定義の詳細

InputCBDefinition タグ（全体情報の定義）

入力用 CB 定義の全体情報を定義します。

class="＜クラス名＞"

入力用 CB 定義の機能が実装されているクラス名を指定します。この値は固定値です。

クラス名は，＜入力コネクター定義＞で指定する入力コネクター定義の種類ごとに異なります。指定するクラス名を次の表に示します。

入力コネクター定義の種類	class 属性の値
ファイル入力コネクター定義	jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileInputCBImpl
HTTP パケット入力コネクター定義	jp.co.Hitachi.soft.sdp.adaptor.callback.io.packetinput.HttpPacketInputCBImpl

name="＜コールバック名＞"

機能を識別するためのコールバック名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は省略できません。コールバック名にはほかの CB 定義と重複した値を指定できますが，アダプタートレースやメッセージログで入力アダプターの処理の識別情報として出力するため，できるだけ重複しないように指定してください。

interval="＜コールバックの実行間隔＞"

コールバックの実行間隔（単位：ミリ秒）を 0～60000 の整数で指定します。コールバックを実行したあと，指定した間隔で入力アダプターの処理が停止します。0 の場合は停止しません。省略した場合，0 が設定されます。

＜入力コネクター定義＞

指定できる CB 定義を次に示します。

- ファイル入力コネクター定義
- HTTP パケット入力コネクター定義

CB 定義については，「9.10　アダプター構成定義ファイルの入出力用 CB 定義」を参照してください。

9.9.2　出力用 CB 定義

出力用 CB 定義（OutputCBDefinition タグ）は，「9.8.2　出力アダプター定義」で説明した出力アダプター定義（OutputAdaptorDefinition タグ）の子要素として定義します。

(1)　記述形式

```
<OutputCBDefinition class="＜クラス名＞"
  name="＜コールバック名＞"
  interval="＜コールバックの実行間隔＞">
  ＜出力コネクター定義＞
</OutputCBDefinition>
```

(2)　定義の詳細

OutputCBDefinition タグ（全体情報の定義）

出力用 CB 定義の全体情報を定義します。

class="＜クラス名＞"

出力用 CB 定義の機能が実装されているクラス名を指定します。この値は固定値です。

クラス名は，＜出力コネクター定義＞で指定する出力コネクター定義の種類ごとに異なります。指定するクラス名を次の表に示します。

出力コネクター定義の種類	class 属性の値
ファイル出力コネクター定義	jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileOutputCBImpl
ダッシュボード出力コネクター定義	jp.co.Hitachi.soft.sdp.adaptor.callback.io.dashboard.DashboardOutputCBImpl

name="＜コールバック名＞"

機能を識別するためのコールバック名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は省略できません。コールバック名にはほかの CB 定義と重複した値を指定できますが，アダプタートレースやメッセージログで出力アダプターの処理の識別情報として出力するため，できるだけ重複しないように指定してください。

interval="＜コールバックの実行間隔＞"

コールバックの実行間隔（単位：ミリ秒）を 0～60000 の整数で指定します。コールバックを実行したあと，指定した間隔で出力アダプターの処理が停止します。0 の場合は停止しません。省略した場合，0 が設定されます。

＜出力コネクター定義＞

指定できる CB 定義を次に示します。

- ファイル出力コネクター定義
- ダッシュボード出力コネクター定義

CB 定義については，「9.10　アダプター構成定義ファイルの入出力用 CB 定義」を参照してください。

9.9.3　編集用 CB 定義

編集用 CB 定義（DataEditCBDefinition タグ）は，次に示す定義の子要素として定義します。

- 「9.8.1　入力アダプター定義」で説明した入力アダプター定義（InputAdaptorDefinition タグ）
- 「9.8.2　出力アダプター定義」で説明した出力アダプター定義（OutputAdaptorDefinition タグ）

(1)　記述形式

```
<DataEditCBDefinition class=”＜クラス名＞”
  name=”＜コールバック名＞”
  interval=”＜コールバックの実行間隔＞”>
  ＜データ編集用CB定義＞
</DataEditCBDefinition>
```

(2)　定義の詳細

DataEditCBDefinition タグ（全体情報の定義）

編集用 CB 定義の全体情報を定義します。

class="＜クラス名＞"

編集用 CB 定義の機能が実装されているクラス名を指定します。この値は固定値です。

クラス名は，＜データ編集用 CB 定義＞で指定するデータ編集用 CB 定義の種類ごとに異なります。また，入力アダプター定義で指定するか，出力アダプター定義で指定するかによって，指定するクラス名が異なる場合があります。指定するクラス名を次の表に示します。

データ編集用 CB 定義の種類	class 属性の値
フォーマット変換定義	入力アダプター定義で指定するか，出力アダプター定義で指定するかによって，値が異なります。 • 入力アダプター定義で指定する値 jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.formattranslate.InputFormatTranslatorCBImpl • 出力アダプター定義で指定する値 jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.formattranslate.OutputFormatTranslatorCBImpl
マッピング定義	入力アダプター定義で指定するか，出力アダプター定義で指定するかによって，値が異なります。 • 入力アダプター定義で指定する値 jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl • 出力アダプター定義で指定する値 jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.OutputMappingCBImpl
フィルター定義	入力アダプター定義で指定する場合，出力アダプター定義で指定する場合，どちらの場合でも次の値を指定します。 jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.filter.FilterCBImpl
レコード抽出定義	レコード抽出定義は，出力アダプター定義では指定できません。 入力アダプター定義では，次の値を指定します。 jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.recordextract.RecordExtractionCBImpl

name="＜コールバック名＞"

機能を識別するためのコールバック名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は省略できません。コールバック名にはほかの CB 定義と重複した値を指定できますが，アダプタートレースやメッセージログで，入力アダプターまたは出力アダプターの処理の識別情報として出力するため，できるだけ重複しないように指定してください。

interval="＜コールバックの実行間隔＞"

コールバックの実行間隔（単位：ミリ秒）を 0～60000 の整数で指定します。コールバックを実行したあと，指定した間隔で，入力アダプターまたは出力アダプターの処理が停止します。0 の場合は停止しません。省略した場合，0 が設定されます。

＜データ編集用 CB 定義＞

指定できる CB 定義を次に示します。

- フォーマット変換定義
- マッピング定義
- フィルター定義
- レコード抽出定義（出力アダプター定義では指定できません）

CB 定義については，「9.11　アダプター構成定義ファイルのデータ編集用 CB 定義」を参照してください。

9.9.4　送信用 CB 定義

送信用 CB 定義（SendCBDefinition タグ）は，「9.8.1　入力アダプター定義」で説明した入力アダプター定義（InputAdaptorDefinition タグ）の子要素として定義します。

(1)　記述形式

```
<SendCBDefinition class=”＜クラス名＞”
  name=”＜コールバック名＞”
  interval=”＜コールバックの実行間隔＞”>
  ＜入力ストリーム定義＞
</SendCBDefinition>
```

(2)　定義の詳細

SendCBDefinition タグ（全体情報の定義）

送信用 CB 定義の全体情報を定義します。

class="＜クラス名＞"

送信用 CB 定義の機能が実装されているクラス名を指定します。この値は固定値です。指定するクラス名を次に示します。

jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.SendCBImpl

name="＜コールバック名＞"

機能を識別するためのコールバック名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は省略できません。コールバック名にはほかの CB 定義と重複した値を指定できますが，アダプタートレースやメッセージログで入力アダプターの処理の識別情報として出力するため，できるだけ重複しないように指定してください。

interval="＜コールバックの実行間隔＞"

コールバックの実行間隔（単位：ミリ秒）を 0～60000 の整数で指定します。コールバックを実行したあと，指定した間隔で入力アダプターの処理が停止します。0 の場合は停止しません。省略した場合，0 が設定されます。

＜入力ストリーム定義＞

入力ストリーム定義については，「9.12.1　入力ストリーム定義」を参照してください。

9.9.5　受信用 CB 定義

受信用 CB 定義（ReceiveCBDefinition タグ）は，「9.8.2　出力アダプター定義」で説明した出力アダプター定義（OutputAdaptorDefinition タグ）の子要素として定義します。

(1)　記述形式

```
<ReceiveCBDefinition class=”＜クラス名＞”
  name=”＜コールバック名＞”
  interval=”＜コールバックの実行間隔＞”>
  ＜出力ストリーム定義＞
</ReceiveCBDefinition>
```

(2)　定義の詳細

ReceiveCBDefinition タグ（全体情報の定義）

受信用 CB 定義の全体情報を定義します。

class="＜クラス名＞"

受信用 CB 定義の機能が実装されているクラス名を指定します。この値は固定値です。指定するクラス名を次に示します。

jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.ReceiveCBImpl

name="＜コールバック名＞"

機能を識別するためのコールバック名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は省略できません。コールバック名にはほかの CB 定義と重複した値を指定できますが，アダプタートレースやメッセージログで出力アダプターの処理の識別情報として出力するため，できるだけ重複しないように指定してください。

interval="＜コールバックの実行間隔＞"

コールバックの実行間隔（単位：ミリ秒）を 0～60000 の整数で指定します。コールバックを実行したあと，指定した間隔で出力アダプターの処理が停止します。0 の場合は停止しません。省略した場合，0 が設定されます。

＜出力ストリーム定義＞

出力ストリーム定義については，「9.12　アダプター構成定義ファイルの送受信用 CB 定義」を参照してください。

9.10　アダプター構成定義ファイルの入出力用 CB 定義

ここでは，アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）の入出力用 CB 定義について説明します。

入力コネクター定義は，「9.9.1　入力用 CB 定義」で説明した入力用 CB 定義（InputCBDefinition タグ）の子要素として定義します。また，出力コネクター定義は，「9.9.2　出力用 CB 定義」で説明した出力用 CB 定義（OutputCBDefinition タグ）の子要素として定義します。

入出力用 CB 定義の一覧を次の表に示します。

表 9-8　入出力用 CB 定義の一覧

項番	入出力用 CB 定義		親要素	参照先
1	入力コネクター定義	ファイル入力コネクター定義（FileInputConnectorDefinition タグ）	入力用 CB 定義	9.10.1
2		HTTP パケット入力コネクター定義（HttpPacketInputConnectorDefinition タグ）		9.10.2
3	出力コネクター定義	ファイル出力コネクター定義（FileOutputConnectorDefinition タグ）	出力用 CB 定義	9.10.3
4		ダッシュボード出力コネクター定義（DashboardOutputConnectorDefinition タグ）		9.10.4

9.10.1　ファイル入力コネクター定義

ファイル入力コネクター定義（FileInputConnectorDefinition タグ）は，入力用 CB 定義（InputCBDefinition タグ）の子要素として定義します。

なお，ファイルの入力の処理については，「11.2　ファイルの入力」を参照してください。

(1)　記述形式

```
<FileInputConnectorDefinition>
  <input readType=" {BATCH | REAL_TIME} "
   compositionType=" {WRAP_AROUND | ANTI_WRAP_AROUND} "
   interval="＜監視時間間隔＞"
   retryCount="＜監視リトライ回数＞"
   readOrder=" {DEFINED | MODIFIED} "
   readUnit="＜読み込み単位＞"/>
  <file path="＜入力パス名＞"
   name="＜入力ファイル名＞"
   messageLog=" {ON | OFF} "/>
</FileInputConnectorDefinition>
```

(2)　定義の詳細

FileInputConnectorDefinition タグ（全体情報の定義）

ファイル入力コネクター定義の全体情報を定義します。この定義は必ず 1 個だけ記述します。

input タグ（読み込み定義）

入力ファイルの読み込み処理，および入力ファイルの監視処理の情報を定義します。この定義は必ず 1 個だけ記述します。

readType="｛BATCH ｜ REAL_TIME｝"

入力ファイルの読み込み方式を指定します。省略した場合，REAL_TIME が仮定されます。

指定できる値を次に示します。

- BATCH

  バッチ処理モードで，入力ファイルを読み込みます。

- REAL_TIME

  リアルタイム処理モードで，入力ファイルを読み込みます。

compositionType="｛WRAP_AROUND ｜ ANTI_WRAP_AROUND｝"

入力ファイルのファイル構成を指定します。省略した場合，ANTI_WRAP_AROUND が仮定されます。

指定できる値を次に示します。

- WRAP_AROUND

  ラップアラウンド構成の入力ファイルを読み込みます。

  なお，readType 属性に BATCH を指定する場合は，WRAP_AROUND を指定できません。また，readType 属性に REAL_TIME，および readOrder 属性に DEFINED を指定する場合も，WRAP_AROUND を指定できません。

- ANTI_WRAP_AROUND

  非ラップアラウンド構成の入力ファイルを読み込みます。

  なお，readOrder 属性に MODIFIED を指定する場合は，ANTI_WRAP_AROUND を指定できません。

interval="＜監視時間間隔＞"

readType 属性で REAL_TIME を指定した場合に，入力ファイル格納ディレクトリを監視して，入力対象のファイルが新規作成されているかどうかを確認する時間間隔（単位：ミリ秒）を 0～1000000 の整数で指定します。0 を指定した場合，監視処理を連続で実行します。省略した場合，1000 が仮定されます。

retryCount="＜監視リトライ回数＞"

入力ファイル格納ディレクトリの監視処理のリトライ回数を 0～1000 の整数で指定します。省略した場合，100 が仮定されます。

リトライ回数がこの属性で指定した値を超えると，入力ファイル格納ディレクトリに入力対象のファイルが存在しないことを示す警告メッセージ KFSP46203-W が 1 回出力され，監視が継続されます。

監視処理のリトライは，その時点の入力対象ファイルごとに実行され，入力対象ファイルの読み込みが行われた時点でリトライ回数が 0 にリセットされます。このため，リトライ回数のしきい値判定は入力ファイルごとに実行され，警告メッセージは入力ファイルごとに 1 回出力されます。

readOrder="｛DEFINED ｜ MODIFIED｝"

入力ファイルを読み込む順番を指定します。省略した場合，DEFINED が仮定されます。

- DEFINED

  file タグの name 属性で指定した順番で，入力ファイルを読み込みます。

- MODIFIED

  入力ファイルの更新時刻の順番で，入力ファイルを読み込みます。

readUnit="＜読み込み単位＞"

ファイル入力コネクターが一度に読み込むレコード数（単位：行数）を 1～10000 の整数で指定します。省略した場合，1 が仮定されます。

file タグ（入力ファイルの定義）

入力ファイルの情報を定義します。この定義は必ず 1 個だけ記述します。

path="＜入力パス名＞"

入力ファイル格納ディレクトリのパス名を 1～160 文字の半角文字で指定します。この属性は省略できません。

この属性に指定できる文字を次に示します。

半角英数字，「!」，「#」，「$」，「%」，「&」，「’」，「(」，「)」，「=」，「~」，「`」，「{」，「}」，「+」，「-」，「^」，「.」，「_」，「@」，「¥」，「/」，「:」，半角スペース

name="＜入力ファイル名＞"

入力対象のファイル名をファイル名，または通番で指定します。この属性は省略できません。

- ファイル名で指定する場合

  ファイル名は，path 属性で指定した入力ファイル格納ディレクトリからの相対パスで指定します。複数指定する場合は，コンマ（,）で区切って指定してください。ただし，サブディレクトリは指定できません。入力ファイル名一つにつき，半角文字で 1～60 文字以内となるように指定してください。

  なお，予約デバイス名（NUL など）をファイル名に指定しないでください。予約デバイス名を指定した場合，動作は保証されません。

- 通番で指定する場合

  path 属性で指定した入力ファイル格納ディレクトリに格納されているファイルを通番で指定します。指定方法については，「(4)　入力ファイル名の通番での指定」を参照してください。

入力ファイル名は，1～1,024 文字の半角文字で指定します。「¥」，「/」，および「:」は指定できません。

この属性に指定できる文字を次に示します。

半角英数字，「!」，「#」，「$」，「%」，「&」，「’」，「(」，「)」，「=」，「~」，「`」，「{」，「}」，「+」，「-」，「^」，「.」，「_」，「@」，半角スペース

messageLog="｛ON｜OFF｝"

入力ファイルの読み込み開始時に，開始時刻と入力ファイル名をメッセージログに出力するかどうかを指定します。省略した場合，OFF が仮定されます。

指定できる値を次に示します。

- ON

  メッセージログに出力します。

- OFF

  メッセージログに出力しません。

(3)　記述例

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
 xmlns:ficon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
FileInputConnectorDefinition">
<!-- 途中略 -->

<!-- 入力用CB定義 -->
<cb:InputCBDefinition
```

```
 class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileInputCBImpl" name="inputer1">
  <!-- ファイル入力コネクター定義 -->
  <ficon:FileInputConnectorDefinition>
    <!-- 読み込み定義 -->
    <ficon:input readType="REAL_TIME" interval="600000"
     retryCount="100" readUnit="1"/>
    <!-- 入力ファイルの定義 -->
    <ficon:file path="C:¥tmp¥" name="a[1-100].txt"/>
  </ficon:FileInputConnectorDefinition>
</cb:InputCBDefinition>
```

(4)　入力ファイル名の通番での指定

file タグの name 属性で，入力ファイル名を通番で指定する場合の記述形式について説明します。

- **通番の記述形式**

  ＜接頭文字列＞＜通番指定1＞＜中間文字列＞＜通番指定2＞＜接尾文字列＞

  指定値について説明します。

  ＜接頭文字列＞，＜中間文字列＞，＜接尾文字列＞

  半角文字列で指定します。

  ＜通番指定１＞，＜通番指定２＞

  「[」と「]」で囲まれた「下限値-上限値」，または「下限値-」で，通番を指定します。下限値，および上限値は，8 けた以下の半角数字で指定します。

- **通番の記述規則**

  - 通番指定ができるファイルは一つだけです。
  - 接頭文字列，中間文字列，および接尾文字列には，「[」または「]」を指定できません。
  - 通番指定の下限値は省略できません。上限値を省略した場合は，上限値として最大値（99999999）が仮定されます。
  - 通番は，「下限値≦上限値」，かつ「上限値のけた数≧下限値のけた数」となるように指定してください。
  - 通番指定のファイルを入力する際の優先順位は，次のとおりです。

    1. 通番指定 1 の値が小さいもの
    2. 通番指定 2 の値が小さいもの

    例えば，「a1b2.txt，a2b1.txt」では「a1b2.txt」が先に入力されて，「a2b2.txt，a2b1.txt」では「a2b1.txt」が先に入力されます。
  - 中間文字列には，数字でない文字を一つ以上含めてください。
  - ファイル入力コネクターは，下限値から上限値までのファイル名を順番に入力します。入力ファイルの通番が不連続な場合も，通番の順序に従って入力します。例えば，通番が「1，2，4」となっていて 3 が抜けている場合，1 と 2 に続いて 4 を入力します。

- **通番の指定例**

  通番の指定例を次の表に示します。

指定例	処理結果	正常時の入力対象のファイルの並び順
a[1-100].txt	○	a1.txt，a2.txt，…（省略）…，a9.txt，a10.txt，a11.txt，a12.txt，…（省略）…，a99.txt，a100.txt
a[01-100].txt	○	a01.txt，a02.txt，…（省略）…，a09.txt，a10.txt，a11.txt，a12.txt，…（省略）…，a99.txt，a100.txt

指定例	処理結果	正常時の入力対象のファイルの並び順
a[001-100].txt	○	a001.txt，a002.txt，…（省略）…，a009.txt，a010.txt，a011.txt，a012.txt，…（省略）…，a099.txt，a100.txt
a[01-10]_[1-100].txt	○	a01_1.txt，a01_2.txt，…（省略）…，a01_9.txt，a01_10.txt，a01_a11.txt，…（省略）…，a01_99.txt，a01_100.txt，a02_1.txt，a02_2.txt，…（省略）…，a02_9.txt，a01_10.txt，a02_a11.txt，…（省略）…，a02_99.txt，a02_100.txt，…（省略）…，a10_1.txt，a10_2.txt，…（省略）…，a10_9.txt，a10_10.txt，a10_a11.txt，…（省略）…，a10_99.txt，a10_100.txt
a[5-3].txt	×※1	－
a[001-9].txt	×※2	－
a[1-005].txt	○	a1.txt，a2.txt，a3.txt，a4.txt，a5.txt
a.txt，b[1-3].txt	×※3	－
a[1-3].txt，b[4-6].txt	×※3	－

（凡例）

○：正常に処理されます。

×：エラーとなります。

－：該当しません。

注※1

下限値が上限値よりも大きいため，エラーとなります。

注※2

下限値のけた数が上限値よりも大きいため，エラーとなります。

注※3

通番指定がある場合，複数ファイルは指定できないため，エラーとなります。

9.10.2　HTTP パケット入力コネクター定義

HTTP パケット入力コネクター定義（HttpPacketInputConnectorDefinition タグ）は，入力用 CB 定義（InputCBDefinition タグ）の子要素として定義します。

なお，HTTP パケットの入力の処理については，「11.3　HTTP パケットの入力」を参照してください。

(1)　記述形式

```
<HttpPacketInputConnectorDefinition>
  <input buffersize="＜入出力バッファーサイズ＞"
   assemblingtime="＜分割パケット組み立て時間＞">
    <packetdata
     globalheader="＜グローバルヘッダー領域長＞"
     packetheader="＜パケットヘッダー領域長＞"
     packetoffset="＜パケットデータサイズ領域オフセット＞"
     packetlength="＜パケットデータサイズ領域長＞"
     timeoffset="＜タイムスタンプ領域オフセット＞"/>
    <command path="＜コマンドパス名＞"
     parameter="＜コマンドパラメーター＞"/>
  </input>
  <output unit="＜最大出力単位＞">
    <record name="＜レコード名＞" type=" {REQUEST | RESPONSE} ">
      <field name="＜フィールド名＞"/>
    </record>
```

```
  </output>
</HttpPacketInputConnectorDefinition>
```

(2)　定義の詳細

HttpPacketInputConnectorDefinition タグ（全体情報の定義）

HTTP パケット入力コネクター定義の全体情報を定義します。この定義は必ず 1 個だけ記述します。

input タグ（読み込み定義）

HTTP パケットの入出力情報を定義します。この定義は必ず 1 個だけ記述します。

buffersize="＜入出力バッファーサイズ＞"

入力バッファーに格納できる HTTP パケット（単位：パケット），および出力バッファーに格納できる共通形式レコード（単位：レコード）の最大数を 1～12288 の整数で指定します。省略した場合，4096 が仮定されます。

assemblingtime="＜分割パケット組み立て時間＞"

分割パケット組み立て時間（単位：ミリ秒）を 1～5000 の整数で指定します。省略した場合，2000 が仮定されます。

分割パケットとは，最大セグメント長（MSS：Maximum Segment Size）に従って TCP 階層で分割されたパケットデータのことです。HTTP パケット入力コネクターが取得したデータが分割パケットの場合は，TCP プロトコルヘッダーの情報を基にパケットを組み立てます。分割パケット組み立て時間とは，新たな分割パケットが到着してから，到着済みのパケットと連結して組み立てるまでに掛かる時間のことで，分割パケットが連結されるごとにリセットしてカウントされます。なお，IP 階層で分割されるパケットデータの組み立てはしません。

分割パケットの組み立てで，ここで指定した時間を超えた場合は，組み立て中の分割パケットは破棄されます。また，組み立てに使用した分割パケットの最新の時刻情報は，タイムスタンプとして使用されます。

packetdata タグ（HTTP パケット定義）

取得する HTTP パケットのフォーマットを定義します。この定義は必ず 1 個だけ記述します。

globalheader="＜グローバルヘッダー領域長＞"

グローバルヘッダー領域の長さ（単位：バイト）を 1～128 の整数で指定します。省略した場合，24 が仮定されます。

packetheader="＜パケットヘッダー領域長＞"

パケットヘッダー領域の長さ（単位：バイト）を 9～128 の整数で指定します。省略した場合，16 が仮定されます。

packetoffset="＜パケットデータサイズ領域オフセット＞"

パケットヘッダーの先頭からパケットデータサイズ領域までのオフセット（単位：バイト）を 0～127 の整数で指定します。省略した場合，8 が仮定されます。

packetlength="＜パケットデータサイズ領域長＞"

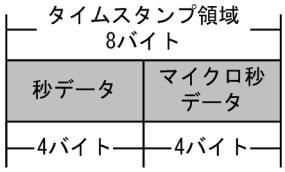
パケットデータサイズ領域の長さ（単位：バイト）を 1～4 の整数で指定します。省略した場合，4 が仮定されます。

timeoffset="＜タイムスタンプ領域オフセット＞"

パケットヘッダーの先頭からタイムスタンプ領域までのオフセット（単位：バイト）を 0～120 の整数で指定します。省略した場合，0 が仮定されます。

タイムスタンプ領域とは，パケットアナライザーが HTTP パケットをキャプチャーしたときのタイムスタンプを格納している，パケットヘッダーの領域です。タイムスタンプ領域のフォーマットを次の図に示します。

図 9-2　タイムスタンプ領域のフォーマット



command タグ（コマンド定義）

使用するパケットアナライザーの起動コマンドの情報を定義します。この定義は必ず 1 個だけ記述します。

Stream Data Platform - AF では，パケットアナライザーとして Windows の場合は WinDump，Linux の場合は tcpdump を使用できます。

WinDump，または tcpdump を使用する場合に，この定義で指定する内容については，「(4) WinDump を使用する場合の command タグの指定内容」，または「(5)　tcpdump を使用する場合の command タグの指定内容」を参照してください。

path="＜コマンドパス名＞"

パケットアナライザーの起動コマンドの絶対パス名を 1～100 文字の半角文字（ASCII コードの 32～126）で指定します。

parameter="＜コマンドパラメーター＞"

パケットアナライザーの起動コマンドに渡すパラメーターを 1～100 文字の半角文字（ASCII コードの 32～126）で指定します。

output タグ（出力定義）

HTTP パケット入力コネクターが HTTP パケットを共通形式レコードに変換する際の，共通形式レコードの出力情報を定義します。この定義は必ず 1 個だけ記述します。

unit="＜最大出力単位＞"

共通形式レコードの最大出力単位（単位：レコード）を 1～1000 の整数で指定します。省略した場合，100 が仮定されます。

record タグ（レコード定義）

共通形式レコードのレコード情報を定義します。この定義は 1～2 個記述します。

name="＜レコード名＞"

共通形式レコードのレコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は省略できません。レコード名は，record タグ内で一意となるように指定してください。

type="｛REQUEST ｜ RESPONSE｝"

共通形式レコードの種類を指定します。この属性は省略できません。

指定できる値を次に示します。

- "REQUEST"

  リクエストレコードです。HTTP プロトコル通信で，クライアントからホストへのリクエスト送信時に発生したデータを格納する共通形式レコードです。

- "RESPONSE"

  レスポンスレコードです。HTTP プロトコル通信で，ホストからクライアントへのレスポンス送信時に発生したデータを格納する共通形式レコードです。

field タグ（フィールド定義）

共通形式レコードを構成するフィールドの情報を定義します。この定義は 1～16 個記述します。

name="＜フィールド名＞"

フィールド名を指定します。

フィールド名には，HTTP パケットから抽出したいデータの識別子を指定します。ここで指定した識別子に対応したデータが，各フィールドの値に格納されます。

フィールド名は，レコード定義内で一意となるように指定します。

フィールド名に指定できる識別子は，record タグの type 属性で指定する共通形式のレコードの種類によって異なります。フィールド名に指定できる識別子については,「(6)　フィールド名に指定できる識別子」を参照してください。

抽出されたデータは，Java のデータ型に変換されてフィールド値に格納されます。フィールド値に格納されるデータの Java のデータ型については，「(7)　フィールド値に格納されるデータの Java のデータ型」を参照してください。

(3)　記述例

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
 xmlns:hpicon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
HttpPacketInputConnectorDefinition">
<!-- 途中略 -->

<!-- 入力用CB定義 -->
<cb:InputCBDefinition
 class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.packetinput.HttpPacketInputCBImpl"
name="inputer2">
  <!-- HTTPパケット入力コネクター定義 -->
  <hpicon:HttpPacketInputConnectorDefinition>
    <!-- 読み込み定義 -->
    <hpicon:input buffersize="096" assemblingtime="2000">
      <!-- HTTPパケット定義 -->
      <hpicon:packetdata globalheader="24" packetheader="16" packetoffset="8"
       packetlength="4" timeoffset="0"/>
      <!-- コマンド定義 -->
      <hpicon:command path="C:¥Program Files¥WinDump¥WinDump.exe"
       parameter=" -i 1 -s 2048 -w - -n &amp;quot;tcp port 80 and host 133.145.224.19&amp;quot;"/>
    </hpicon:input>
    <!-- 出力定義 -->
    <hpicon:output unit="100">
      <!-- レコード定義 -->
      <hpicon:record name="RECORD1" type="REQUEST">
        <!-- フィールド定義 -->
        <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)　WinDump を使用する場合の command タグの指定内容

パケットアナライザーとして WinDump を使用する場合に，command タグで指定する内容について説明します。なお，ここでは，WinDump のバージョン 3.9.5 を使用する場合の例を示します。WinDump の起動コマンドの詳細については，WinDump のドキュメントを参照してください。

HTTP パケット入力コネクターでは，次の書式で記述された WinDump の起動コマンド（WinDump.exe）をサポートします。

WinDump.exe△-i△＜ネットワークデバイスの番号＞△-s△＜内部バッファーサイズ＞△-w△-△-n△"tcp△port△＜ポート番号＞△and△host△＜IPアドレス＞"

書式の「△」は半角スペースを表します。command タグの parameter 属性では，半角スペースを省略しないで記述してください。

書式に示した値について説明します。

WinDump.exe

WinDump の起動コマンドです。command タグの path 属性で，WinDump.exe のパスを絶対パスで指定してください。

-i△＜ネットワークデバイスの番号＞

解析対象のコンピュータに接続されているネットワークデバイスの番号を指定するオプションです。このオプションと値は，command タグの parameter 属性で指定します。

-s△＜内部バッファーサイズ＞

キャプチャーしたパケットデータを格納する内部バッファーのサイズ（単位：バイト）を指定します。HTTP では TCP のパケットサイズ以上を指定する必要があり，通常は，2,048 バイトを指定すれば十分です。このオプションと値は，command タグの parameter 属性で指定します。

-w△-

キャプチャーしたパケットの出力先として，ファイル，または標準出力を指定するオプションです。HTTP パケット入力コネクターを使用する場合には，パケットを標準出力するため，「-w△-」と指定します。このオプションと値は，command タグの parameter 属性で指定します。

-n△"tcp△port△＜ポート番号＞△and△host△＜ IP アドレス＞"

パケットキャプチャーライブラリー（libpcap）のフィルター書式でキャプチャーフィルターを指定するオプションです。HTTP パケット入力コネクターを使用する場合には，HTTP プロトコルで使用するポート番号と，解析対象のコンピュータの IP アドレスを指定します。このオプションと値は，command タグの parameter 属性で指定します。

なお，ダブルクォーテーション（"）は特殊文字に該当するため，「"」と置換して記述してください。

(5)　tcpdump を使用する場合の command タグの指定内容

パケットアナライザーとして tcpdump を使用する場合に，command タグで指定する内容について説明します。なお，ここでは，tcpdump のバージョン 3.9.4 を使用する場合の例を示します。tcpdump の起動コマンドの詳細については，tcpdump のドキュメントを参照してください。

HTTP パケット入力コネクターでは，次の書式で記述された tcpdump の起動コマンドをサポートします。

```
/usr/sbin/tcpdump△-i△＜LANインタフェース名＞△-s△＜内部バッファーサイズ＞△-w△-△-n△tcp△port△＜ポート番号＞△and△host△＜IPアドレス＞
```

書式の「△」は半角スペースを表します。command タグの parameter 属性では，半角スペースを省略しないで記述してください。

書式に示した値について説明します。

tcpdump

tcpdump の起動コマンドです。command タグの path 属性で，tcpdump のパスを絶対パスで指定してください。

-i△＜ LAN インタフェース名＞

解析対象のコンピュータに接続されている LAN インタフェース名を指定するオプションです。このオプションと値は，command タグの parameter 属性で指定します。

-s△＜内部バッファーサイズ＞

キャプチャーしたパケットデータを格納する内部バッファーのサイズ（単位：バイト）を指定します。HTTP では TCP のパケットサイズ以上を指定する必要があり，通常は，65,535 バイトを指定します。このオプションと値は，command タグの parameter 属性で指定します。

-w△-

キャプチャーしたパケットの出力先として，ファイル，または標準出力を指定するオプションです。HTTP パケット入力コネクターを使用する場合には，パケットを標準出力するため，「-w△-」と指定します。このオプションと値は，command タグの parameter 属性で指定します。

-n△tcp△port△＜ポート番号＞△and△host△＜ IP アドレス＞

パケットキャプチャーライブラリー（libpcap）のフィルター書式でキャプチャーフィルターを指定するオプションです。HTTP パケット入力コネクターを使用する場合には，HTTP プロトコルで使用するポート番号と，解析対象のコンピュータの IP アドレスを指定します。このオプションと値は，command タグの parameter 属性で指定します。

(6)　フィールド名に指定できる識別子

field タグの name 属性のフィールド名として指定できる識別子を次の表に示します。

表 9-9　field タグの name 属性のフィールド名として指定できる識別子

項番	識別子	データ	内容	プロトコル	レコードの種類ごとの指定可否	
					リクエスト	レスポンス
1	TIME	時刻※1	パケットデータが到着した時刻	－	○	○
2	PACKET_LENGTH	パケットサイズ※2	パケットデータの長さ（単位：バイト）	－	○	○
3	SEND_MAC	送信元 MAC アドレス	パケット送信元の MAC アドレス	Ethernet	○	○
4	SEND_IP	送信元 IP アドレス	パケット送信元の IP アドレス	IP	○	○
5	RECEIVE_IP	送信先 IP アドレス	パケット送信先の IP アドレス	IP	○	○
6	SEND_PORT	送信元ポート番号	パケット送信元のポート番号	TCP	○	○
7	RECEIVE_PORT	送信先ポート番号	パケット送信先のポート番号	TCP	○	○
8	MESSAGE_TYPE	メッセージ種別	Request，Response の種別	HTTP	○	○
9	METHOD_NAME	メソッド情報	GET，POST などのメソッド情報	HTTP	○	×
10	TARGET_URI	URI 情報※3	アクセス先の URI 情報	HTTP	○	×
11	REFERER	Referer※3	リンク元の URI 情報	HTTP	○	×

項番	識別子	データ	内容	プロトコル	レコードの種類ごとの指定可否	
					リクエスト	レスポンス
12	COOKIE	Cookie※3　※4	クッキー情報※5	HTTP	○	○
13	STATUS_CODE	ステータスコード	要求の処理結果	HTTP	×	○
14	CONNECTION	Connection	接続の永続性情報	HTTP	○	○
15	CONTENT_LENGTH	Content-Length	コンテンツの長さ（単位：バイト）	HTTP	○	○
16	CONTENT_TYPE	Content-Type	コンテンツの種類	HTTP	○	○
17	MESSAGE_BODY	メッセージ・ボディ※3	実データ	HTTP	○※6	×

（凡例）

○：識別子を指定できます。

×：識別子を指定できません。

－：該当しません。

注※1

時刻は，パケットヘッダーのタイムスタンプから取得します。

注※2

パケットサイズは，HTTP メッセージの開始行，ヘッダーサイズ，および HTTP ヘッダー"Content-Length"が参照する値の合計値です。"Content-Length"がない場合，"Content-Length"が参照する値は 0 とします。

注※3

パーセントエンコードされている文字列は，UTF-8 でデコードします。

注※4

リクエストレコードとレスポンスレコードで Cookie データの取得方法が異なります。

リクエストレコードの場合は，HTTP ヘッダー"Cookie"が参照するデータを取得します。

レスポンスレコードの場合は，HTTP ヘッダー"Set-Cookie"が参照するデータを取得します。HTTP ヘッダー"Cookie2"と"Set-Cookie2"が参照するデータは取得しません。

注※5

クッキー情報には複数のクッキーが含まれている場合があります。それぞれのクッキーをフィールドとして扱いたい場合は，マッピング定義の map タグの function 属性で regexsubstring を指定して，正規表現の文字列を取得するようにしてください。

注※6

メッセージ・ボディは，メソッド情報が POST，Content-Type のメディアタイプが text で（サブタイプの値は問わない），かつ Content-Length が存在する場合だけ取得できます。

(7)　フィールド値に格納されるデータの Java のデータ型

field タグの name 属性で指定した識別子に従って抽出された各プロトコルデータは，次の表に示す Java のデータ型に変換されます。Java のデータ型への変換の際に，プロトコルデータの値が値の範囲の上限を超えている場合は，上限値までがフィールド値に格納されます。指定した識別子に対応するデータがプロトコルデータにない場合，フィールド値には，String 型のときは空文字，Integer 型のときは-1 が格納されます。

表 9-10　フィールド値に格納されるデータの Java のデータ型

項番	データ	プロトコル	Java のデータ型	値の範囲
1	時刻※	−	Timestamp	1970/01/01 00:00:00.000000 ～ 2261/12/31 23:59:59.999999
2	パケットサイズ	−	Integer	0～2,147,483,647
3	送信元 MAC アドレス	Ethernet	String	17 文字（00:00:00:00:00:00 ～ FF:FF:FF:FF:FF:FF）
4	送信元ポート番号	TCP	Integer	0～65,535
5	送信先ポート番号	TCP	Integer	
6	送信元 IP アドレス	IP	String	• IPv4 の場合 7～15 文字（0.0.0.0 ～ 255.255.255.255） • IPv6 の場合 40 文字（0000:0000:0000:0000:0000:0000:0000:0000 ～ FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF）
7	送信先 IP アドレス	IP	String	
8	データ種別	HTTP	String	7～8 文字（Request または Response）
9	メソッド情報	HTTP	String	1～127 文字（GET，CONNECT など）
10	URI 情報	HTTP	String	1～255 文字
11	Referer	HTTP	String	
12	Cookie	HTTP	String	1～4,096 文字
13	ステータスコード	HTTP	String	3 文字（200，404 など）
14	Connection	HTTP	String	1～127 文字（close または Keep-Alive）
15	Content-Length	HTTP	Integer	0～2,147,483,647
16	Content-Type	HTTP	String	3～255 文字
17	メッセージ・ボディ	HTTP	String	0～2,048 文字

（凡例）

−：該当しません。

注※

フィールドに時刻データを指定する場合は，クエリ定義ファイルの CQL データ型（TIMESTAMP）の有効けた数も合わせて 6 けた以上を指定するようにしてください。

9.10.3　ファイル出力コネクター定義

ファイル出力コネクター定義（FileOutputConnectorDefinition タグ）は，出力用 CB 定義（OutputCBDefinition タグ）の子要素として定義します。

なお，ファイルの出力の処理については，「11.6　ファイルへの出力」を参照してください。

(1)　記述形式

```
<FileOutputConnectorDefinition>
  <output compositionType=”{WRAP_AROUND｜ANTI_WRAP_AROUND}”
   maxNumber=”＜出力ファイル最大数＞”
   maxSize=”＜出力ファイル最大サイズ＞”/>
  <file path=”＜出力パス名＞”
   prefix=”＜プレフィックス名＞”
   addDate=”{ON｜OFF}”
   extension=”＜拡張子＞”/>
</FileOutputConnectorDefinition>
```

(2)　定義の詳細

FileOutputConnectorDefinition タグ（全体情報の定義）

ファイル出力コネクター定義の全体情報を定義します。この定義は必ず 1 個だけ記述します。

output タグ（書き込み定義）

出力ファイルの書き込み処理の情報を定義します。この定義は必ず 1 個だけ記述します。

compositionType="{WRAP_AROUND ｜ ANTI_WRAP_AROUND}"

出力ファイル構成を指定します。ラップアラウンドと非ラップアラウンドがあります。どちらの場合も，出力先ファイル名と同名のファイルが存在する場合には上書きします。省略した場合，WRAP_AROUND が仮定されます。

指定できる値を次に示します。

- WRAP_AROUND

  ラップアラウンド構成でファイルを出力します。

  maxNumber 属性で指定したファイル数に達するまでレコードを出力し，指定したファイル数を超えると，最初に作成した出力ファイルに上書きして出力します。ディスク容量の圧迫によるシステムの動作不正を防ぐため，通常の運用では WRAP_AROUND を指定してください。

  また，WRAP_AROUND を指定する場合は，file タグの addDate 属性に OFF を指定してください。

- ANTI_WRAP_AROUND

  非ラップアラウンド構成でファイルを出力します。

  maxNumber 属性で指定したファイル数に達するまでレコードを出力し，指定したファイル数を超えると，標準提供アダプターはレコードの出力を停止します。

maxNumber="＜出力ファイル最大数＞"

出力先のファイルの最大数を 1～100000 の整数で指定します。省略した場合，256 が仮定されます。

maxSize="＜出力ファイル最大サイズ＞"

1 ファイルに書き込むレコードの行数を 1～10000 の整数で指定します。この属性は省略できません。

file タグ（出力ファイルの定義）

出力ファイルの情報を定義します。この定義は必ず 1 個だけ記述します。

出力ファイルは，ここで定義した内容に従って生成されます。出力ファイル名の生成規則については，「(4)　出力ファイル名の生成規則」を参照してください。

path="＜出力パス名＞"

ファイルを出力するディレクトリのパス名を 1～160 文字の半角文字で指定します。ここで指定したディレクトリが存在しない場合は，そのディレクトリを生成します。この属性は省略できません。

この属性に指定できる文字を次に示します。

半角英数字，「!」，「#」，「$」，「%」，「&」，「'」，「(」，「)」，「=」，「~」，「`」，「{」，「}」，「+」，「-」，「^」，「.」，「_」，「@」，「¥」，「/」，「:」，半角スペース

prefix="＜プレフィックス名＞"

ファイル名のプレフィックス（接頭辞）を 1～60 文字の半角文字で指定します。path 属性で指定した出力パス名からの相対パスで指定します。この属性は省略できません。

「¥」，「/」，および「:」は指定できません。

この属性に指定できる文字を次に示します。

半角英数字，「!」，「#」，「$」，「%」，「&」，「'」，「(」，「)」，「=」，「~」，「`」，「{」，「}」，「+」，「-」，「^」，「.」，「_」，「@」，半角スペース

なお，予約デバイス名（NUL など）をファイル名に指定しないでください。予約デバイス名を指定した場合，動作は保証されません。予約デバイス名を指定したファイルをオープンしようとした場合，メッセージ KFSP46211-E が出力されてエラーとなります。

addDate="｛ON | OFF｝"

出力ファイル名に日時（hhmmss_MMDD_YYYY 形式）を付与するかどうかを指定します。省略した場合，OFF が仮定されます。

指定できる値を次に示します。

- ON

  日時を付与します。

- OFF

  日時を付与しません。

extension="＜拡張子＞"

出力ファイル名に付与する拡張子を指定します。省略した場合，拡張子なしのファイルが出力されます。

この属性に指定できる文字を次に示します。

半角英数字，「!」，「#」，「$」，「%」，「&」，「'」，「(」，「)」，「=」，「~」，「`」，「{」，「}」，「+」，「-」，「^」，「.」，「_」，「@」，半角スペース

(3)　記述例

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
 xmlns:focon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
FileOutputConnectorDefinition">
<!-- 途中略 -->

<!-- 出力用CB定義 -->
<cb:OutputCBDefinition class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileOutputCBImpl"
name="outputer1">
  <!-- ファイル出力コネクター定義 -->
  <focon:FileOutputConnectorDefinition>
    <!-- 書き込み定義 -->
    <focon:output compositionType="WRAP_AROUND" maxNumber="100" maxSize="10"/>
    <!-- 出力ファイルの定義 -->
    <focon:file path="D:¥home¥" prefix="out" addDate="OFF" extension="csv"/>
  </focon:FileOutputConnectorDefinition>]
</cb:OutputCBDefinition>
```

(4)　出力ファイル名の生成規則

出力ファイルは，file タグで指定した内容に従って生成されます。

出力されるファイル名の形式を次に示します。

’[’プレフィックス名’]’　【’[’日時_’]’】’[’通番’]’【.拡張子】

（凡例）

[ ]：この記号で囲まれた項目は，この記号を省略しないでそのまま記述することを示します。

【】：この記号で囲まれた項目は省略できることを示します。

ファイル名の値について説明します。

プレフィックス名

prefix 属性で指定した値が使用されます。

日時

日時は，hhmmss_MMDD_YYYY 形式です。

- hh：時（00～23）
- mm：分（00～59）
- ss：秒（00～59）
- MM：月（01～12）
- DD：日（01～31）
- YYYY：年（西暦 4 けた表記）

出力ファイル名に日時を付与するかどうかは，addDate 属性で指定します。addDate 属性に ON を指定した場合は日時が付与され，OFF を指定した場合は日時が付与されません。

通番

通番は，出力ファイルが生成された順番になります。通番は，1 から output タグの maxNumber 属性で指定した値までです。標準提供アダプターを再起動した場合も同様に，通番は 1 からになります。

出力ファイル名に日時を付与する場合，年月日が変更された場合も，継続して通番が付与されます。通番の例を次に示します。

（例）

- path 属性の指定値：D:¥home¥
- prefix 属性の指定値：a
- addDate 属性の指定値：ON
- extension 属性の指定値：csv

この例の場合のファイルの生成番号と生成されるファイル名を次の表に示します。

生成番号	生成されるファイル名
1	D:¥home¥a235015_0706_2009_1.csv
2	D:¥home¥a235959_0706_2009_2.csv
3	D:¥home¥a000130_0707_2009_3.csv

なお，output タグの compositionType 属性で WRAP_AROUND を指定し，かつ addDate 属性で ON を指定している場合，日時が付与されるため，通番が同じでも別ファイルとなります。

拡張子

extension 属性に指定した値が使用されます。

9.10.4　ダッシュボード出力コネクター定義

ダッシュボード出力コネクター定義（DashboardOutputConnectorDefinition タグ）は，出力用 CB 定義（OutputCBDefinition タグ）の子要素として定義します。

なお，ダッシュボードへの出力の処理については，「11.7　ダッシュボードへの出力」を参照してください。

(1)　記述形式

```
<DashboardOutputConnectorDefinition
 MaxNum=”＜最大レコード保持数＞”
 Record=”＜レコード名＞”
 ReadRecordRemoveFlag=”｛ON｜OFF｝”>
  <RecordHoldTime
   DateReference=”｛CURRENT_DATE｜LAST_UPDATE｝”
   RecordTime=”＜レコード保持期間＞”
   DateFieldPosition=”＜時刻フィールド番号＞”/>
  <DataProcessingDefinition
   Name=”｛HistoryRecorder｜NoDataProcessing｝”>
    <HistoryRecorder/>
    <NoDataProcessing/>
  </DataProcessingDefinition>
</DashboardOutputConnectorDefinition>
```

(2)　定義の詳細

DashboardOutputConnectorDefinition タグ（全体情報の定義）

ダッシュボード出力コネクター定義の全体情報を定義します。この定義は必ず 1 個だけ記述します。

MaxNum="＜最大レコード保持数＞"

レコード保持領域内でのレコードの最大保持数を 1～10000 の整数で指定します。省略した場合，1000 が仮定されます。

レコード数がこの属性で指定した値を超えると，古いレコードから順番に削除します。

Record="＜レコード名＞"

ダッシュボード出力の対象となるレコードを指定します。この属性は省略できません。

ReadRecordRemoveFlag="｛ON ｜ OFF｝"

Dashboard Server が取得したレコードをダッシュボード出力コネクターから削除するかどうかを指定します。省略した場合，OFF が仮定されます。

指定できる値を次に示します。

- ON

  レコードを削除します。

- OFF

  レコードを削除しません。

RecordHoldTime タグ（レコード保持期間の定義）

保持期間によってダッシュボード出力コネクターのレコードを削除する場合に，レコードを保持する期間を定義します。この定義を省略した場合，保持期間によるレコードの削除は行われません。

なお，レコードが削除されるのは，次の条件を満たす場合です。

（DateReference属性の基準時刻-RecordTime属性の保持期間）
＞DateFieldPosition属性のフィールドの時刻

例えば，基準時刻が 10:00:10，保持期間が 5 秒，フィールドの時刻が 10:00:04 だった場合，そのレコードは削除されます。

DateReference="｛CURRENT_DATE ｜ LAST_UPDATE｝"

レコードを削除するための基準時刻を指定します。この属性は省略できません。

指定できる値を次に示します。

- CURRENT_DATE

  システムの現在の時刻を基準時刻とします。

- LAST_UPDATE

  最後にタプルを受信した時刻を基準時刻とします。

RecordTime="＜レコード保持期間＞"

レコードの保持期間（単位：秒）を 0～86400 の整数で指定します。この属性は省略できません。

DateFieldPosition="＜時刻フィールド番号＞"

レコード内で時刻が設定されているフィールドの番号を 1～3000 の整数で指定します。この属性は省略できません。

DataProcessingDefinition タグ（データの加工処理の定義）

ダッシュボード出力コネクターが出力するダッシュボード表示用データの加工処理について定義します。この定義を省略した場合，データの加工処理は行われません。

Name="｛HistoryRecorder ｜ NoDataProcessing｝"

ダッシュボード表示用データの加工処理を行うかどうかを指定します。この属性は省略できません。

指定できる値を次に示します。

- HistoryRecorder

  加工処理を行います。グラフを表示したい場合には，HistoryRecorder を指定します。加工処理を行う場合，最大レコード保持数，およびレコード保持期間の定義に従うすべてのレコードをダッシュボード表示用データ内に保持します。

  HistoryRecorder を指定した場合，DataProcessingDefinition タグの下に空の HistoryRecorder タグを記述します。

- NoDataProcessing

  加工処理を行いません。加工処理が不要な場合には，NoDataProcessing を指定します。加工処理を行わない場合，ダッシュボード出力コネクターからレコードを取得するたびに，それまでのダッシュボード表示用データがすべて削除されます。

  NoDataProcessing を指定した場合，DataProcessingDefinition タグの下に空の NoDataProcessing タグを記述します。

HistoryRecorder タグ（履歴保持定義）

DataProcessingDefinition タグの Name 属性で HistoryRecorder を指定した場合に，DataProcessingDefinition タグの下に空の HistoryRecorder タグを記述します。

Name 属性で NoDataProcessing を指定した場合には，このタグを記述する必要はありません。

NoDataProcessing タグ（無加工出力定義）

DataProcessingDefinition タグの Name 属性で NoDataProcessing を指定した場合に，DataProcessingDefinition タグの下に空の NoDataProcessing タグを記述します。

Name 属性で HistoryRecorder を指定した場合には，このタグを記述する必要はありません。

(3)　記述例

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
xmlns:docon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
```

```
DashboardOutputConnectorDefinition">
<!-- 途中略 -->

<!-- 出力用CB定義 -->
<cb:OutputCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.dashboard.DashboardOutputCBImpl"
name="outputer2">
  <!-- ダッシュボード出力コネクター定義 -->
  <docon:DashboardOutputConnectorDefinition MaxNum="1000" Record="RECORD2"
   ReadRecordRemoveFlag="OFF">
    <!-- レコード保持期間の定義 -->
    <docon:RecordHoldTime DateReference="LAST_UPDATE"
     RecordTime="300" DateFieldPosition="1"/>
    <!-- データの加工処理の定義 -->
    <docon:DataProcessingDefinition Name="NoDataProcessing">
      <docon:NoDataProcessing/>
    </docon:DataProcessingDefinition>
  </docon:DashboardOutputConnectorDefinition>
</cb:OutputCBDefinition>
```

9.11　アダプター構成定義ファイルのデータ編集用 CB 定義

ここでは，アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）のデータ編集用 CB 定義について説明します。

データ編集用 CB 定義は，「9.9.3　編集用 CB 定義」で説明した編集用 CB 定義（DataEditCBDefinition タグ）の子要素として定義します。

データ編集用 CB 定義の一覧を次の表に示します。

表 9-11　データ編集用 CB 定義の一覧

項番	データ編集用 CB 定義	親要素	参照先
1	フォーマット変換定義 （FormatDefinition タグ）	編集用 CB 定義	9.11.1
2	マッピング定義 （MappingDefinition タグ）		9.11.2
3	フィルター定義 （FilterDefinition タグ）		9.11.3
4	レコード抽出定義 （RecordExtractionDefinition タグ）		9.11.4

9.11.1　フォーマット変換定義

フォーマット変換定義（FormatDefinition タグ）は，「9.9.3　編集用 CB 定義」で説明した編集用 CB 定義（DataEditCBDefinition タグ）の子要素として定義します。

なお，フォーマット変換の処理については，「11.2.2(3)　フォーマット変換」，または「11.6.2(4)　フォーマット変換」を参照してください。

(1)　記述形式

```
<FormatDefinition ioType=" {INPUT | OUTPUT} ">
  <common>
    <unmatchedFormat> {IGNORE | WARNING | ERROR}
    </unmatchedFormat>
    <format timestampformat="＜形式番号＞" year="＜開始年＞" month="＜開始月＞"/>
  </common>
  <records>
    <record name="＜レコード名＞" exp="＜レコード構成＞"
      timestampformat="＜形式番号＞" year="＜開始年＞" month="＜開始月＞">
      <field name="＜フィールド名＞"
        type=" {INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT | DOUBLE | STRING | DATE | TIME |
TIMESTAMP} "
        pattern="＜パターン＞"/>
    </record>
  </records>
</FormatDefinition>
```

(2)　定義の詳細

FormatDefinition タグ（全体情報の定義）

フォーマット変換定義の全体情報を定義します。この定義は必ず 1 個だけ記述します。

ioType="｛INPUT ｜ OUTPUT｝"

この定義で指定する標準提供アダプターの種類を指定します。この属性は省略できません。

指定できる値を次に示します。

- INPUT

  入力アダプターでフォーマット変換を定義する場合に指定します。

- OUTPUT

  出力アダプターでフォーマット変換を定義する場合に指定します。

common タグ（共通定義）

フォーマット変換定義の共通情報を定義します。この定義は必ず 1 個だけ記述します。

unmatchedFormat タグ（変換パターンに一致しないレコードの定義）

このタグでは，フォーマット変換のパターンに一致しないレコードを検知したときの対処を指定します。

この定義は 1 個だけ記述できます。また，この定義は省略できます。省略した場合，ERROR が仮定されます。

指定できる値を次に示します。

- IGNORE

  検知した内容を無視して，標準提供アダプターの処理を続行します。

- WARNING

  警告メッセージを出力して，標準提供アダプターの処理を続行します。

- ERROR

  エラーメッセージを出力して，標準提供アダプターを停止します。

format タグ（フォーマット定義）

データ型種別で扱う TIMESTAMP 型の文字列表現について定義します。この定義は 1 個だけ記述できます。また，TIMESTAMP 型の文字列表現を使用しない場合，この定義は省略できます。

timestampformat="＜形式番号＞"

TIMESTAMP 型の形式番号を 1～4 の整数で指定します。省略した場合，1 が仮定されます。

なお，format タグの timestampformat 属性を指定し，かつ records タグの timestampformat 属性を指定した場合には，records タグでの指定内容が有効になります。

形式番号として指定できる値の意味を次の表に示します。

形式番号	文字列表現の形式	形式
1（デフォルト値）	年-月-日 時:分:秒.ミリ秒（1～9 けた）	yyyy-MM-dd HH:mm:ss.fffffffff
2※1	月名※2 日 時:分:秒	MMM dd HH:mm:ss
3	年/月/日 時:分:秒.ミリ秒（3 けた）	yyyy/MM/dd HH:mm:ss.SSS
4	日/月名※2/年:時:分:秒	dd/MMM/yyyy:HH:mm:ss

注※1

形式番号 2 は，形式に年がありません。このため，入力アダプターで形式番号 2 を指定する場合には，year 属性および month 属性で開始年月を指定してください。出力アダプターで形式番号 2 を指定する場合には，year 属性および month 属性は指定できません。

なお，year 属性および month 属性を指定しない場合，標準提供アダプターの起動時点のシステムの年月が TIMESTAMP 型の開始年月となります。

注※2

英字 3 けたの英語の月名です。Jan（1 月），Feb（2 月），Mar（3 月），Apr（4 月），May（5 月），Jun（6 月），Jul（7 月），Aug（8 月），Sep（9 月），Oct（10 月），Nov（11 月），Dec（12 月）です。

year="＜開始年＞"

timestampformat 属性で 2 を指定した場合，この属性で TIMESTAMP 型の開始年を指定します。年は，1970～2261 の整数で指定します。この属性は，出力アダプターの場合には指定できません。

TIMESTAMP 型の開始年月の指定については，「(4)　TIMESTAMP 型の開始年月の指定」を参照してください。

month="＜開始月＞"

timestampformat 属性で 2 を指定した場合，この属性で TIMESTAMP 型の開始月を指定します。年は，1～12 の整数で指定します。この属性は，出力アダプターの場合には指定できません。

TIMESTAMP 型の開始年月の指定については，「(4)　TIMESTAMP 型の開始年月の指定」を参照してください。

records タグ（レコード群定義）

レコード群の情報を定義します。この定義は必ず 1 個だけ記述します。この定義は省略できません。

record タグ（レコード定義）

各レコードの情報を定義します。この定義は 10 個まで記述できます。この定義は省略できません。

name="＜レコード名＞"

レコードの情報を識別するための名称を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

レコード名は，records タグ内で一意となるように指定してください。この属性は省略できません。

exp="＜レコード構成＞"

レコードを構成するフィールドを 1～1,000,000 文字で指定します。

レコード構成内に，このレコード固有の区切り文字を指定できます。また，フィールドごとに異なる区切り文字を指定することもできます。この属性は省略できません。

レコード構成の記述形式については，「(5)　レコード構成の記述形式」を参照してください。

timestampformat="＜形式番号＞"

TIMESTAMP 型の形式番号を 1～4 の整数で指定します。省略した場合，format タグの timestampformat 属性の指定値が仮定されます。

なお，format タグの timestampformat 属性を指定し，かつ records タグの timestampformat 属性を指定した場合には，records タグでの指定内容が有効になります。

形式番号として指定できる値の意味については，format タグの timestampformat 属性の説明を参照してください。

year="＜開始年＞"

timestampformat 属性で 2 を指定した場合，この属性で TIMESTAMP 型の開始年を指定します。年は，1970～2261 の整数で指定します。この属性は，出力アダプターの場合には指定できません。

TIMESTAMP 型の開始年月の指定については，「(4)　TIMESTAMP 型の開始年月の指定」を参照してください。

month="＜開始月＞"

timestampformat 属性で 2 を指定した場合，この属性で TIMESTAMP 型の開始月を指定します。年は，1～12 の整数で指定します。この属性は，出力アダプターの場合には指定できません。

TIMESTAMP 型の開始年月の指定については，「(4)　TIMESTAMP 型の開始年月の指定」を参照してください。

field タグ（フィールド定義）

フィールド情報を定義します。この定義は 3,000 個まで記述できます。この定義は省略できません。

name="＜フィールド名＞"

フィールドの情報を識別するための名称を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

フィールド名は，records タグ内で一意となるように指定してください。この属性は省略できません。

type=" {INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT | DOUBLE | STRING | DATE | TIME | TIMESTAMP} "

フィールドの Java のデータ型に対応するデータ型を指定します。この属性は省略できません。

この属性に指定する値と，対応する Java のデータ型および CQL のデータ型については，「(6) type 属性の指定値と，対応する Java のデータ型および CQL のデータ型」を参照してください。

pattern="＜パターン＞"

type 属性に STRING を指定した場合，フィールド値のパターンを正規表現で指定します。正規表現の解析には，java.util.regex.Pattern クラスを使用します。このため，正規表現は，java.util.regex.Pattern クラスがサポートする正規表現の範囲で記述してください。「$」は使用できません。

なお，type 属性に STRING 以外を指定している場合，または出力アダプターで定義している場合には，この属性は指定できません。指定した場合は，エラーとなります。

パターンには定数が指定できます。パターンに定数が指定された場合，フィールド値を定数として扱います。

この属性を省略した場合のデータ型種別の表記規則については，「(7)　データ型種別の表記規則」を参照してください。

(3)　記述例

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
xmlns:form="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
FormatDefinition">
<!-- 途中略 -->

<!-- 編集用CB定義 -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.formattranslate.InputFormatTranslatorCB
Impl" name="editor1">
  <!-- フォーマット変換定義 -->
  <form:FormatDefinition ioType="INPUT">
    <form:common/>
      <!-- レコード群定義 -->
      <form:records>
        <form:record name="R1" exp="($_F1),($_F2),($_F3),($_F4),($_F5)">
        <!-- フィールド定義 -->
          <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)　TIMESTAMP 型の開始年月の指定

入力アダプターで format タグの timestampformat 属性に形式番号 2 を指定し，format タグの year 属性および month 属性を指定した場合には，次のように TIMESTAMP 型のフィールドの開始年月が決定されます。

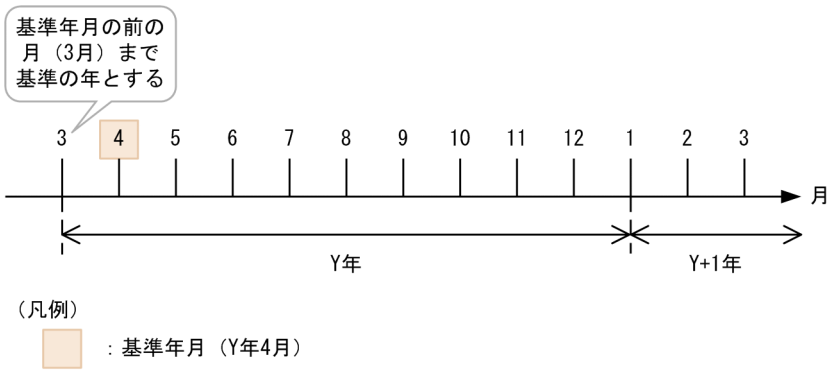
- 入力アダプター起動直後の一つ目の入力レコード
  format タグの year 属性および month 属性で指定した値が，TIMESTAMP 型のフィールドの開始年月となります。
- 二つ目以降の入力レコード
  前回の入力レコードの TIMESTAMP 型のフィールドの年月を基準年月として，開始年月を決定します。

なお，二つ目以降の入力レコードの開始年月は，入力レコードのフィールドの月の値を基準年月と比較して，次のように決定されます。

- 「入力レコードのフィールドの月の値 >= 基準年月の前の月」の場合
  基準年月の年がフィールドの年の値となります。ただし，基準年月の前の月が 12 月で，かつフィールドの月の値が 12 月の場合は，基準年月の前の年がフィールドの年の値となります。
- 「入力レコードのフィールドの月の値 < 基準年月の先月」の場合
  基準年月の翌年がフィールドの年の値となります。ただし，基準年月の前の月が 12 月の場合は，基準年月の年がフィールドの年の値となります。

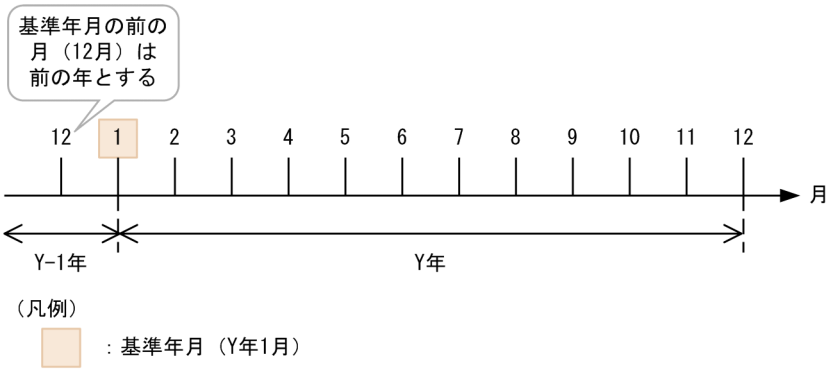
基準年月を Y 年 4 月とした場合の例を次の図に示します。入力レコードのフィールドの月の値が 3～12 の場合，フィールドの年の値は Y となります。月の値が 1 または 2 の場合，フィールドの年の値は Y ＋ 1 となります。

図 9-3　基準年月を Y 年 4 月とした場合



基準年月を Y 年 1 月とした場合の例を次の図に示します。入力レコードのフィールドの月の値が 12 の場合，フィールドの年の値は Y－1 となります。月の値が 1～11 の場合，フィールドの年の値は Y となります。

図 9-4　基準年月を Y 年 1 月とした場合



(5)　レコード構成の記述形式

record タグの exp 属性で指定するレコード構成の記述形式について説明します。

記述形式を次に示します。

```
{〔＜区切り文字指定＞〕（$＜フィールド名指定＞）〔＜区切り文字指定＞〕
｜〔＜区切り文字指定＞〕（$＜フィールド名指定＞）〔＜区切り文字指定＞〕…}
```

＜区切り文字指定＞

このレコードを構成する個々のフィールドを区切る文字列を指定します。FormatDefinition タグの ioType 属性が INPUT の場合，文字列として正規表現が使用できます。正規表現で特別な意味を持つ文字（「(」，「)」，「[」，「]」，「{」，「}」，「-」，「|」，「/」，「¥」，「.」，「*」，「?」，「+」，「^」，「$」）を本来の文字として指定するには，円記号（¥）でエスケープする必要があります。

FormatDefinition タグの ioType 属性が OUTPUT の場合，文字列として正規表現を使用できません。そのため，正規表現で特別な意味を持つ文字に対してエスケープする必要はありません。

＜フィールド名指定＞

このレコードを構成する個々のフィールド名を指定します。

レコード構成の記述規則を次に示します。

- 区切り文字指定に対するダブルクォーテーション（"）は不要です。
- フィールド名の指定の先頭には「($_」を，末尾には「)」を指定してください。
- フィールド名の指定では，このレコードを構成するすべてのフィールド名を指定してください。
- field タグでは，このレコードを構成するフィールド名を左から順番に指定してください。
- フィールド名の指定では，このレコードを構成するフィールド名を重複して指定できません。また，このレコードを構成するフィールド名以外のフィールド名は指定できません。
- 特殊文字を使用する場合は，文字を置換してください。文字の置換については，「9.2　アダプター用定義ファイル作成上の注意事項」の特殊文字（記号）の対応表を参照してください。

レコード構成の記述例を次に示します。

記述例 1

- フィールド F1～F5 という五つのフィールドがある。
- すべてのフィールドは，コンマ「,」で区切られている。

このようなレコードのレコード構成は次のように記述します。

”($_F1),($_F2),($_F3),($_F4),($_F5)”

記述例 2

- フィールド F1～F3 という三つのフィールドがある。
- レコードの先頭が「<」で始まる。
- フィールド F1 とフィールド F2 が「> MSG」の区切り文字で区切られている。
- フィールド F2 とフィールド F3 が半角スペースの区切り文字で区切られている。

このようなレコードのレコード構成は次のように記述します。

”<($_F1)> MSG($_F2) ($_F3)”

「<」と「>」は特殊文字のため，それぞれ「<」および「>」に置き換えます。

(6)　type 属性の指定値と，対応する Java のデータ型および CQL のデータ型

field タグの type 属性の指定値と，対応する Java のデータ型および CQL のデータ型を次の表に示します。

なお，各データ型の表記規則については，field タグの pattern 属性の各データ型種別の表記規則の説明を参照してください。また，CQL のデータ型については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

表 9-12　type 属性の指定値と，対応する Java のデータ型および CQL のデータ型

項番	データ型種別 (type 属性の指定値)	分類	データ形式	Java のデータ型	CQL のデータ型
1	INT	数値データ	整数型 4 バイト	プリミティブ型 int	INT［EGER］
2	SHORT		整数型 2 バイト	プリミティブ型 short	SMALLINT
3	BYTE		整数型 1 バイト	プリミティブ型 byte	TINYINT
4	LONG		整数型 8 バイト	プリミティブ型 long	BIGINT
5	BIG_DECIMAL		固定小数点数	java.math.BigDecimal クラス	DEC［IMAL］※1 NUMERIC※1
6					DECIMAL(m)※2 NUMERIC(m)※2
7	FLOAT		実数型 4 バイト	プリミティブ型 float	REAL
8	DOUBLE		実数型 8 バイト	プリミティブ型 double	FLOAT DOUBLE
9	STRING	文字データ	文字列	java.lang.String クラス	CHAR［ACTER］※3
10					CHAR［ACTER］(n)※4 VARCHAR(p)※5
11	DATE	日付/時刻データ	日付（年月日）	java.sql.Date クラス	DATE
12	TIME		時間（時分秒）	java.sql.Time クラス	TIME
13	TIMESTAMP		日時(年月日＋時分秒＋ナノ秒)	java.sql.Timestamp クラス	TIMESTAMP※6

項番	データ型種別（type 属性の指定値）	分類	データ形式	Java のデータ型	CQL のデータ型
14	TIMESTAMP	日付/時刻データ	日時（年月日＋時分秒＋ナノ秒）	java.sql.Timestamp クラス	TIMESTAMP〔(q)〕※7

注※1

けた数として 15 を仮定します。けた数が 15 を超える場合，タプル送信時にエラーとなります。

注※2

m は正整数で，1≦m≦38 です。けた数が m を超える場合，タプル送信時にエラーとなります。

注※3

文字数として 1 を仮定します。文字数が 1 を超える場合，タプル送信時にエラーとなります。

注※4

n は正整数で，1≦n≦255 です。文字数が n を超える場合，タプル送信時にエラーとなります。

注※5

p は正整数で，1≦p≦32767 です。文字数が p を超える場合，タプル送信時にエラーとなります。

注※6

年月日＋時分秒＋ミリ秒（3 けた）を仮定します。ミリ秒以上の精度を指定した場合，タプル送信エラーとなります。

注※7

q は整数で，0≦q≦9 です。q は小数秒以下のけた数を示します。けた数に q 以上の精度を指定した場合，タプル送信エラーとなります。

(7)　データ型種別の表記規則

field タグの pattern 属性を省略した場合のデータ型種別の表記規則を次の表に示します。

表 9-13　field タグの pattern 属性を省略した場合のデータ型種別の表記規則

項番	データ型種別（type 属性の指定値）	各データ型のパターン（正規表現）	説明	変更の可否※
1	INT	"〔-〕{0,1}〔0-9〕＋"	先頭がマイナス（-）の符号が 0 回または 1 回出現し，さらに 0～9 の数字が 1 回以上繰り返される文字列のパターンです。	×
2	SHORT			×
3	BYTE			×
4	LONG			×
5	BIG_DECIMAL	"〔-〕{0,1}〔0-9〕＋¥.〔0-9〕＋"	次の文字列のパターンです。 • 先頭がマイナス（-）の符号が 0 回または 1 回出現します。 • 0～9 の数字が 1 回以上繰り返されます。 • ピリオド（.）を挟んで，0～9 の数字が 1 回以上繰り返されます。	×
6	FLOAT	"〔-〕{0,1}〔0-9〕＋¥.〔0-9〕＋"	次の文字列のパターンです。 • 先頭がマイナス（-）の符号が 0 回または 1 回出現します。 • 0～9 の数字が 1 回以上繰り返されます。	×
7	DOUBLE			×

項番	データ型種別（type 属性の指定値）	各データ型のパターン（正規表現）	説明	変更の可否※
7	DOUBLE	"〔-〕{0,1}〔0-9〕+¥.〔0-9〕+"	• ピリオド（.）を挟んで，0～9 の数字が 1 回以上繰り返されます。	×
8	STRING	"〔^, ;〕*"	スペース，コンマ（,），およびセミコロン（;）を除く任意の文字が繰り返される文字列のパターンです。 データ型種別が STRING の場合，パターンの変更（指定）ができます。 パターン変更（指定）例 区切り文字にピリオド（.）だけを使用する場合 ”〔^ .〕+”	○
9	DATE	"〔0-9〕{1,4}-〔0-9〕{1,2}-〔0-9〕{1,2}"	yyyy-mm-dd 形式の文字列のパターンです。ただし，指定できる文字は次の範囲とし，けた数だけチェックします。 • yyyy：0～9999 の数字 • mm：0～99 の数字 • dd：0～99 の数字 mm，dd がそれぞれ MM，DD の指定範囲外の値の場合，エラーメッセージ KFSP46322-E が出力されます。	×
10	TIME	"〔0-9〕{1,2}:〔0-9〕{1,2}:〔0-9〕{1,2}"	hh:mm:ss 形式の文字列のパターンです。ただし，指定できる文字は次の範囲とし，けた数だけチェックします。 • hh：0～99 の数字 • mm：0～99 の数字 • ss：0～99 の数字 hh，mm，ss がそれぞれ HH，MM，SS の指定範囲外の値の場合，java.sql.Time の仕様に準拠して正規の時刻に換算します。 （例） 16:22:66→16:23:06	×
11	TIMESTAMP	条件によって異なります。詳細については「表 9-14　データ型種別が TIMESTAMP の場合の表記規則」を参照してください。	条件によって異なります。詳細については「表 9-14　データ型種別が TIMESTAMP の場合の表記規則」を参照してください。	×

（凡例）

○：変更できます。

×：変更できません。

注※

pattern 属性を指定することで，各データ型種別の文字列のパターンを変更できるかどうかを表します。

表 9-14　データ型種別が TIMESTAMP の場合の表記規則

条件	各データ型のパターン（正規表現）	説明
timestampformat 属性の値が 1 の場合	"[0-9]{1,4}-[0-9]{1,2}-[0-9]{1,2} [0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}¥.[0-9]{1,9}"	yyyy-mm-dd hh:mm:ss.fffffffff 形式の文字列のパターンです。ただし，指定できる文字は次の範囲とし，けた数だけチェックします。※ • yyyy：0～9999 の数字 • mm：0～99 の数字 • dd：0～99 の数字 • hh：0～99 の数字 • mm：0～99 の数字 • ss：0～99 の数字 • fffffffff：0～999999999 の数字
timestampformat 属性の値が 2 の場合	"[A-Za-z]{3}[ ]+[0-9]{1,2}[ ]+[0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}"	MMM dd HH:mm:ss 形式の文字列のパターンです。ただし，指定できる文字は次の範囲とし，けた数だけチェックします。※ • MMM：A～Z，a～z の英字（3 けたの英語の月名） • dd：0～99 の数字 • HH：0～99 の数字 • mm：0～99 の数字 • ss：0～99 の数字
timestampformat 属性の値が 3 の場合	"[0-9]{1,4}/[0-9]{1,2}/[0-9]{1,2}[ ]+[0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}¥.[0-9]{3}"	yyyy/MM/dd HH:mm:ss.SSS 形式の文字列のパターンです。ただし，指定できる文字は次の範囲とし，けた数だけチェックします。※ • yyyy：0～9999 の数字 • MM：0～99 の数字 • dd：0～99 の数字 • HH：0～99 の数字 • mm：0～99 の数字 • ss：0～99 の数字 • SSS：000～999 の数字
timestampformat 属性の値が 4 の場合	"[0-9]{1,2}/[A-Za-z]{3}/[0-9]{1,4}:[0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}"	dd/MMM/yyyy:HH:mm:ss 形式の文字列のパターンです。ただし，指定できる文字は次の範囲とし，けた数だけチェックします。※ • dd：0～99 の数字 • MMM：A～Z，a～z の英字（3 けたの英語の月名） • yyyy：0～9999 の数字 • HH：0～99 の数字 • mm：0～99 の数字 • ss：0～99 の数字

注※

mm, dd, hh, mm, ss がそれぞれ MM（月）, DD, HH, MM（分）, SS の指定範囲外の値の場合, java.sql.Timestamp の仕様に準拠して正規の年月日時刻に換算します。

（例）

2001-13-10 16:22:66.101→2002-01-10 16:23:06.101

9.11.2　マッピング定義

マッピング定義（MappingDefinition タグ）は,「9.9.3　編集用 CB 定義」で説明した編集用 CB 定義（DataEditCBDefinition タグ）の子要素として定義します。

なお, マッピングの処理については,「11.2.2(4)　マッピング」, または「11.6.2(3)　マッピング」を参照してください。

(1)　記述形式

```
<MappingDefinition ioType=" {INPUT | OUTPUT} ">
  <source>
    <streams>
      <stream name="＜ストリーム名＞"
       querygroup="＜クエリグループ名＞">
        <column name="＜列名＞"
         type=" {INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT | DOUBLE | STRING  | DATE | TIME |
TIMESTAMP} "/>
      </stream>
    </streams>
  </source>
  <target>
    <streams>
      <stream name="＜ストリーム名＞"
       querygroup="＜クエリグループ名＞">
        <column name="＜列名＞"
         type=" {INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT | DOUBLE | STRING  | DATE | TIME |
TIMESTAMP} "/>
      </stream>
    </streams>
    <records>
      <record name="＜レコード名＞">
        <field name="＜フィールド名＞" type=" {INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT |
DOUBLE | STRING  | DATE | TIME | TIMESTAMP} "/>
    </records>
  </target>
  <intermediate>
    <mappings source="＜変換元名＞"
     querygroup="＜クエリグループ名＞"
     target="＜変換先名＞">
      <map source="＜変換元パス式＞"
       function="＜組み込み関数名＞" argument1="＜第1引数＞"
       argument2="＜第2引数＞" target="＜変換先パス式＞"/>
    </mappings>
  </intermediate>
</MappingDefinition>
```

(2)　定義の詳細

MappingDefinition タグ（全体情報の定義）

マッピング変換定義の全体情報を定義します。この定義は必ず 1 個だけ記述します。

ioType=" {INPUT | OUTPUT} "

この定義で指定する標準提供アダプターの種類を指定します。この属性は省略できません。

指定できる値を次に示します。

- INPUT

入力アダプターでマッピングを定義する場合に指定します。

- OUTPUT

  出力アダプターでマッピングを定義する場合に指定します。

source タグ（マッピング変換元定義）

マッピングの変換元のストリームの情報を定義します。

このタグに指定する内容は，次の表に示すように，MappingDefinition タグの ioType 属性の指定値とマッピングの対象によって異なります。

ioType 属性の指定値	マッピングの対象	source タグに指定する内容
INPUT	レコードとストリーム間のマッピング	空要素タグを指定します。
	レコード間のマッピング	
OUTPUT	レコードとストリーム間のマッピング	streams タグを指定します。
	レコード間のマッピング	空要素タグを指定します。

target タグ（マッピング変換先定義）

マッピングの変換先のストリーム，またはレコードの情報を定義します。

このタグに指定する内容は，次の表に示すように，MappingDefinition タグの ioType 属性の指定値とマッピングの対象によって異なります。

ioType 属性の指定値	マッピングの対象	target タグに指定する内容
INPUT	レコードとストリーム間のマッピング	streams タグを指定します。
	レコード間のマッピング	records タグを指定します。
OUTPUT	レコードとストリーム間のマッピング	空要素タグを指定します。
	レコード間のマッピング	records タグを指定します。

streams タグ（ストリーム群定義）

レコードとストリームとの間のマッピングの場合に，source タグまたは target タグ下で，変換元または変換先のストリーム群の情報を定義します。

この定義を記述する場合は，必ず 1 個だけ記述します。

stream タグ（ストリーム定義）

変換元または変換先のストリームの情報を定義します。

この定義は 1,024 個まで記述できます。この定義は省略できません。

name="＜ストリーム名＞"

ストリーム名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。名前の先頭は半角英字だけ指定できます。また，ストリーム名とクエリグループ名の組は，streams タグ内で一意となるように指定してください。この属性は省略できません。

querygroup="＜クエリグループ名＞"

クエリグループ名を 1～64 文字の半角英数字，およびアンダーライン（_）で指定します。名前の先頭は半角英字だけ指定できます。また，ストリーム名とクエリグループ名の組は，streams タグ内で一意となるように指定してください。この属性は省略できません。

column タグ（列定義）

変換元または変換先のストリームの列の情報を定義します。

この定義は 3,000 個まで記述できます。この定義は省略できません。

name="＜列名＞"

列名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。名前の先頭は半角英字だけ指定できます。また，列名は，streams タグ内の列定義の中で一意となるように指定してください。この属性は省略できません。

type=" {INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT | DOUBLE | STRING | DATE | TIME | TIMESTAMP} "

CQL のデータ型に対応するデータ型種別を指定します。

この属性に指定する値と，対応する CQL のデータ型については，「9.11.1(6)　type 属性の指定値と，対応する Java のデータ型および CQL のデータ型」を参照してください。

records タグ（レコード群定義）

レコード間のマッピングの場合に，target タグ下で，変換先のレコード群の情報を定義します。

この定義を記述する場合は，必ず 1 個だけ記述します。

record タグ（レコード定義）

変換先のレコードの情報を定義します。この定義は 1,024 個まで記述できます。この定義は省略できません。

name="＜レコード名＞"

レコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

この属性は 1 個だけ記述できます。この属性は省略できません。レコード名は，該当の標準提供アダプター内で一意となるように指定してください。

field タグ（フィールド定義）

変換先のレコードのフィールド情報を定義します。この定義は 3,000 個まで記述できます。この定義は省略できません。

name="＜フィールド名＞"

フィールド名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この属性は 1 個だけ記述できます。この属性は省略できません。フィールド名は，該当のレコード内で一意となるように指定してください。

type=" {INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT | DOUBLE | STRING | DATE | TIME | TIMESTAMP} "

フィールドの Java のデータ型に対応するデータ型を指定します。この属性は省略できません。

この属性に指定する値と，対応する CQL のデータ型については，「9.11.1(6)　type 属性の指定値と，対応する Java のデータ型および CQL のデータ型」を参照してください。

intermediate タグ（マッピング中間定義）

マッピング変換元からマッピング変換先へのマッピング情報を定義します。

この定義は必ず 1 個だけ記述します。

mappings タグ（マッピング群定義）

マッピング情報を定義します。

この定義は 1,024 個まで記述できます。この定義は省略できません。

source="＜変換元名＞"

変換元のストリーム名，またはレコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。この属性は省略できません。

この属性に指定する内容は，次の表に示すように，MappingDefinition タグの ioType 属性の指定値とマッピングの対象によって異なります。

ioType 属性の指定値	マッピングの対象	source 属性に指定する内容
INPUT	レコードとストリーム間のマッピング	変換元のレコード名を指定します。
	レコード間のマッピング	
OUTPUT	レコードとストリーム間のマッピング	stream タグの name 属性に指定したストリーム名を指定します。
	レコード間のマッピング	変換元のレコード名を指定します。

querygroup="＜クエリグループ名＞"

レコードとストリームとの間のマッピングの場合に，変換元または変換先のストリームが属するクエリグループ名を 1～64 文字の半角英数字，およびアンダーライン（_）で指定します。名前の先頭は半角英字だけ指定できます。また，クエリグループ名と変換元名の組，またはクエリグループ名と変換先名の組は，mappings タグ内で一意となるように指定してください。

この属性に指定する内容は，次の表に示すように，MappingDefinition タグの ioType 属性の指定値とマッピングの対象によって異なります。

ioType 属性の指定値	マッピングの対象	querygroup 属性に指定する内容
INPUT	レコードとストリーム間のマッピング	target タグ下の streams タグ内のクエリグループ名を指定します。
	レコード間のマッピング	指定しません。
OUTPUT	レコードとストリーム間のマッピング	source タグ下の streams タグ内のクエリグループ名を指定します。
	レコード間のマッピング	指定しません。

target="＜変換先名＞"

変換先のストリーム名，またはレコード名を 1～100 文字で指定します。変換先名は mappings タグ内で一意となるように指定してください。この属性は省略できません。

この属性に指定する内容は，次の表に示すように，MappingDefinition タグの ioType 属性の指定値とマッピングの対象によって異なります。

ioType 属性の指定値	マッピングの対象	target 属性に指定する内容
INPUT	レコードとストリーム間のマッピング	stream タグの name 属性に指定したストリーム名を指定します。
	レコード間のマッピング	record タグの name 属性に指定したレコード名を指定します。
OUTPUT	レコードとストリーム間のマッピング	変換先のレコード名を指定します。

ioType 属性の指定値	マッピングの対象	target 属性に指定する内容
OUTPUT	レコード間のマッピング	record タグの name 属性に指定したレコード名を指定します。

map タグ（マッピング情報定義）

マッピング情報を定義します。このタグの指定順は，変換先となるストリーム，またはレコードの構成の順番と一致させる必要があります。この定義は，3,000 個まで記述できます。この定義は省略できません。

source="＜変換元パス式＞"

変換元パス式として，変換元のフィールド名を 1～1,024 文字で指定します。この属性は省略できません。

この属性に指定する内容は，次の表に示すように，MappingDefinition タグの ioType 属性の指定値とマッピングの対象によって異なります。

ioType 属性の指定値	マッピングの対象	source 属性に指定する内容
INPUT	レコードとストリーム間のマッピング	mappings タグの source 属性に指定したレコード名に属するフィールド名を指定します。
	レコード間のマッピング	
OUTPUT	レコードとストリーム間のマッピング	mappings タグの source 属性に指定したストリーム名に属するフィールド名を指定します。
	レコード間のマッピング	mappings タグの source 属性に指定したレコード名に属するフィールド名を指定します。

function="＜組み込み関数名＞"

レコード間のマッピングを行う場合は，map タグの function 属性で**組み込み関数**を使用して，マッピングの変換元の共通形式レコードから文字列や時刻を取得し，取得した文字列や時刻をマッピングの変換先の共通形式レコードに反映できます。

この属性では，組み込み関数を使用して文字列や時刻を取得する場合に，組み込み関数名を指定します。なお，レコードとストリーム間のマッピング（入力ストリーム用の共通形式レコードへのマッピング，または出力ストリーム用の共通形式レコードからのマッピング）では，この属性は指定できません。

この属性に指定できる文字列を次に示します。

- regexsubstring

  入力文字列から，正規表現の部分文字列を取得する場合に指定します。戻り値は，String 型の文字列です。

  regexsubstring を指定する場合，argument1 属性，および argument2 属性で引数を指定してください。

- getTupleTime

  この値は，出力アダプターの場合だけ指定できます。出力アダプターで出力ストリームから出力されたタプルの時刻を取得する場合に指定します。戻り値は，TIMESTAMP 型の時刻（単位：ミリ秒）です。

  getTupleTime を指定する場合，argument1 属性，および argument2 を指定する必要はありません。

なお，MappingDefinition タグの ioType 属性の指定値が INPUT の場合には，getTupleTime は指定できません。

- subTime

  二つのフィールドの時刻の差分を算出する場合に指定します。二つの時刻のうち，大きい値から小さい値を引いて，時刻の差分を算出します。戻り値は，LONG 型の時刻（単位：ミリ秒）です。

  subTime を指定する場合，argument1 属性，および argument2 属性で引数を指定してください。

argument1="＜第 1 引数＞"

function 属性で指定した組み込み関数の第 1 引数を指定します。

- function 属性で regexsubstring を指定した場合

  入力文字列が格納されている String 型のフィールドのフィールド名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。

- function 属性で subTime を指定した場合

  時刻が格納されている TIMESTAMP 型のフィールドのフィールド名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。

argument2="＜第 2 引数＞"

function 属性で指定した組み込み関数の第 2 引数を指定します。

- function 属性で regexsubstring を指定した場合

  部分文字列の抽出条件を 2～1,024 文字の次に示す形式で指定します。

  [正規表現] "("[正規表現]")"[正規表現]

  正規表現の解析には，java.util.regex.Pattern クラスを使用します。正規表現は，java.util.regex.Pattern クラスがサポートする正規表現の範囲で記述してください。

  第 1 引数で指定したフィールドの入力文字列が，第 2 引数で指定した正規表現の文字列に一致する場合には，「(」と「)」で囲まれた部分に相当する文字列（String 型）が戻り値となります。一致しない場合には，空文字が戻り値となります。

- function 属性で subTime を指定した場合

  時刻が格納されている TIMESTAMP 型のフィールドのフィールド名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。

target="＜変換先パス式＞"

変換先パス式として，変換先のフィールド名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。フィールド名は mappings タグ内で一意となるように指定してください。この属性は省略できません。

次に示すとおり，この属性の指定内容は，MappingDefinition タグの ioType 属性に指定した内容によって異なります。

- ioType 属性に INPUT を指定した場合

  変換先のストリーム名またはレコード名に属するフィールド名を指定します。

- ioType 属性に OUTPUT を指定した場合

  変換先のレコード名に属するフィールド名を指定します。

(3)　記述例

- 入力アダプターでレコードとストリーム間のマッピングの場合

  入力アダプターで，レコードとストリーム間のマッピングをする場合の記述例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
<!-- 途中略 -->
 xmlns:map="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
MappingDefinition">
<!-- 途中略 -->

<!-- 編集用CB定義 -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor1">
<!-- マッピング定義 -->
  <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>
```

- 入力アダプターでレコード間のマッピングの場合

  入力アダプターで，レコード間のマッピングをする場合の記述例を次に示します。この例記述では，function 属性で組み込み関数を使用して，正規表現文字列と時刻の差分を取得しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
<!-- 途中略 -->
 xmlns:map="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
MappingDefinition">
<!-- 途中略 -->

<!-- 編集用CB定義 -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor1">
<!-- マッピング定義 -->
  <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>
```

- 出力アダプターでレコードとストリーム間のマッピングの場合

出力アダプターで，レコードとストリーム間のマッピングをする場合の記述例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
<!-- 途中略 -->
 xmlns:map="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
MappingDefinition">
<!-- 途中略 -->

<!-- 編集用CB定義 -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor1">
<!-- マッピング定義 -->
  <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>
```

- 出力アダプターでレコード間のマッピングの場合

出力アダプターで，レコード間のマッピングをする場合の記述例を次に示します。この記述例では，function 属性で組み込み関数を使用して，正規表現文字列，タプルの時刻，および時刻の差分を取得しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
<!-- 途中略 -->
 xmlns:map="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
MappingDefinition">
<!-- 途中略 -->

<!-- 編集用CB定義 -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor1">
<!-- マッピング定義 -->
  <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　フィルター定義

フィルター定義（FilterDefinition タグ）は，「9.9.3　編集用 CB 定義」で説明した編集用 CB 定義（DataEditCBDefinition タグ）の子要素として定義します。

なお，レコードのフィルタリングの処理については，「11.4　レコードのフィルタリング」を参照してください。

(1)　記述形式

```
<FilterDefinition>
  <record source="＜フィルター対象のレコード名＞"
   conditionName="＜レコード条件名＞" condition="{AND｜OR}">
    <field source="＜フィールド名＞"
     condition="{eq｜ge｜gt｜le｜lt｜ne}" value="＜条件値＞"/>
  </record>
</FilterDefinition>
```

(2)　定義の詳細

FilterDefinition タグ（全体情報の定義）

フィルター定義の全体情報を定義します。この定義は必ず 1 個だけ記述します。

record タグ（レコード条件の定義）

レコード条件を定義します。この定義は 10 個まで記述できます。この定義は省略できません。

レコード条件とは，フィルター対象のレコードを取捨選択するための条件の一つです。レコード条件には，フィルター対象のレコード名を指定します。

source="＜フィルター対象のレコード名＞"

レコードのフィルタリングの前に処理したコールバックから渡されたレコード形式の中で，フィルター対象とするレコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

この属性は 1 個だけ記述できます。

conditionName="＜レコード条件名＞"

レコード条件名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

この定義は必ず 1 個だけ記述します。また，FilterDefinition タグ内で一意となるように指定してください。

condition="{AND | OR}"

レコード条件の論理条件を指定します。省略した場合，AND が仮定されます。

指定できる値を次に示します。

- AND

  複数のフィールド条件にすべて一致したフィールドを持つレコードを出力結果として出力します。

- OR

  複数のフィールド条件に一つでも一致したフィールドを持つレコードを出力結果として出力します。

field タグ（フィールド条件の定義）

フィールド条件を定義します。この定義は 10 個まで記述できます。この定義は省略できません。

フィールド条件とは，フィルター対象のレコードを取捨選択するための条件の一つです。フィールド条件には，フィルター対象のレコードのフィールド名，比較演算記号，および条件値を指定します。

source="＜フィールド名＞"

record タグの source で指定したレコード形式のフィールド名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

この属性は省略できません。

condition="｛eq | ge | gt | le | lt | ne｝"

フィールド条件の比較演算子を表す値を指定します。

この属性に指定できる値は，field タグの value 属性に文字データを指定するか，数値データを指定するかによって異なります。指定できる値を次の表に示します。省略した場合，eq が仮定されます。

この属性の指定値	比較演算子	value 属性の指定値	
		文字データ	数値データ
eq	=	○	○
ge	>=	×	○
gt	>	×	○
le	<=	×	○
lt	<	×	○
ne	!=	○	○

（凡例）

○：指定できます。

×：指定できません。

比較演算子の意味を次の表に示します。

比較演算子	比較演算子の使用例	使用例の意味
=	A = B	A は B と等しい
>=	A >= B	A は B 以上
>	A > B	A は B より大きい
<=	A <= B	A は B 以下
<	A < B	A は B より小さい
!=	A != B	A は B と等しくない

value="＜条件値＞"

フィールド条件の条件値を文字データ，または数値データで指定します。使用できる文字，または数値を次に示します。この属性は省略できません。

- 文字データの場合

  指定できる文字数は 1～128 文字です。

  文字列として正規表現が使用できます。正規表現の解析には，java.util.regex.Pattern クラスを使用します。このため，正規表現は，java.util.regex.Pattern クラスがサポートする正規表現の範囲で記述してください。

  正規表現で特別な意味を持つ文字を本来の文字として使用するには，円記号（¥）でエスケープする必要があります。

正規表現で特別な意味を持つ文字を次に示します。

「(」，「)」，「[」，「]」，「{」，「}」，「-」，「|」，「/」，「¥」，「.」，「*」，「?」，「+」，「^」，「$」

- 数値データの場合

-9223372036854775808～9223372036854775807 の整数を指定できます。

(3)　記述例

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
<!-- 途中略 -->
 xmlns:filter="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
FilterDefinition">
<!-- 途中略 -->

<!-- 編集用CB定義 -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.filter.FilterCBImpl" name="filter1">
  <!-- フィルター定義 -->
  <filter:FilterDefinition>
    <!-- レコード条件の定義 -->
    <filter:record source="R1" conditionName="filterName1" condition="AND">
      <!-- フィールド条件の定義 -->
      <filter:field source="F11" condition="ge"  value="1"/>
      <filter:field source="F11" condition="le"  value="100"/>
      <filter:field source="F21" condition="eq"  value=".*TARO.*"/>
    </filter:record>
    <!-- レコード条件の定義 -->
    <filter:record source="R2" conditionName="filterName2" operator="OR">
      <!-- フィールド条件の定義 -->
      <filter:field source="F21" value="1"/>
      <filter:field source="F22" condition="ne"  value=".*HANAKO.*"/>
    </filter:record>
  </filter:FilterDefinition>
</cb:DataEditCBDefinition>
```

9.11.4　レコード抽出定義

レコード抽出定義（RecordExtractionDefinition タグ）は，「9.9.3　編集用 CB 定義」で説明した編集用 CB 定義（DataEditCBDefinition タグ）の子要素として定義します。

なお，レコードの抽出の処理については，「11.5　レコードの抽出」を参照してください。

(1)　記述形式

```
<RecordExtractionDefinition>
  <targetrecords>
    <targetrecord name="＜抽出対象レコード名＞">
      <record source="＜レコード名＞"
       timeposition="＜時刻フィールド名＞" condition=" {AND | OR} ">
        <field source="＜フィールド名＞"
         condition=" {eq | ge | gt | le | lt | ne} " value="＜条件値＞"/>
      </record>
    </targetrecord>
  </targetrecords>
  <extractions size="＜レコード最大保持数＞" timeout=" {ON | OFF} "
   samerecord=" {overwrite | delete} "
    <extraction name="＜抽出条件名＞" timelimit="＜タイムアウト時間＞">
      <targets>
        <target sourceL ="＜抽出対象レコード名＞"
         sourceR="＜抽出対象レコード名＞" condition="AND">
          <fieldcondition sourceL="＜フィールド名＞"
           condition="eq" sourceR="＜フィールド名＞"/>
        </target>
      </targets>
      <extractrecord name="＜抽出レコード名＞">
        <select source="＜抽出対象レコード名＞"/>
      </extractrecord>
```

```
      <timeoutrecord name="＜タイムアウトレコード名＞" />
    </extraction>
  </extractions>
</RecordExtractionDefinition>
```

(2) 定義の詳細

RecordExtractionDefinition タグ（全体情報の定義）

レコード抽出定義の全体情報を定義します。この定義は必ず 1 個だけ記述します。

targetrecords タグ（抽出対象条件群の定義）

抽出対象条件群の情報を定義します。抽出対象条件とは，抽出対象のレコードを取捨選択するための条件の一つです。抽出対象条件には，レコード条件とフィールド条件を定義します。レコード条件（record タグ）では抽出対象のレコード名を，フィールド条件（field タグ）では抽出対象のレコードのフィールド値を指定します。

targetrecord タグ（抽出対象条件の定義）

抽出対象条件を定義します。この定義は 10 個まで記述できます。この定義は省略できません。

name="＜抽出対象レコード名＞"

抽出したレコードに付けるレコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。この定義は 1 個だけ記述できます。

この属性で指定したレコード名は，select タグの source 属性，target タグの sourceL 属性と sourceR 属性で指定します。

record タグ（レコード条件定義）

レコード条件を定義します。この定義は 1 個だけ記述できます。

source="＜レコード名＞"

レコードの抽出の前に処理したコールバックから渡されたレコード形式の中で，抽出対象とするレコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。この属性は省略できません。

timeposition="＜時刻フィールド名＞"

TIMESTAMP 型の時刻情報のフィールドを 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。この属性は省略できません。

condition=" {AND | OR} "

レコード条件の論理条件を定義します。省略した場合，AND が仮定されます。

指定できる値を次に示します。

- AND

  複数のフィールド条件にすべて一致したフィールドを持つレコードを出力結果として出力します。

- OR

  複数のフィールド条件に一つでも一致したフィールドを持つレコードを出力結果として出力します。

field タグ（フィールド条件定義）

フィールド条件を定義します。この定義は 10 個まで記述できます。この定義は省略できません。

フィールド条件は，抽出対象とするレコードのフィールドごとに指定します。

source="＜フィールド名＞"

record タグの source で指定したレコード形式のフィールド名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。この属性は省略できません。

condition="｛eq ｜ ge ｜ gt ｜ le ｜ lt ｜ ne｝"

フィールド条件の比較演算子を表す値を指定します。

この属性に指定できる値は，field タグの value 属性に文字データを指定するか，数値データを指定するかによって異なります。指定できる値を次に示します。省略した場合，eq が仮定されます。

- 文字データの場合

  eq，または ne が指定できます。

- 数値データの場合

  eq，ge，gt，le，lt，および ne が指定できます。

指定できる値の意味については，「9.11.3　フィルター定義」の field タグの condition 属性の説明を参照してください。

value="＜条件値＞"

フィールド条件の条件値を文字データ，または数値データで指定します。使用できる文字，または数値を次に示します。この属性は省略できません。

- 文字データの場合

  指定できる文字数は 1～128 文字です。

  文字列として正規表現が使用できます。正規表現の解析には，java.util.regex.Pattern クラスを使用します。このため，正規表現は，java.util.regex.Pattern クラスがサポートする正規表現の範囲で記述してください。

  正規表現で特別な意味を持つ文字を本来の文字として使用するには，円記号（¥）でエスケープする必要があります。

  正規表現で特別な意味を持つ文字を次に示します。

  「(」，「)」，「[」，「]」，「{」，「}」，「-」，「|」，「/」，「¥」，「.」，「*」，「?」，「+」，「^」，「$」

- 数値データの場合

  -9223372036854775808～9223372036854775807 の整数を指定できます。

extractions タグ（抽出条件群の定義）

抽出条件群の情報を定義します。抽出条件とは，抽出対象のレコードを取捨選択するための条件の一つです。この定義は省略できません。

size="＜レコード最大保持数＞"

レコードバッファーに保持するレコード数の上限値を 1～1000000 の整数で指定します。この属性を省略した場合，10000 が仮定されます。

timeout="｛ON ｜ OFF｝"

extraction タグの timelimit 属性で指定した時間が経過し，レコードバッファーに保持しているレコードがタイムアウトした場合に，レコード（タイムアウトレコード）を出力するかどうかを指定します。

この属性を省略した場合，OFF が仮定されます。

指定できる値を次に示します。

- ON

  タイムアウトした場合に，タイムアウトレコードを出力します。

- OFF

  タイムアウトした場合に，タイムアウトレコードを出力しません。

なお，タイムアウトレコードについては，timeoutrecord タグで，レコード名を定義します。

samerecord="｛overwrite ｜ delete｝"

同一の抽出対象のレコードが入力された場合の対処を指定します。省略した場合，overwrite が仮定されます。

指定できる値を次に示します。

- overwrite

  同一レコードと見なされた既存のレコードが，レコードバッファーのどの位置にあるかによって処理が異なります。

  最も後ろのレコードの場合，新しいレコードで，レコードバッファーに保持していた既存のレコードを上書きします。最も後ろのレコードでない場合，上書きしたレコードより後ろのレコードを破棄して，新しいレコードを保持します。

- delete

  入力されたレコードを破棄します。

extraction タグ（抽出条件の定義）

抽出条件を定義します。この定義は 10 個まで記述できます。この定義は省略できません。

name="＜抽出条件名＞"

抽出条件名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

この定義は 1 個だけ記述できます。

timelimit="＜タイムアウト時間＞"

レコードバッファーに保持したレコードをタイムアウトと見なすまでの，レコードの生存時間（単位：ミリ秒）を 1～3600000 の整数（1 ミリ秒～1 時間）で指定します。

この属性を省略した場合，10000 が仮定されます。

targets タグ（レコード間条件群の定義）

レコード間条件群を定義します。

レコード間条件とは，抽出条件を成立させるための条件の一つです。target タグではレコードの入力順序を指定し，fieldcondition タグではレコード間でフィールド値を比較するためのフィールド名を指定します。

target タグ（レコード間条件の定義）

レコード間条件として，レコードの入力順序を指定します。この定義は 9 個まで記述できます。この定義は省略できません。

この定義では，sourceL 属性，および sourceR 属性でレコード名を指定して，レコードの入力順序を定義します。例えば，次のように指定した場合には，レコードの入力順序は，R1→R2→R3 となり，この入力順序どおりに入力されたレコードがレコード条件に一致するレコードとなります。

```
       :
<rex:target sourceL="R1" sourceR="R2">
       :
<rex:target sourceL="R2" sourceR="R3">
       :
```

sourceL="＜抽出対象レコード名＞"

レコード間条件の左辺のレコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

sourceR="＜抽出対象レコード名＞"

レコード間条件の右辺のレコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

condition="AND"

レコード間条件の論理条件を指定します。省略した場合，AND が仮定されます。

指定できる値を次に示します。

- AND

  複数のレコード間条件にすべて一致したレコードを出力結果として出力します。

fieldcondition タグ（フィールドの定義）

レコード間条件として，フィールド名を指定します。この定義は 10 個まで記述できます。この定義は省略できません。

この定義では，sourceL 属性，および sourceR 属性で，target タグで指定したレコードのフィールド名を指定します。例えば，次のように指定した場合には，レコード R1 のフィールド F11 と，レコード R2 のフィールド F21 の値を比較することになります。レコード R1 のフィールド F11 と，レコード R2 のフィールド F21 の値が一致する場合は，レコード間条件に一致すると判定されます。

```
      :
<rex:target sourceL="R1" sourceR="R2">
<rex:fieldcondition sourceL="F11" condition="eq" sourceR="F21"/>
      :
```

sourceL="＜フィールド名＞"

target タグの sourceL 属性で指定したレコードのフィールド名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

condition="eq"

レコード条件の比較演算子を表す値を指定します。指定できる値は，文字データと数値データのどちらを指定する場合も，比較演算子「=」を表す，eq です。省略した場合，eq が仮定されます。

sourceR="＜フィールド名＞"

target タグの sourceR 属性で指定したレコードのフィールド名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。

extractrecord タグ（抽出レコードの定義）

抽出レコードを定義します。

抽出レコードとは，抽出条件に一致したレコードを結合して，新たに生成するレコードのことです。

name="＜抽出レコード名＞"

抽出レコードのレコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。また，抽出レコード名は標準提供アダプター内で一意となるように指定してください。

select タグ（抽出対象レコードの定義）

抽出レコード生成のために結合するレコードを定義します。この定義は 10 個まで記述できます。この定義は省略できません。

source="＜抽出対象レコード名＞"

targetrecord タグの name 属性で指定したレコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。この属性は省略できません。

timeoutrecord タグ（タイムアウトレコードの定義）

タイムアウトの発生時に出力するレコードを定義します。この定義は，extractions タグの timeout 属性に ON を指定した場合は，必ず定義してください。timeout 属性に OFF を指定した場合は，この定義は無視されます。

name="<タイムアウトレコード名>"

出力するレコード名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。この属性は省略できません。また，タイムアウトレコード名は標準提供アダプター内で一意となるように指定してください。

(3) 記述例

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
 xmlns:rex ="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
RecordExtractionDefinition">
<!-- 途中略 -->

<!-- 編集用CB定義 -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.recordextract.RecordExtractionCBImpl"
name="extractor1">
  <!-- レコード抽出定義 -->
  <rex:RecordExtractionDefinition>
    <!-- 抽出対象条件群の定義 -->
    <rex:targetrecords>
      <rex:targetrecord name="element1">
        <rex:record source="R1" timeposition="F11" condition="AND">
          <rex:field source="F12" condition="eq" value="TARO"/>
          <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="JIRO"/>
          <rex:field source="F23" condition="le" value="50"/>
        </rex:record>
      </rex:targetrecord>
    </rex:targetrecords>
    <!-- 抽出条件群の定義 -->
    <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>
        <!-- 抽出レコードの定義 -->
        <rex:extractrecord name="ER1">
          <rex:select source="element1"/>
          <rex:select source="element2"/>
        </rex:extractrecord>
        <!-- タイムアウトレコードの定義 -->
        <rex:timeoutrecord name="TIMEOUT"/>
      </rex:extraction>
    </rex:extractions>
  </rex:RecordExtractionDefinition>
</cb:DataEditCBDefinition>
```

この記述例での定義内容を次に示します。

- 抽出対象のレコード形式

  R1，R2

- R1，R2 のフィールドの構成

  - R1 の場合

    F11（TIMESTAMP 型），F12（STRING 型），F13（INT 型），F14（INT 型）

  - R2 の場合

    F21（TIMESTAMP 型），F22（STRING 型），F23（INT 型），F24（INT 型）

- 抽出対象条件
  - R1 の場合
    F12 が TARO であり，かつ F13 が 100 より大きい。
  - R2 の場合
    F22 が JIRO，または F23 が 50 以下である。
- 抽出条件
  レコードは，R1，R2 の順番で入力される。
  R1 の F13 と，R2 の F23 の値が等しい。
  R1 の F14 と，R2 の F24 の値が等しい。
- 抽出レコード
  R1 と R2 を結合したレコードを生成する。
- タイムアウト
  レコードがタイムアウトするまでの時間は 10 秒である。また，タイムアウトした場合，TIMEOUT というレコード名でレコードを出力する。

9.12　アダプター構成定義ファイルの送受信用 CB 定義

ここでは，アダプター構成定義ファイル（AdaptorCompositionDefinition.xml）の送受信用 CB 定義について説明します。

入力ストリーム定義（streamInfo タグ）は，「9.9.4　送信用 CB 定義」で説明した送信用 CB 定義（SendCBDefinition タグ）の子要素として定義します。また，出力ストリーム定義（streamInfo タグ）は，「9.9.5　受信用 CB 定義」で説明した受信用 CB 定義（ReceiveCBDefinition タグ）の子要素として定義します。

送受信用 CB 定義の一覧を次の表に示します。

表 9-15　送受信用 CB 定義の一覧

項番	編集用 CB 定義	親要素	参照先
1	入力ストリーム定義 （streamInfo タグ）	送信用 CB 定義	9.12.1
2	出力ストリーム定義 （streamInfo タグ）	受信用 CB 定義	9.12.2

9.12.1　入力ストリーム定義

送信用 CB 定義では送信先となる入力ストリームに関する情報を指定します。

(1)　記述形式

```
<streamInfo name=”＜入力ストリーム名＞”
 querygroup=”＜クエリグループ名＞”/>
```

(2)　定義の詳細

この定義は 1,024 個まで記述できます。この定義は省略できません。

streamInfo タグ（全体情報の定義）

入力ストリーム定義の全体情報を定義します。

name="＜入力ストリーム名＞"

入力ストリーム名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。名前の先頭は半角英字だけ指定できます。入力ストリーム名は送信用 CB 定義内で一意となるように指定してください。この属性は省略できません。

なお，クエリ定義ファイルの register stream 句で指定したストリーム名が，アダプター構成定義ファイルで指定する入力ストリーム名となります。register stream 句については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

querygroup="＜クエリグループ名＞"

クエリグループ名を 1～64 文字の半角英数字，およびアンダーライン（_）で指定します。名前の先頭は半角英字だけ指定できます。この属性は省略できません。

9.12.2　出力ストリーム定義

受信用 CB 定義では送信元となる出力ストリームに関する情報を指定します。

(1)　記述形式

```
<streamInfo name=”＜出力ストリーム名＞”
 querygroup=”＜クエリグループ名＞”/>
```

(2)　定義の詳細

この定義は 1,024 個まで記述できます。また，この定義は省略できません。

streamInfo タグ（全体情報の定義）

入力ストリーム定義の全体情報を定義します。

name="＜出力ストリーム名＞"

出力ストリーム名を 1～100 文字の半角英数字，およびアンダーライン（_）で指定します。名前の先頭は半角英字だけ指定できます。出力ストリーム名は受信用 CB 定義内で一意となるように指定してください。この属性は省略できません。

クエリ定義ファイルの register query 句で指定したクエリ名が，アダプター構成定義ファイルで指定する出力ストリーム名となります。register query 句については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

querygroup="＜クエリグループ名＞"

クエリグループ名を 1～64 文字の半角英数字，およびアンダーライン（_）で指定します。名前の先頭は半角英字だけ指定できます。この属性は省略できません。

9.13　アダプター構成定義ファイルの記述例

ここでは，アダプター構成定義ファイルの記述例を示します。

9.13.1　記述例 1

ここでは，次のサンプルファイルに記述されているアダプター構成定義ファイルの記述例を示したあとに，記述例の内容を説明します。

＜インストールディレクトリ＞¥samples¥file¥conf¥xml¥AdaptorCompositionDefinition.xml

(1)　アダプター構成定義ファイルの記述例

```
<?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">

 <!-- 共通定義 -->
 <cmn:CommonDefinition>
  <!-- アダプタトレース定義 -->
  <cmn:AdaptorTraceDefinition trace="OFF"/>
 </cmn:CommonDefinition>

 <!-- ↓↓↓↓↓↓↓↓ インプロセスグループ定義 ↓↓↓↓↓↓↓↓ -->
 <!-- インプロセスグループ定義1 -->
 <adp:InprocessGroupDefinition name="InprocessAPTest">
  <!-- ↓↓↓↓↓↓ 入力アダプタ定義 複数定義可能 ↓↓↓↓↓↓ -->
  <!-- 入力アダプタ定義1 -->
  <adp:InputAdaptorDefinition name="InputAdaptor1"
   interval="0" charCode="MS932" lineFeed="CR_LF">

   <!-- 入力用CB定義 -->
   <cb:InputCBDefinition class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileInputCBImpl"
name="inputer">
    <!-- 入力コネクタ定義 -->
    <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>

   <!-- データ編集用CB定義 -->
   <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.formattranslate.InputFormatTranslatorCB
Impl" name="editor1">
    <!-- フォーマット変換定義 -->
    <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>

   <!-- データ編集用CB定義 -->
   <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor3">
    <!-- マッピング定義 -->
    <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>

   <!-- 送信用CB定義 -->
   <cb:SendCBDefinition class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.SendCBImpl"
name="sender">
    <cb:streamInfo name="DATA0" querygroup="Inprocess_QueryGroupTest"/>
   </cb:SendCBDefinition>

  </adp:InputAdaptorDefinition>
  <!-- ↑↑↑↑↑↑ 入力アダプタ定義 複数定義可能 ↑↑↑↑↑↑ -->

  <!-- ↓↓↓↓↓↓ 出力アダプタ定義 複数定義可能 ↓↓↓↓↓↓ -->
  <adp:OutputAdaptorDefinition name="OutputAdaptor1"
   charCode="MS932" lineFeed="CR_LF">

   <!-- 受信用CB定義 -->
   <cb:ReceiveCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.ReceiveCBImpl" name="receiver">
    <cb:streamInfo name="FILTER1" querygroup="Inprocess_QueryGroupTest"/>
   </cb:ReceiveCBDefinition>

   <!-- データ編集用CB定義 -->
   <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.OutputMappingCBImpl"
name="editor1">
    <!-- マッピング定義 -->
    <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>

   <!-- データ編集用CB定義 -->
   <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.formattranslate.OutputFormatTranslatorC
BImpl" name="editor3">
    <!-- フォーマット変換定義 -->
```

```
    <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>

   <!-- 出力用CB定義 -->
   <cb:OutputCBDefinition class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileOutputCBImpl"
name="outputer">
    <!-- 出力コネクタ定義 -->
    <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>
  <!-- ↑↑↑↑↑↑ 出力アダプタ定義 複数定義可能 ↑↑↑↑↑↑ -->

 </adp:InprocessGroupDefinition>
 <!-- ↑↑↑↑↑↑↑↑↑ インプロセスグループ定義 ↑↑↑↑↑↑↑↑↑ -->

</root:AdaptorCompositionDefinition>
```

(2)　記述例の内容

この記述例では，標準提供アダプターと SDP サーバは，インプロセスで連携します。入力アダプター「InputAdaptor1」でファイルを入力し，ストリームデータの集計・分析結果のデータは，出力アダプター「OutputAdaptor1」でファイルへ出力します。

入力アダプター「InputAdaptor1」では，次の表に示す処理を実施します。なお，括弧内は，アダプター構成定義ファイルでの定義名です。

コールバックの種類	コールバックでの処理
入力用コールバック（入力用 CB 定義）	ファイルの入力（ファイル入力コネクター定義）
編集用コールバック（編集用 CB 定義）	フォーマット変換（フォーマット変換定義）
	レコードとストリーム間のマッピング（マッピング定義）
送信用コールバック（送信用 CB 定義）	タプル送信（入力ストリーム定義）

入力アダプター「InputAdaptor1」の各定義の内容を次に示します。

- **ファイル入力コネクター定義の内容**

  入力データの格納先：C:¥temp¥input¥Inprocess¥

  ファイル名：Inprocess_Data1.csv

  読み込み処理モード：リアルタイム処理モード

  読み込み単位：一度に読み込むレコード数は 1 行

- **フォーマット変換定義の内容**

  ファイル入力コネクターで出力された入力形式レコードを共通形式レコードに変換します。

- **マッピング定義の内容**

  レコードとストリーム間のマッピングで，フォーマット変換で出力された共通形式レコードと，入力ストリームの形式に対応した共通形式レコードとを関連づけます。

- 入力ストリーム定義の内容
  マッピングで出力された共通形式レコードをタプルに変換して，クエリグループ「Inprocess_QueryGroupTest」の入力ストリーム「DATA0」に送信します。

出力アダプター「OutputAdaptor1」では，次の表に示す処理を実施します。なお，括弧内は，アダプター構成定義ファイルでの定義名です。

コールバックの種類	コールバックでの処理
受信用コールバック（受信用 CB 定義）	タプル受信（出力ストリーム定義）
編集用コールバック（編集用 CB 定義）	レコードとストリーム間のマッピング（マッピング定義）
	フォーマット変換（フォーマット変換定義）
出力用コールバック（出力用 CB 定義）	ファイルへの出力（ファイル出力コネクター定義）

出力アダプター「OutputAdaptor1」の各定義での定義内容を次に示します。

- 出力ストリーム定義の内容
  クエリグループ「Inprocess_QueryGroupTest」の出力ストリーム「FILTER1」からタプルを受信して，共通形式レコードに変換します。
- マッピング定義の内容
  レコードとストリーム間のマッピングで，出力ストリームの形式に対応した共通形式レコードと，フォーマット変換で入力する共通形式レコードとを関連づけます。
- フォーマット変換定義の内容
  マッピングで出力された共通形式レコードを出力形式レコードに変換します。
- ファイル出力コネクター定義の内容
  フォーマット変換で出力された出力形式レコードを次の条件でファイルに出力します。
  ファイル構成：ラップアラウンド構成
  出力データの格納先：C:¥temp¥output¥Inprocess¥
  ファイル名および出力順：out1.csv，out2.csv，…，out256.csv
- アダプター構成定義ファイルで指定するストリーム名
  アダプター構成定義ファイルで指定するストリーム名は，クエリ定義ファイルで指定した内容と合わせてください。
  - 入力ストリームの場合
    クエリ定義ファイルの register stream 句で指定したストリーム名「DATA0」が，アダプター構成定義ファイルで指定する入力ストリーム名となります。
  - 出力ストリームの場合
    クエリ定義ファイルの register query 句で指定したクエリ名「FILTER1」が，アダプター構成定義ファイルで指定する出力ストリーム名となります。

この記述例の場合のクエリ定義ファイルを次に示します。

```
register stream DATA0(name VARCHAR(10), num BIGINT);
register query FILTER1 ISTREAM(SELECT * FROM DATA0[ROWS 1] WHERE DATA0.NUM <= 24);
```

クエリの定義については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

9.13.2　記述例 2

ここでは，次のサンプルファイルに記述されているアダプター構成定義ファイルの記述例を示したあとに，記述例の内容を説明します。

＜インストールディレクトリ＞¥samples¥httppacket¥conf¥xml¥AdaptorCompositionDefinition.xml

(1)　アダプター構成定義ファイルの記述例

```
<?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"
 >

 <!-- 共通定義 -->
 <cmn:CommonDefinition>
  <!-- アダプタトレース定義 -->
  <cmn:AdaptorTraceDefinition trace="OFF"/>
 </cmn:CommonDefinition>

 <!-- ↓↓↓↓↓↓↓↓↓ インプロセスグループ定義 ↓↓↓↓↓↓↓↓↓ -->
 <!-- インプロセスグループ定義1 -->
 <adp:InprocessGroupDefinition name="InprocessAPTest">
  <!-- ↓↓↓↓↓↓↓ 入力アダプタ定義 複数定義可能 ↓↓↓↓↓↓↓ -->
  <!-- 入力アダプタ定義1 -->
  <adp:InputAdaptorDefinition name="InputAdaptor1" interval="0" charCode="MS932"
lineFeed="CR_LF">

   <!-- 入力用CB定義 -->
   <cb:InputCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.packetinput.HttpPacketInputCBImpl"
name="inputer">
    <!-- パケット入力コネクタ定義 -->
    <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 &amp;quot;tcp port 80 or port 8080&amp;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>

   <!-- データ編集用CB定義 -->
   <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.filter.FilterCBImpl" name="editor1">
    <!-- フィルター定義 -->
    <filter:FilterDefinition>
     <!-- レコード条件 -->
     <filter:record source="REQUEST" conditionName="filterName1" condition="AND">
      <!-- フィールド条件 -->
      <filter:field source="TARGET_URI" condition="eq" value="http:¥/¥/www.*"/>
     </filter:record>
    </filter:FilterDefinition>
   </cb:DataEditCBDefinition>

   <!-- データ編集用CB定義 -->
   <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.recordextract.RecordExtractionCBImpl"
name="editor2">

    <!-- レコード抽出定義 -->
    <rex:RecordExtractionDefinition>
     <!-- 抽出対象レコード群定義 -->
     <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>
     <!-- 抽出条件群定義 -->
     <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>
       <!-- 抽出レコード定義 -->
       <rex:extractrecord name="BINDRECORD">
        <rex:select source="element1"/>
        <rex:select source="element2"/>
       </rex:extractrecord>
      </rex:extraction>
     </rex:extractions>
    </rex:RecordExtractionDefinition>
   </cb:DataEditCBDefinition>

   <!-- データ編集用CB定義 -->
   <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor3">
    <!-- マッピング定義 -->
    <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>

   <!-- データ編集用CB定義 -->
   <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor4">
    <!-- マッピング定義 -->
    <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>

   <!-- 送信用CB定義 -->
   <cb:SendCBDefinition class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.SendCBImpl"
name="sender">
    <cb:streamInfo name="s1" querygroup="Inprocess_QueryGroupTest"/>
   </cb:SendCBDefinition>
  </adp:InputAdaptorDefinition>
  <!-- ↑↑↑↑↑↑ 入力アダプタ定義 複数定義可能 ↑↑↑↑↑↑ -->

  <!-- ↓↓↓↓↓↓ 出力アダプタ定義 複数定義可能 ↓↓↓↓↓↓ -->
  <adp:OutputAdaptorDefinition name="OutputAdaptor1" charCode="MS932" lineFeed="CR_LF">

   <!-- 受信用CB定義 -->
   <cb:ReceiveCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.ReceiveCBImpl" name="receiver">
    <cb:streamInfo name="q1" querygroup="Inprocess_QueryGroupTest"/>
   </cb:ReceiveCBDefinition>

   <!-- データ編集用CB定義 -->
   <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.OutputMappingCBImpl"
name="editor1">
    <!-- マッピング定義 -->
    <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>

    <!-- データ編集用CB定義 -->
    <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor2">
     <!-- マッピング定義 -->
     <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>

    <!-- 出力用CB定義 -->
    <cb:OutputCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.dashboard.DashboardOutputCBImpl"
name="outputer">
     <!-- ダッシュボード出力コネクタ定義 -->
     <docon:DashboardOutputConnectorDefinition Record="RECORD2">
      <docon:RecordHoldTime
       DateReference="LAST_UPDATE"
                     RecordTime="300"
                     DateFieldPosition="8" />
```

```
    </docon:DashboardOutputConnectorDefinition>
   </cb:OutputCBDefinition>
  </adp:OutputAdaptorDefinition>
  <!-- ↑↑↑↑↑↑ 出力アダプタ定義 複数定義可能 ↑↑↑↑↑↑ -->

 </adp:InprocessGroupDefinition>
 <!-- ↑↑↑↑↑↑↑↑ インプロセスグループ定義 ↑↑↑↑↑↑↑↑ -->

</root:AdaptorCompositionDefinition>
```

(2)　記述例の内容

この記述例では，標準提供アダプターと SDP サーバは，インプロセスで連携します。入力アダプター「InputAdaptor1」では，HTTP パケット情報の中から対応するリクエスト情報とレスポンス情報を結合し，それぞれが持っていた時刻情報の差から通信時間を算出します。出力アダプター「OutputAdaptor1」では，HTTP 通信情報を取得し，現在時刻から過去 5 分間の情報をダッシュボード出力コネクターで出力します。

入力アダプター「InputAdaptor1」では，次の表に示す処理を実施します。なお，括弧内は，アダプター構成定義ファイルでの定義名です。

コールバックの種類	コールバックでの処理
入力用コールバック（入力用 CB 定義）	HTTP パケットの入力（HTTP パケット入力コネクター定義）
編集用コールバック（編集用 CB 定義）	レコードのフィルタリング（フィルター定義）
	レコードの抽出（レコード抽出定義）
	レコード間のマッピング（マッピング定義）
	レコードとストリーム間のマッピング（マッピング定義）
送信用コールバック（送信用 CB 定義）	タプル送信（入力ストリーム定義）

入力アダプター「InputAdaptor1」の各定義の内容を次に示します。

- **HTTP パケット入力コネクター定義の内容**

  パケットアナライザーとして，WinDump を使用します。コマンドパラメーターには，解析対象コンピュータ（以降，サーバとします）の次の情報を指定します。

  - ネットワークデバイスの番号：1
  - HTTP プロトコルに用いるポート番号：80 または 8080

  HTTP パケット入力コネクターで，サーバが受信したパケット（リクエストパケット）と送信したパケット（レスポンスパケット）をレコード化します。各レコードでは，次のフィールドを保持します。

  - リクエストレコード

    送信元 IP アドレス，送信先 IP アドレス，送信元ポート番号，送信先ポート番号，URI 情報，およびサーバがパケットを受信した時刻

  - レスポンスレコード

    送信元 IP アドレス，送信先 IP アドレス，送信元ポート番号，送信先ポート番号，およびサーバがパケットを送信した時刻

- **フィルター定義の内容**

  リクエストレコードのうち，URI が「http:¥/¥/www.*」のレコードだけを取得します。なお，*は任意の 0 文字以上の文字列です。

- レコード抽出定義の内容

  次の条件に一致するリクエストレコードとレスポンスレコードを結合した抽出レコードを生成します。

  - リクエストレコードの送信元 IP アドレスが，レスポンスレコードの送信先 IP アドレスと等しい。
  - リクエストレコードの送信先 IP アドレスが，レスポンスレコードの送信元 IP アドレスと等しい。
  - リクエストレコードの送信元ポート番号が，レスポンスレコードの送信先ポート番号と等しい。
  - リクエストレコードの送信先ポート番号が，レスポンスレコードの送信元ポート番号と等しい。

- マッピング定義の内容

  - 1 個目のマッピング（レコード間のマッピング）で，レコードの抽出で結合されたレコードから，次のフィールドを保持するレコードを生成します。

    送信元 IP アドレス，送信先 IP アドレス，送信元ポート番号，送信先ポート番号，URI 情報（パラメーターを除く），通信応答時間（サーバがパケットを受信，および送信した時刻の差），およびサーバがパケットを送信した時刻

    なお，通信応答時間は，function 属性で subTime を指定して取得します。
  - 2 個目のマッピング（レコードとストリーム間のマッピング）で，1 個目のマッピングで出力された共通形式レコードと，入力ストリームの形式に対応した共通形式レコードとを関連づけます。

- 入力ストリーム定義の内容

  マッピングで出力された共通形式レコードをタプルに変換して，クエリグループ「Inprocess_QueryGroupTest」の入力ストリーム「s1」に送信します。

出力アダプター「OutputAdaptor1」では，次の表に示す処理を実施します。なお，括弧内は，アダプター構成定義ファイルでの定義名です。

コールバックの種類	コールバックでの処理
受信用コールバック（受信用 CB 定義）	タプル受信（出力ストリーム定義）
編集用コールバック（編集用 CB 定義）	レコードとストリーム間のマッピング（マッピング定義）
	レコード間のマッピング（マッピング定義）
出力用コールバック（出力用 CB 定義）	ダッシュボードへの出力（ダッシュボード出力コネクター定義）

出力アダプター「OutputAdaptor1」の各定義の内容を次に示します。

- 出力ストリーム定義の内容

  クエリグループ「Inprocess_QueryGroupTest」の出力ストリーム「q1」からタプルを受信して，共通形式レコードに変換します。

- マッピング定義の内容

  - 1 個目のマッピング（レコードとストリーム間のマッピング）で，出力ストリームの形式に対応した共通形式レコードと，次のコールバックの形式に対応した共通形式レコードとを関連づけます。
  - 2 個目のマッピング（レコード間のマッピング）で，1 個目のマッピングで出力されたレコードから，次のフィールドを保持するレコードを生成します。

    送信元 IP アドレス，送信先 IP アドレス，送信元ポート番号，送信先ポート番号，URI 情報（パラメーターを除く），通信応答時間（サーバがパケットを受信，および送信した時刻の差），サーバがパケットを送信した時刻，および出力ストリームから出力されたタプルの時刻

    なお，出力ストリームから出力されたタプルの時刻は，function 属性で getTupleTime を指定して取得します。

- ダッシュボード出力コネクター定義の内容

  マッピングで出力されたレコードを次の条件でダッシュボードに出力します。

  - ダッシュボード出力コネクターが最後にタプルを受信した時刻を基準時刻とする。
  - 基準時刻から 5 分（300 秒）以内のレコードを保持する。
  - タプルの時刻情報は，8 番目のフィールド（GET_TUPLE_TIME）にある。
  - RMI サーバのポート番号は「20421」を使用する（デフォルトのため，定義では指定しない）。

  なお, Dashboard Viewer でデータを表示するときの接続名は「InprocessAPTest/OutputAdaptor1/outputer」となります。

- アダプター構成定義ファイルで指定するストリーム名

  アダプター構成定義ファイルで指定するストリーム名は，クエリ定義ファイルで指定した内容と合わせてください。

  - 入力ストリームの場合

    クエリ定義ファイルの register stream 句で指定したストリーム名（s1）が，アダプター構成定義ファイルで指定する入力ストリーム名となります。

  - 出力ストリームの場合

    クエリ定義ファイルの register query 句で指定したクエリ名（q1）が，アダプター構成定義ファイルで指定する出力ストリーム名となります。

この記述例の場合のクエリ定義ファイルを次に示します。

```
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]);
```

クエリの定義については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

10 外部定義関数定義ファイル

この章では，外部定義関数を使用する場合に作成する，外部定義関数定義ファイルについて説明しています。

なお，外部定義関数定義ファイルの説明の記述形式，および作成上の注意事項は，アダプター用定義ファイルの場合と同じです。説明の記述形式，および作成上の注意事項については，「9.1　アダプター用定義ファイルの説明の記述形式」，および「9.2　アダプター用定義ファイル作成上の注意事項」を参照してください。

10.1　外部定義関数定義ファイル (ExternalFunctionDefinition.xml)

外部定義関数定義ファイルでは，外部定義関数の関数名や戻り値と，ユーザーが Java で作成したクラスとの関連づけについて定義します。

ここでは，外部定義関数定義ファイルの概要と，外部定義関数定義ファイルの名前空間 URI について説明します。定義ファイルの各定義の詳細は，次の節以降で説明します。また，定義ファイルの記述例については，「10.4　外部定義関数定義ファイルの記述例」で説明します。

10.1.1　外部定義関数定義ファイル (ExternalFunctionDefinition.xml) の概要

(1)　記述形式

記述形式は，次の節以降で，定義ごとに説明します。

(2)　ファイル名

ExternalFunctionDefinition.xml

(3)　ファイルの格納先

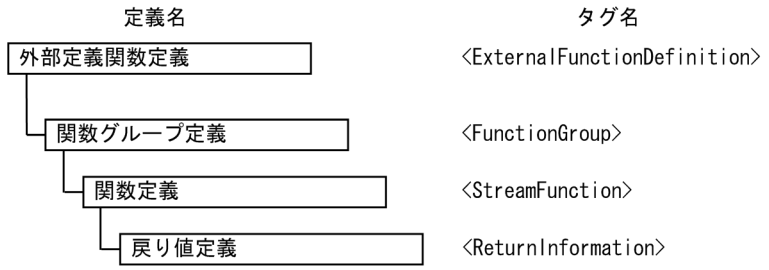
このファイルは，必ず次のディレクトリに格納してください。

＜運用ディレクトリ＞¥conf¥xml¥

(4)　定義の詳細

外部定義関数定義ファイルの定義は，階層構造になっています。外部定義関数定義ファイルの構造を次の図に示します。

図 10-1　外部定義関数定義ファイルの構造



階層構造になっている定義は，親子関係にあります。例えば，関数定義は，関数グループ定義の**子要素**として，関数グループ定義のタグ内に定義します。また，関数グループ定義は，関数定義の**親要素**となります。

外部定義関数定義ファイルの定義の一覧を次の表に示します。それぞれの定義の詳細は，表中の参照先で説明します。

表 10-1　外部定義関数定義ファイルの定義の一覧

項番	定義	説明	参照先
1	関数グループ定義 (FunctionGroup タグ)	外部定義関数の関数名や戻り値と，ユーザーが Java で作成したクラスとの関連づけを定義します。	10.2
2	関数定義 (StreamFunction タグ)		10.3
3	戻り値定義 (ReturnInformation タグ)		

10.1.2　外部定義関数定義ファイルの名前空間 URI

外部定義関数定義ファイルでは，**名前空間 URI** は，次の二つを宣言してください。

```
http://www.hitachi.co.jp/soft/xml/sdp/function
http://www.hitachi.co.jp/soft/xml/sdp/function/functiongroup
```

10.2　外部定義関数定義ファイルの関数グループ定義

ここでは，外部定義関数定義ファイル（ExternalFunctionDefinition.xml）の関数グループ定義について説明します。関数グループ定義の一覧を次の表に示します。

表 10-2　関数グループ定義の一覧

項番	関数グループ定義	親要素	参照先
1	関数グループ定義 （FunctionGroup タグ）	外部定義関数定義	10.2.1

10.2.1　関数グループ定義

関数グループ定義（FunctionGroup タグ）では，外部定義関数の処理内容が実装されているクラスを定義するグループ（関数グループ）を定義します。

関数グループ定義は，64 個まで記述できます。この定義は省略できます。

(1)　記述形式

```
<FunctionGroup name=”＜関数グループ名＞”
  path=”＜クラスパス＞”>
  ＜関数定義＞
</FunctionGroup>
```

(2)　定義の詳細

FunctionGroup タグ（全体情報の定義）

関数グループ定義の全体情報を定義します。

name="＜関数グループ名＞"

関数グループを識別するための名称を 1～100 文字の半角英数字，またはアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。CQL の予約語と同じ名前は使用できません。この属性は省略できません。関数グループ名は，大文字小文字の区別なく，外部定義関数定義内で一意となるように指定してください。

path="＜クラスパス＞"

外部定義関数の処理内容が実装されているクラスのクラスパスを，jar ファイルのパス，またはディレクトリのパスで指定します。指定できるクラスパスは 1 個です。この属性は省略できません。クラスパスを相対パスで指定する場合，運用ディレクトリからの相対パスで記述してください。パスの区切り文字，およびパスの大文字小文字の区別の有無は，OS ごとに異なります。

＜関数定義＞

関数定義については，「10.3　外部定義関数定義ファイルの関数定義」を参照してください。

10.3　外部定義関数定義ファイルの関数定義

ここでは，外部定義関数定義ファイル（ExternalFunctionDefinition.xml）の関数定義について説明します。関数定義の一覧を次の表に示します。

表 10-3　関数定義の一覧

項番	関数定義	親要素	参照先
1	関数定義 （StreamFunction タグ）	関数グループ定義	10.3.1
2	戻り値定義 （ReturnInformation タグ）	関数定義	10.3.2

10.3.1　関数定義

関数定義（StreamFunction タグ）では，外部定義関数の関数名とユーザーが Java で作成したクラスの関連づけを定義します。

この定義は関数グループ定義ごとに 16 個まで記述できます。この定義は省略できません。

(1)　記述形式

```
<StreamFunction name=”＜関数名＞”
  class=”＜クラス名＞”>
  ＜戻り値定義＞
</StreamFunction>
```

(2)　定義の詳細

StreamFunction タグ（全体情報の定義）

関数定義の全体情報を定義します。

name="＜関数名＞"

外部定義関数を識別するための名称を 1～100 文字の半角英数字，またはアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。CQL の予約語と同じ名前は使用できません。この属性は省略できません。関数名は，大文字小文字の区別なく，同一の関数グループ定義内で一意となるように指定してください。

class="＜クラス名＞"

外部定義関数の処理内容が実装されているクラス名を指定します。クラスをパッケージ化している場合は，パッケージ名を含んだ形式で指定します。この属性は省略できません。
「jp.co.Hitachi.soft.sdp」から始まるパッケージ名は指定できません。クラス名は，大文字小文字を区別して，同一の関数グループ定義内で一意となるように指定してください。

＜戻り値定義＞

戻り値定義については，「10.3.2　戻り値定義」を参照してください。

10.3.2　戻り値定義

戻り値定義（ReturnInformation タグ）では，外部定義関数の処理結果となる出力ストリームの各列の列名および型名を定義します。

この定義は関数定義ごとに 3,000 個まで記述できます。この定義は省略できません。

(1)　記述形式

```
<ReturnInformation name=”＜列名＞”
  type=”＜型名＞”>
</ReturnInformation >
```

(2)　定義の詳細

ReturnInformation タグ（全体情報の定義）

戻り値定義の全体情報を定義します。

name="＜列名＞"

外部定義関数の処理結果となる出力ストリームの列を識別するための名称を 1～100 文字の半角英数字，またはアンダーライン（_）で指定します。先頭の文字に指定できるのは，半角英字だけです。CQL の予約語と同じ名前は使用できません。この属性は省略できません。列名は，大文字小文字の区別なく，同一の関数定義内で一意となるように指定してください。

type="＜型名＞"

外部定義関数の処理結果となる出力ストリームの列に対応する型名を大文字で指定します。型名に指定できるのは，CQL のデータ型です。

10.4　外部定義関数定義ファイルの記述例

ここでは，次のサンプルファイルに記述されている外部定義関数定義ファイルの記述例を示したあとに，記述例の内容を説明します。

＜インストールディレクトリ＞¥samples¥external¥conf¥xml¥ExternalFunctionDefinition.xml

外部定義関数定義ファイルの記述例

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2013, Hitachi, Ltd.  -->
<root:ExternalFunctionDefinition
 xmlns:root="http://www.hitachi.co.jp/soft/xml/sdp/function"
  xmlns:group="http://www.hitachi.co.jp/soft/xml/sdp/function/functiongroup">

 <!--関数グループ定義 -->
 <group:FunctionGroup name="FG1" path="samples/external/src">
  <!-- 関数定義 -->
  <group:StreamFunction name="FUNC1" class="samples.ExternalStreamFunction">
   <!-- 戻り値定義 -->
   <group:ReturnInformation name="R1" type="INT" />
   <group:ReturnInformation name="R2" type="VARCHAR(10)" />
   <group:ReturnInformation name="R3" type="BIGINT" />
   <group:ReturnInformation name="R4" type="TIMESTAMP(9)" />
  </group:StreamFunction>
 </group:FunctionGroup>

</root:ExternalFunctionDefinition>
```

記述例の内容

この記述例では，関数グループ名「FG1」，関数名「FUNC1」の外部定義関数を定義しています。

各定義の内容を次に示します。

- 関数グループ定義の内容

  関数グループ名：FG1

  クラスパス：samples/external/src

- 関数定義の内容

  関数名：FUNC1

  クラス名：samples.ExternalStreamFunction

- 戻り値定義の内容

  外部定義関数の処理結果となる出力ストリームの各列の列名および型名を，次の表に示すとおりに定義しています。

出力ストリームの列	列名	型名
1列目	R1	INT
2列目	R2	VARCHAR(10)
3列目	R3	BIGINT
4列目	R4	TIMESTAMP(9)

この記述例の場合のクエリ定義ファイルを次に示します。

```
REGISTER STREAM DATA0(ID VARCHAR(10), VAL BIGINT);

REGISTER QUERY Q1 SELECT ID, VAL FROM DATA0[PARTITION BY ID ROWS 1];
REGISTER QUERY Q2 ISTREAM (SELECT * FROM Q1);
REGISTER QUERY Q3 DSTREAM (SELECT * FROM Q1);
REGISTER QUERY SUM1 $*FG1.FUNC1[3](Q2,Q3);
```

クエリの定義については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

11 定義ファイルでの定義内容の詳細

この章では，「8. SDP サーバ用定義ファイル」および「9. アダプター用定義ファイル」の定義ファイルで設定するデータの入出力やデータ編集の機能について，各処理の概要や処理の流れ，および必要な設定を説明しています。

11.1　この章の構成

この章では，標準提供アダプターでのデータ処理の概要や流れ，およびデータ処理をするために必要な定義ファイルの設定について説明します。また，標準提供アダプターまたはカスタムアダプターで，SDP サーバがタプルのタイムスタンプを調整するタイムスタンプ調整についても説明します。

この章で説明する機能の一覧，および参照先を次の表に示します。

表 11-1　機能の一覧および参照先

項番	機能		入力アダプター	出力アダプター	参照先
1	標準提供アダプターで使用できる機能	ファイルの入力	○	×	11.2
2		HTTP パケットの入力	○	×	11.3
3		レコードのフィルタリング	○	○	11.4
4		レコードの抽出	○	×	11.5
5		ファイルへの出力	×	○	11.6
6		ダッシュボードへの出力	×	○	11.7
7	タプルのタイムスタンプの調整		○	×	11.8

（凡例）

○：使用できます。

×：使用できません。

11.2　ファイルの入力

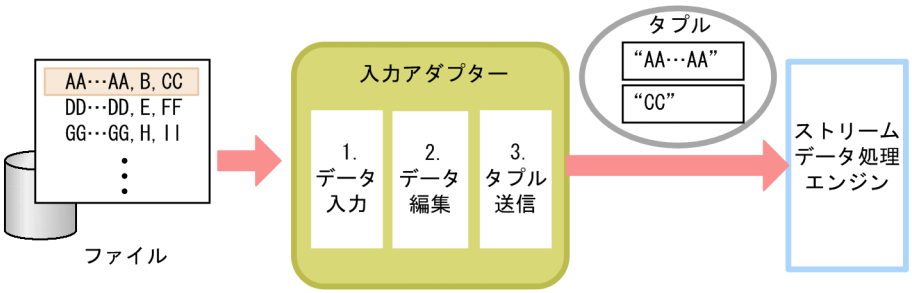
ここでは，ファイル入力の概要，データ処理の流れ，および定義ファイルでの設定内容について説明します。

11.2.1　ファイルの入力の概要

ファイルの入力では，エラーログやアクセスログなどのファイルからデータを読み込み，読み込んだデータをストリームデータ処理エンジンで処理できる形式に変換してからストリームデータ処理エンジンに送信します。ファイルの入力の処理は入力アダプターで実施します。

ファイルの入力の概要を次の図に示します。

図 11-1　ファイルの入力の概要

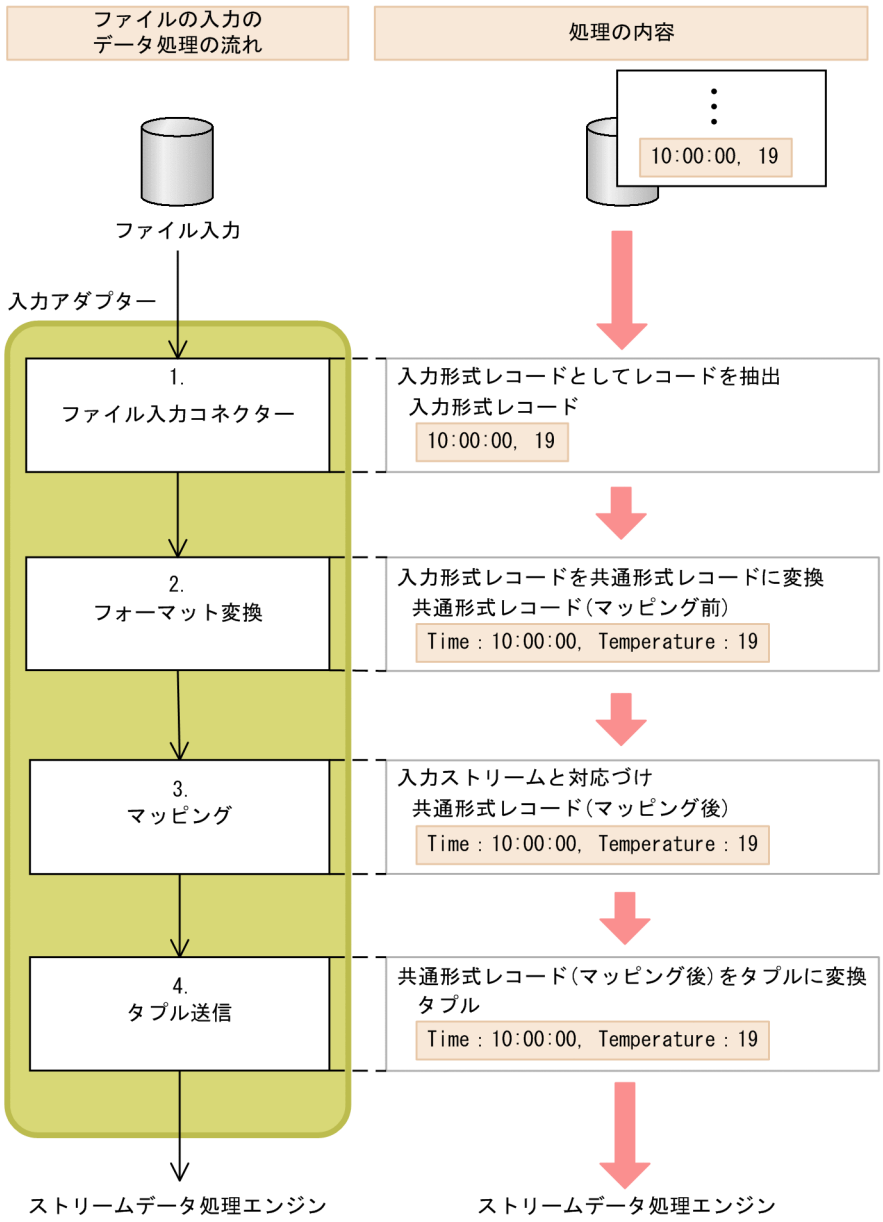


1. データ入力
   ファイル入力コネクターで，入力ファイルからデータを読み込みます。
2. データ編集
   読み込んだデータのフォーマット変換およびマッピングをして，データを編集します。
3. タプル送信
   編集したデータをタプルに変換して，ストリームデータ処理エンジンに送信します。

11.2.2　ファイルの入力のデータ処理の流れ

入力アダプターでのファイルの入力のデータ処理の流れと処理の内容を次の図に示します。

図 11-2　ファイルの入力のデータ処理の流れと処理の内容



入力アダプターで扱うデータ形式について，「(1)　入力アダプターで扱うデータ形式」で説明します。また，各処理の詳細について，「(2)　ファイル入力コネクターによるファイルの入力」以降で説明します。

(1)　入力アダプターで扱うデータ形式

入力アダプターで扱うデータ形式には，入力形式レコード，および共通形式レコードがあります。それぞれのデータ形式について説明します。

入力形式レコード

入力ファイルから取得した，行単位のデータのことです。入力アダプターでは，1 行のデータを 1 入力形式レコードとして扱います。

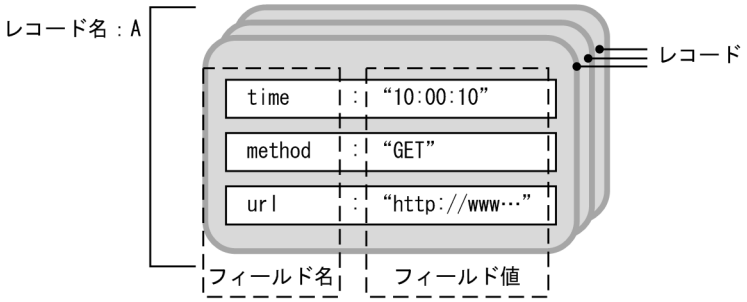
共通形式レコード

複数のフィールド名とその値であるフィールド値から成る，データの集合のことです。フィールド名とは，入力形式レコードから意味のある単位（フィールド）に切り出したデータに付けた名前で，フィールド値とはその値のことを指します。

共通形式レコードは，入力アダプターおよび出力アダプター内部で扱う共通のデータ形式です。

標準提供アダプターでは，共通形式レコードの複数のフィールド名とそれぞれに対応するフィールド値の組み合わせをレコード構成として管理し，レコード名で識別します。

図 11-3　共通形式レコードの構成



(2)　ファイル入力コネクターによるファイルの入力

ファイル入力についての定義は，アダプター構成定義ファイルのファイル入力コネクター定義で定義します。

ファイル入力コネクターでは，入力ファイル格納ディレクトリに格納されている一つ以上のファイルから行単位でデータを読み込んで，入力形式レコードに変換します。ファイル入力コネクターで読み込んだ 1 行のデータが，1 入力形式レコードとなります。

なお，入力アダプターでは複数の入力形式レコードを一度に読み込むことができます。ファイル入力コネクターで読み込んだ入力形式レコード数が，フォーマット変換，マッピング，タプル送信の各処理で一度に処理するレコード数となります。

ここでは，ファイル入力コネクターで読み込む入力ファイルの種類や構成などについて説明します。

ファイルの種類

ファイル入力コネクターで読み込む入力ファイルは，文字データだけで構成されたテキストファイルです。レコードは可変長です。

ファイル構成

ファイル入力コネクターで読み込む入力ファイルは，次のファイル構成のファイルです。

ファイル構成	説明	前提条件
ラップアラウンド	入力対象のファイル数が固定されている場合に，設定されたファイルの順番に情報を書き込むファイル構成です。すべてのファイルの書き込みが完了すると，最初のファイルを更新して書き込みます。	情報を書き込むファイルの順番が決まっていて，必ずその順番で書き込みます。 書き込みが完了したファイルに書き込む場合は，データをクリアしてから新しい情報を書き込みます。
非ラップアラウンド	入力対象のファイル数が固定されていない場合に，設定されたファイルの順番に情報を書き込むファイル構成です。	情報を書き込むファイルの順番が決まっていて，必ずその順番で書き込みます。

また，レコードおよびファイルの生成順序は時系列である必要があります。

ファイル名

ファイル入力コネクターで読み込むファイルは，ファイル入力コネクター定義の file タグの name 属性で，ファイル名または通番で指定します。name 属性については，「9.10.1　ファイル入力コネクター定義」を参照してください。

ファイルを読み込む順番

ファイル入力コネクターは，ファイル入力コネクター定義の file タグの name 属性で指定したファイル名の順番，または入力ファイルの更新時刻の順番で入力ファイルを読み込みます。ファイルの読み込み順は，input タグの readOrder 属性で指定します。

なお，入力ファイルの更新時刻が等しい場合，読み込みの順番は保証されません。

ファイル読み込み処理モード

ファイル入力コネクターがファイルを読み込む方式には，**バッチ処理モード**と**リアルタイム処理モード**があります。それぞれの処理モードについて次の表に示します。なお，どちらの処理モードの場合も，読み込みが完了すると入力アダプターが自動で停止します。

処理モード	読み込み処理の内容	
	読み込み方式	読み込み完了条件
バッチ処理モード	入力アダプターの起動時に，入力ファイル格納ディレクトリに入力ファイルがあるかどうかを確認して，入力ファイルがあった場合に読み込み処理をします。	次の条件のどちらかに一致した時点で，読み込みを完了します。 • 入力アダプターの起動時に，入力ファイル格納ディレクトリにあった，すべてのファイルの読み込みが完了したとき。 • 定義に指定したすべてのファイルの読み込みが完了したとき。
リアルタイム処理モード	入力ファイル格納ディレクトリを一定間隔で監視し，入力ファイルが出現した時点で，逐次読み込み処理をします。 入力ファイル格納ディレクトリの監視間隔，および監視回数はファイル入力コネクター定義の input タグで定義します。監視回数を超えた場合は，警告メッセージを出力したあと，処理を続行します。	次の条件のどちらかに一致した時点で，読み込みを完了します。 • 入力ファイルを通番で指定する場合：通番の最後に当たるファイルの読み込みが完了したとき。 • 入力ファイルをファイル名で指定する場合：定義に指定したすべてのファイルの読み込みが完了したとき。

ファイル入力コネクターの動作中に，入力ファイルを入力ファイル格納ディレクトリにコピーまたは移動中の場合，ファイル入力コネクターは 1 秒間待機したあと，再度ファイルの読み込み処理をします。この処理は読み込みができるまで，最大 5 回繰り返し，読み込みができない場合はメッセージ KFSP46200-E が出力されます。

また，入力ファイル格納ディレクトリで入力ファイルを新規作成，または入力ファイルに追加の書き込みがあった場合，新規作成または追加書き込みしたレコードに読み込まれません。

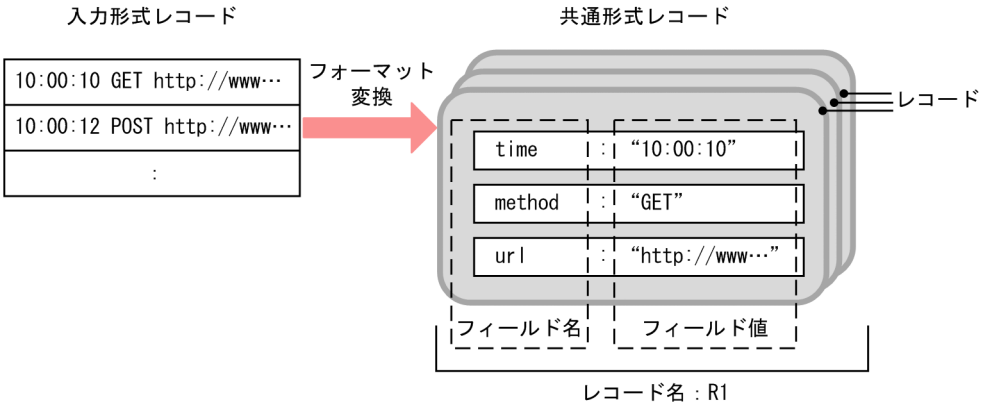
(3)　フォーマット変換

フォーマット変換についての定義は，アダプター構成定義ファイルのフォーマット変換定義で定義します。

フォーマット変換では，入力形式レコードから意味のある単位（フィールド）に切り出して，共通形式レコードに変換します。

入力形式レコードから共通形式レコードへのフォーマット変換の例を次の図に示します。この例では，入力形式レコードは，空白で区切られた三つのフィールドで構成され，第 1 フィールドは TIME 型，第 2，3 フィールドは文字列型にフォーマット変換されます。

図 11-4　入力形式レコードから共通形式レコードへのフォーマット変換の例



この図の場合の共通形式レコードのレコード構成と指定するフォーマット変換定義のタグを次の表に示します。

レコード構成	タグ
レコード名：R1 レコード構成：($_time) △ ($_method) △ ($_url)	record タグ（レコード定義）
フィールド名：time，型：TIME	field タグ（フィールド定義）
フィールド名：method，型：STRING	
フィールド名：url，型：STRING	

（凡例）

△：半角スペース

変換できるデータ型および共通形式レコードのレコード構成の設定については，「9.11.1　フォーマット変換定義」を参照してください。

なお，フォーマット変換では，複数のレコード構成を定義できます。複数のレコード構成が定義された場合は，どのレコード構成に該当するかを自動で選択してフォーマット変換します。

複数のレコード構成が定義された場合，レコード構成は次のように選択されます。

1. ファイル入力コネクターで読み込んだ入力形式レコードの構成が，フォーマット変換定義の record タグで指定したレコード構成に一致するかどうかを，レコード構成を定義した順にチェックします。一致した最初のレコード構成を選択します。
2. 1.で一致するレコード構成がない場合，入力形式レコードは破棄されます。この場合，入力アダプターはフォーマット変換定義の unmatchedFormat タグで定義された，次のどれかの処理をします。
   - 処理を続行します。
   - 警告メッセージを出力し，処理を続行します。
   - エラーメッセージを出力し，入力アダプターを停止します。

参考

フォーマット変換のあとには，レコードのフィルタリング，レコードの抽出，およびレコード間のマッピングが順不同で実施できます。必要に応じて実施してください。

レコードのフィルタリングについては，「11.4　レコードのフィルタリング」を参照してください。レコードの抽出については，「11.5　レコードの抽出」を参照してください。また，レコード間のマッピングについては，「(4)　マッピング」を参照してください。

(4)　マッピング

マッピングについての定義は，アダプター構成定義ファイルのマッピング定義で定義します。

マッピングには，**レコードとストリーム間のマッピング**，および**レコード間のマッピング**があります。それぞれのマッピングの概要を次の表に示します。

表 11–2　マッピングの概要

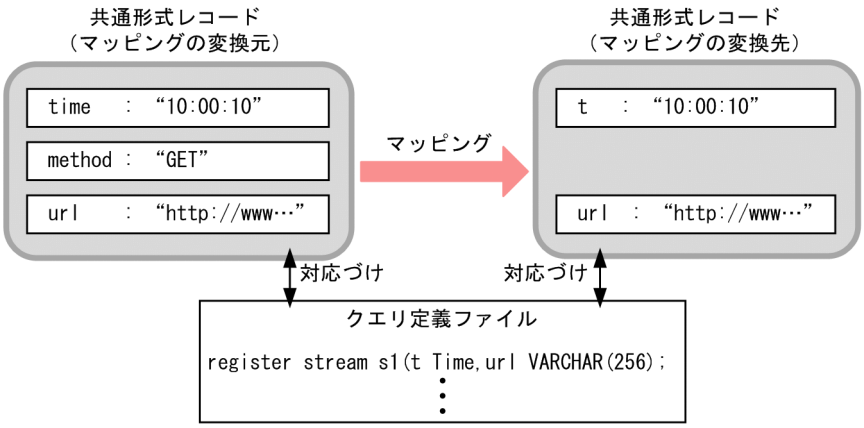
項番	マッピングの種類	説明
1	レコードとストリーム間のマッピング	マッピングの前のコールバックで出力される共通形式レコード（マッピングの変換元）と，入力ストリームの形式に従った共通形式レコード（マッピングの変換先）との関連づけをします。 レコードとストリーム間のマッピングは，タプル送信の前に必ず実施します。
2	レコード間のマッピング	マッピングの前のコールバックで出力される共通形式レコード（マッピングの変換元）を編集し，マッピングの変換先の共通形式レコードに変換します。 レコード間のマッピングは，フォーマット変換のあと，かつレコードとストリーム間のマッピングの前に，必要に応じて実施します。 マッピングの変換元の共通形式レコードのフィールド名を変更したい場合や，複数のフィールドから次のコールバックの処理に不要なフィールドを削除したい場合などに使用できます。 また，組み込み関数※を使用して，マッピングの変換元の共通形式レコードから文字列や時刻を取得し，取得した文字列や時刻をマッピングの変換先の共通形式レコードに反映することもできます。 なお，レコード間のマッピングは複数定義できます。

注※

レコード間のマッピングで使用できる組み込み関数は，アダプター構成定義ファイルの map タグの function 属性で指定します。function 属性で指定できる組み込み関数については，「9.11.2　マッピング定義」を参照してください。

入力アダプターでのレコードとストリーム間のマッピングの例を次の図に示します。この例では，入力ストリーム s1 に必要な time フィールドと url フィールドを入力ストリームのスキーマにマッピングして，共通形式レコード（マッピングの変換先）に変換しています。

図 11-5　入力アダプターでのレコードとストリーム間のマッピングの例



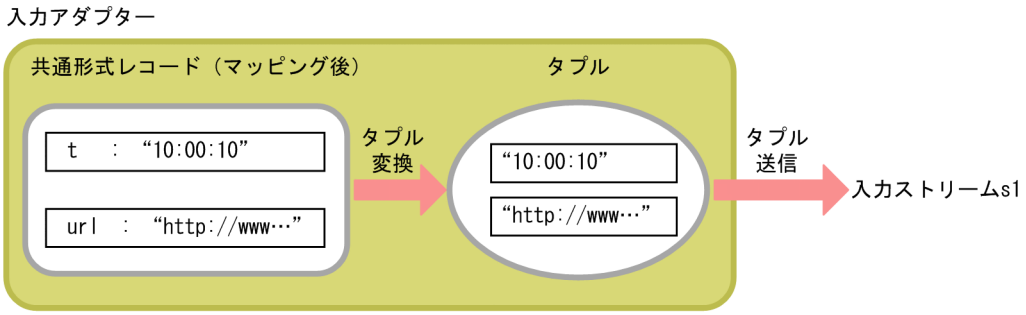
(5)　タプル送信

タプル送信についての定義は，アダプター構成定義ファイルの入力ストリーム定義に定義します。

タプル送信では，マッピング結果を基に，共通形式レコードをタプルに変換したあとで，入力ストリーム定義に従って入力ストリームにタプルを送信します。

入力アダプターから入力ストリームへのタプル送信の例を次の図に示します。この例では，入力ストリーム s1 にタプルを送信します。

図 11-6　入力アダプターからのタプル送信の例



11.2.3　ファイルの入力の設定

ファイル入力について設定が必要な定義ファイルは，アダプター構成定義ファイルです。アダプター構成定義ファイルの入力アダプター定義で，次の CB 定義に設定します。

- **入力用 CB 定義**

  ファイル入力コネクター定義に，入力コネクターの情報を定義します。定義の詳細については，「9.9.1　入力用 CB 定義」，および「9.10.1　ファイル入力コネクター定義」を参照してください。

- **編集用 CB 定義**

  フォーマット変換定義に，入力データのデータ形式を定義します。マッピング定義には，マッピング情報を定義します。定義の詳細については，「9.9.3　編集用 CB 定義」，「9.11.1　フォーマット変換定義」，または「9.11.2　マッピング定義」を参照してください。

- **送信用 CB 定義**

入力ストリーム定義に，入力アダプターが接続する入力ストリームの情報を定義します。定義の詳細については，「9.9.4　送信用 CB 定義」，および「9.12.1　入力ストリーム定義」を参照してください。

11.3　HTTP パケットの入力

ここでは，HTTP パケットの入力の概要，データ処理の流れ，および定義ファイルでの設定内容について説明します。

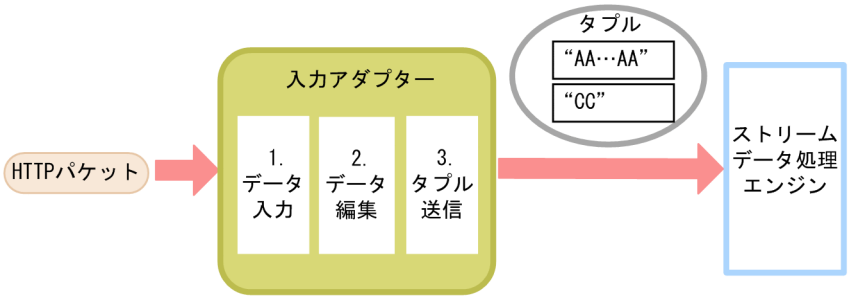
11.3.1　HTTP パケットの入力の概要

HTTP パケットの入力では，HTTP パケットデータを標準入力で取得し，取得したデータを解析して，ストリームデータ処理エンジンで処理できる形式に変換してからストリームデータ処理エンジンに送信します。

HTTP パケットの入力の処理は入力アダプターで実施します。

HTTP パケットの入力の概要を次の図に示します。

図 11-7　HTTP パケットの入力の概要



1. データ入力

   HTTP パケット入力コネクターで，パケットアナライザーから標準出力されたデータを読み込みます。読み込んだ HTTP パケットデータを解析して，データを変換します。

2. データ編集

   マッピングをして，データを編集します。

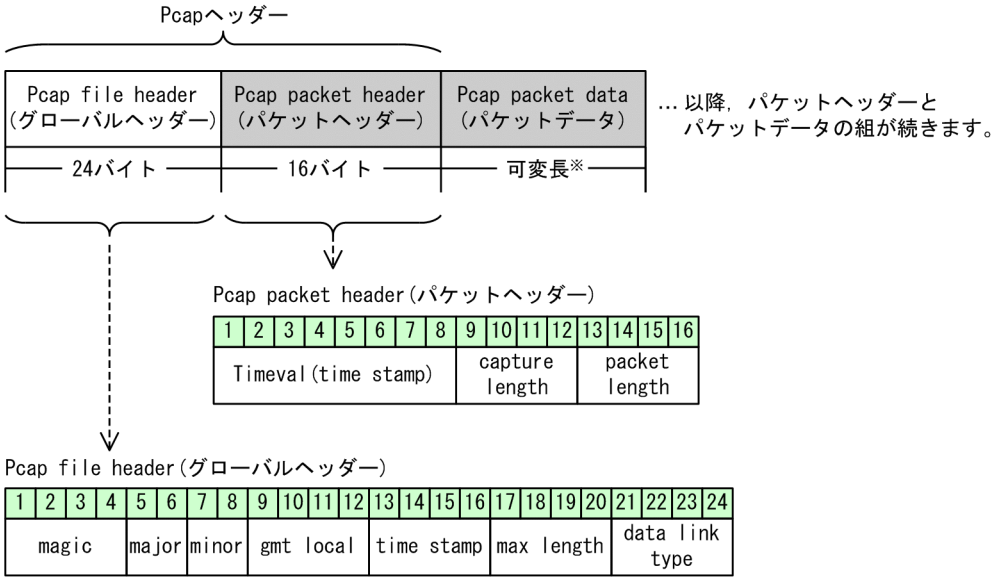
3. タプル送信

   編集したデータをタプルに変換して，SDP サーバに送信します。

なお，HTTP パケットを入力するには，Pcap 形式でパケットを出力できるパケットアナライザーが必要です。Stream Data Platform - AF では，Windows の場合は WinDump，Linux の場合は tcpdump を使用できます。パケットアナライザーのバージョンについては，Stream Data Platform - AF のリリースノートを参照してください。

HTTP パケットの入力で扱えるのは，次の図に示す Pcap 形式のフォーマットで出力される HTTP パケットです。

図 11-8　Pcap 形式のフォーマット



注※　サイズはパケットヘッダーから取得します。

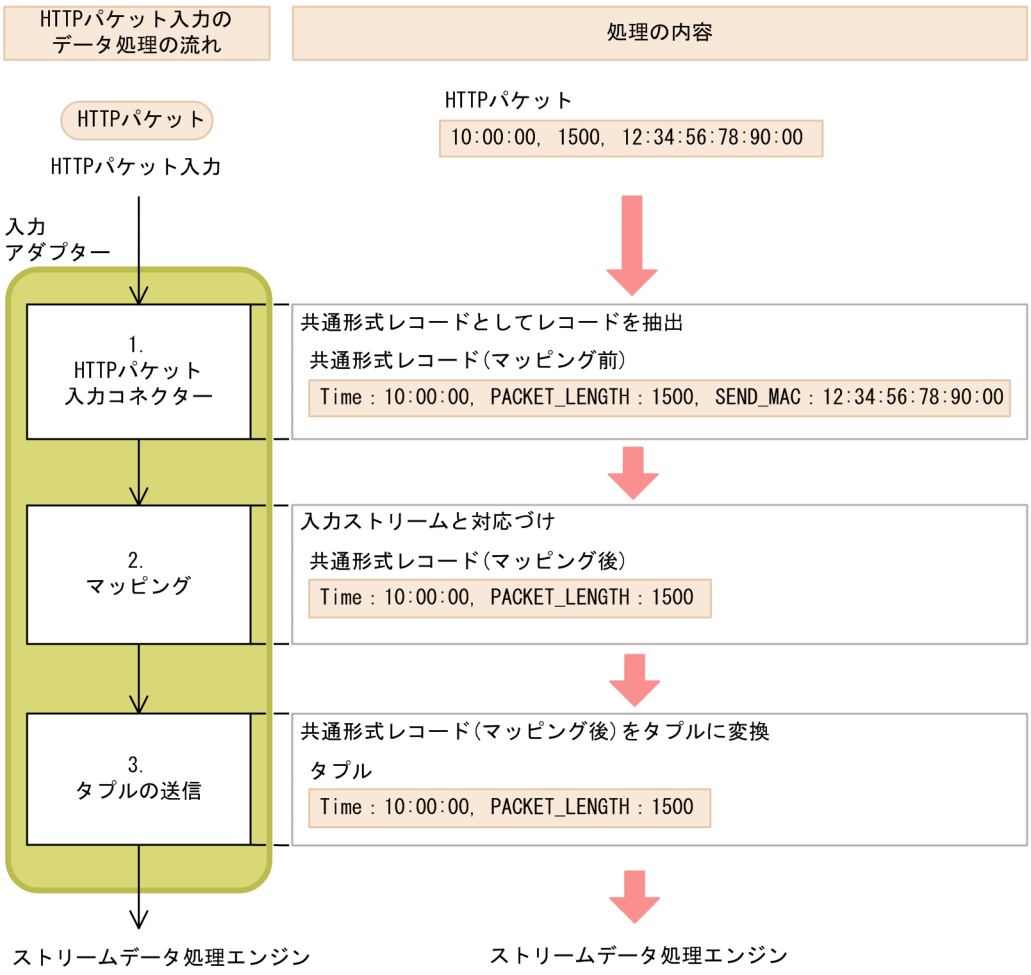
Pcap 形式のフォーマットでは，ネットワークを流れるパケット部分に Pcap ヘッダーが付加されています。Pcap ヘッダーのうち，Pcap file header は，パケットアナライザー起動後，最初に作成されるグローバルヘッダーです。グローバルヘッダーが作成されたあと，パケットヘッダーである Pcap packet header とパケットデータである Pcap packet data を 1 組としたデータが作成され続けます。

Pcap 形式のフォーマット，またはそれ以外の HTTP パケットのフォーマットに関する情報は，アダプター構成定義ファイルの HTTP パケット入力コネクター定義に定義します。

11.3.2　HTTP パケットの入力のデータ処理の流れ

入力アダプターでの HTTP パケットの入力の，データ処理の流れと処理の内容を次の図に示します。

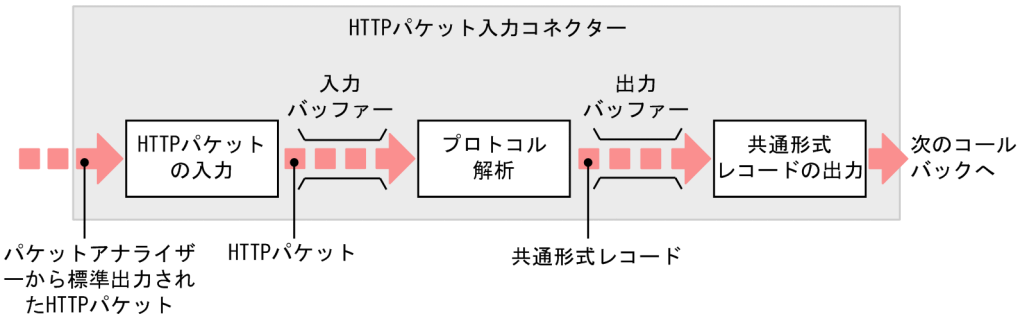
図 11-9　HTTP パケットの入力のデータ処理の流れと処理の内容



ここでは，HTTP パケット入力コネクターによる HTTP パケットの入力について説明します。マッピング，およびタプル送信については，「11.2.2(4)　マッピング」および「11.2.2(5)　タプル送信」を参照してください。入力アダプターで扱うデータ形式については，「11.2.2(1)　入力アダプターで扱うデータ形式」を参照してください。

HTTP パケット入力コネクターでの HTTP パケット入力の処理の概要を次の図に示します。

図 11-10　HTTP パケット入力コネクターでの HTTP パケットの入力の処理の概要



1. HTTP パケットの入力

HTTP パケット入力コネクターでは，パケットアナライザーから標準出力された HTTP パケットを標準入力で取得します。取得した HTTP パケットは入力バッファーに格納されます。

2. **プロトコル解析**

   格納されたデータのプロトコルを解析して，解析したプロトコルデータから共通形式レコードに変換します。なお，HTTP パケットの入力とプロトコル解析の処理は非同期で行われます。変換された共通形式レコードは出力バッファーに格納されます。

3. **共通形式レコードの出力**

   出力バッファーに格納された共通形式レコードを次のコールバックに出力します。

11.3.3　HTTP パケットの入力の設定

HTTP パケットの入力について設定が必要な定義ファイルは，アダプター構成定義ファイルです。アダプター構成定義ファイルの入力アダプター定義で，次の CB 定義に設定します。

- **入力用 CB 定義**

  HTTP パケット入力コネクター定義に，HTTP パケット入力コネクターの情報を定義します。定義の詳細については，「9.9.1　入力用 CB 定義」，および「9.10.2　HTTP パケット入力コネクター定義」を参照してください。

- **編集用 CB 定義**

  マッピング定義に，マッピング情報を定義します。定義の詳細については，「9.9.3　編集用 CB 定義」，および「9.11.2　マッピング定義」を参照してください。

- **送信用 CB 定義**

  入力ストリーム定義に，入力アダプターが接続する入力ストリームの情報を定義します。定義の詳細については，「9.9.4　送信用 CB 定義」，および「9.12.1　入力ストリーム定義」を参照してください。

11.4　レコードのフィルタリング

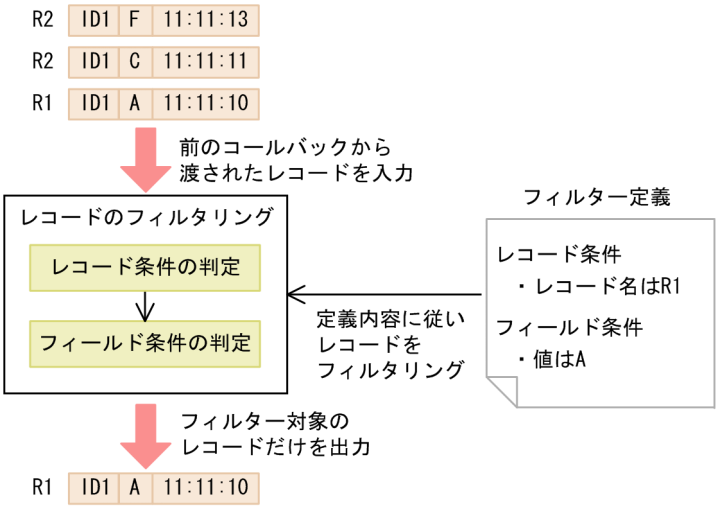
ここでは，レコードのフィルタリングの概要，データ処理の流れ，および必要な設定について説明します。

11.4.1　レコードのフィルタリングの概要

特定のレコードだけをストリームデータ処理の対象にしたい場合，編集用コールバックとして，フィルターを使用します。

レコードのフィルタリングの概要を次の図に示します。

図 11-11　レコードのフィルタリングの概要



レコードのフィルタリングでは，アダプター構成定義ファイルのフィルター定義で定義した内容に従って，入力されたレコードが条件に一致するかどうかを判定します。レコード形式，およびレコードに設定されているフィールド値から目的のレコードを取捨選択して出力します。レコードのフィルタリングの処理の詳細については，「11.4.2　レコードのフィルタリングのデータ処理の流れ」で説明します。

フィルターの入出力データの形式

フィルターの入出力データの形式は，共通形式レコードです。

フィルターを実施するタイミング

フィルターは，入力アダプター，または出力アダプターの編集用コールバックとして実施します。

入力アダプターの場合は，入力コネクターによるデータ入力またはフォーマット変換のあとにフィルターを実施します。出力アダプターの場合は，タプル受信後のマッピングのあとにフィルターを実施します。

編集用コールバックを実施するタイミングについては，「2.5.2　データの編集方法の検討」を参照してください。

11.4.2　レコードのフィルタリングのデータ処理の流れ

レコードのフィルタリングでは，レコード条件の判定，およびフィールド条件の判定を実施します。レコード条件の判定，およびフィールド条件の判定には，次の内容を指定します。

- **レコード条件**

  record タグで，フィルター対象のレコード名を指定します。

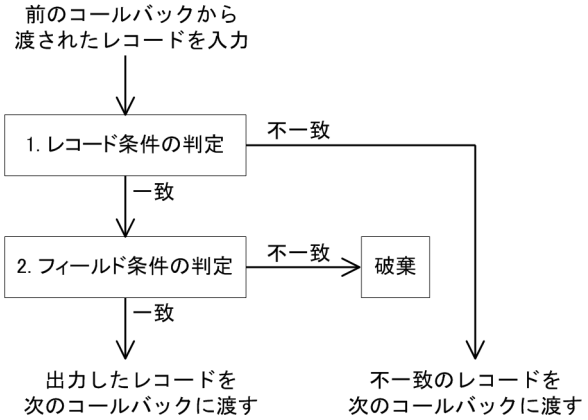
- フィールド条件

  field タグで，フィルター対象のレコードのフィールド名，比較演算記号，および条件値を指定します。

例えば，レコード名が R1，フィールド名が F1，フィールド値が 100 よりも大きいという条件を指定したい場合には，レコード条件のレコード名には「source="R1"」，フィールド条件には「source="F1" condition="gt" value="100"」と指定します。

レコードのフィルタリングのデータ処理の流れを次の図に示します。

図 11-12　レコードのフィルタリングのデータ処理の流れ



1. レコード条件の判定

   入力されたレコードのレコード名が，レコード条件に指定したレコード名と一致するかどうかを判定します。

   - レコード条件に一致したレコード

     フィールド条件の判定の処理へ進みます。

   - レコード条件に一致しなかったレコード

     一致しなかったレコードは，そのまま出力されて，次のコールバックに渡されます。

2. フィールド条件の判定

   レコード条件に一致したレコードのフィールド値が，フィールド条件に指定したフィールド値と一致するかどうかを判定します。

   - フィールド条件に一致したレコード

     レコード条件，およびフィールド条件が成立したレコードが出力されます。出力されたレコードは，次のコールバックに渡されます。

   - フィールド条件に一致しなかったレコード

     一致しなかったレコードは，そこで破棄されます。

レコードを破棄した場合，破棄したレコード数は，アダプタートレースに出力されます。アダプタートレースについては，「6.3.3　アダプタートレースの詳細」を参照してください。

11.4.3　レコードのフィルタリングの設定

レコードのフィルタリングについて設定が必要なファイルは，アダプター構成定義ファイルです。アダプター構成定義ファイルの入力アダプター定義，または出力アダプター定義で，次の CB 定義に設定します。

- 編集用 CB 定義

フィルター定義に，レコードをフィルタリングするためのレコード条件とフィールド条件を定義します。定義の詳細については，「9.9.3　編集用 CB 定義」，および「9.11.3　フィルター定義」を参照してください。

11.5　レコードの抽出

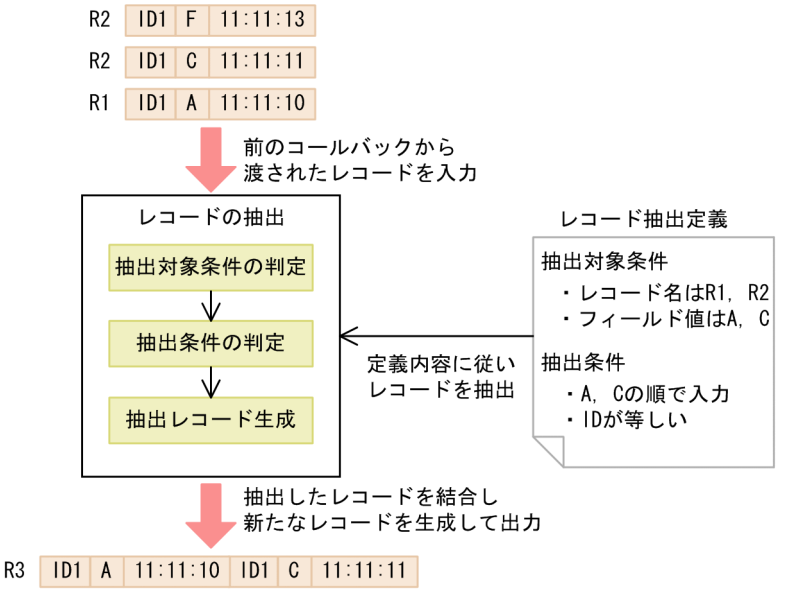
ここでは，レコードの抽出の概要，データ処理の流れ，および必要な設定について説明します。

11.5.1　レコードの抽出の概要

特定のレコードを抽出したあと，それらのレコードを意味のある一つのレコードにする場合，編集用コールバックとして，レコード抽出を使用します。

レコードの抽出の概要を次の図に示します。

図 11-13　レコードの抽出の概要



レコードの抽出では，アダプター構成定義ファイルのレコード抽出定義で定義した内容に従って，入力されたレコードが条件に一致するかどうかを判定します。レコード形式，レコードに設定されているフィールド値，レコードの入力順序などから目的のレコードを抽出したあと，それらのレコードを結合して新たなレコードを生成して出力します。レコードの抽出で，レコードを結合して新たに生成するレコードのことを**抽出レコード**といいます。レコードの抽出の処理の詳細については，「11.5.2　レコードの抽出のデータ処理の流れ」で説明します。

レコード抽出の入出力データの形式

レコード抽出の入出力データの形式は，共通形式レコードです。

レコード抽出を実施するタイミング

レコード抽出は，入力アダプターの編集用コールバックとして実施します。HTTP パケット入力コネクターによるデータ入力またはフォーマット変換のあとにレコード抽出を実施します。

なお，出力アダプターでは，レコード抽出は使用できません。

編集用コールバックを実施するタイミングについては，「2.5.2　データの編集方法の検討」を参照してください。

レコード抽出の使用例

例えば，「11.3　HTTP パケットの入力」で説明した HTTP パケット入力コネクターとレコード抽出を組み合わせた場合には，HTTP パケットの要求から応答までの時間を測定できます。この場合，レコード抽出では，送信元 IP アドレスとポート番号，および送信先 IP アドレスとポート番号が等しいリクエ

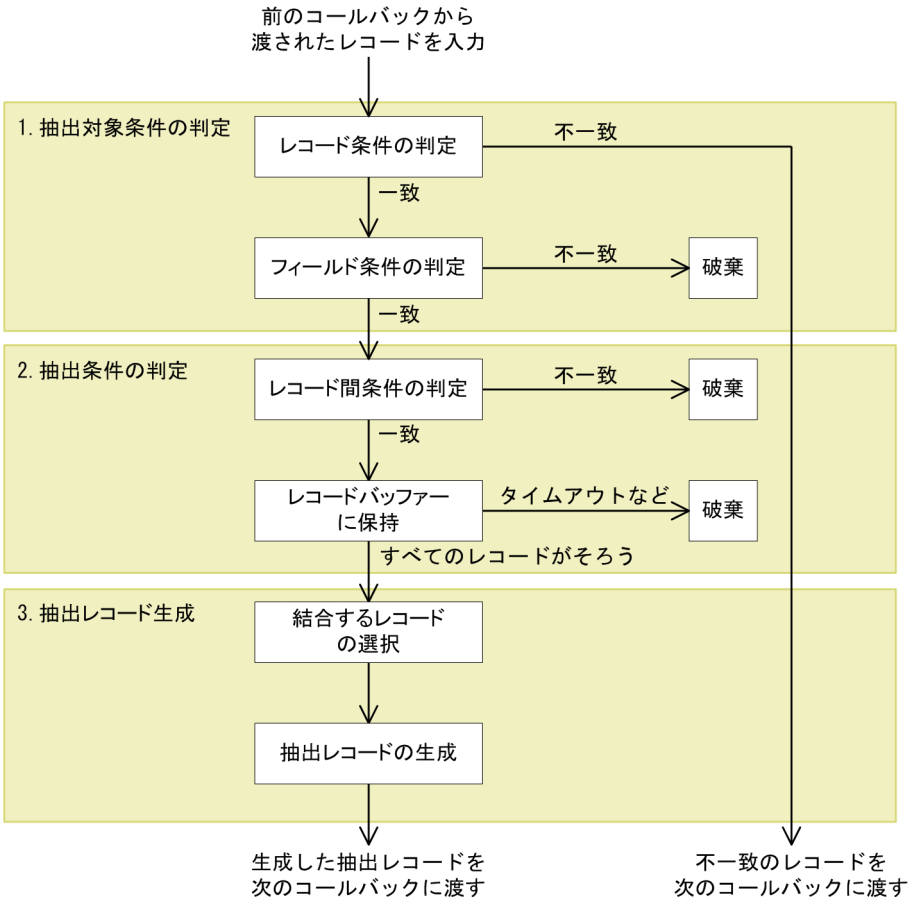
ストとレスポンスのパケットのレコードを抽出し，それらのレコードを対にして結合します。結合されたレコードで，リクエストとレスポンスのパケットの時間を比較することで，要求から応答までの時間を測定できます。

また，結合されたレコードに対して，さらにレコード抽出を行い，指定した URL のレコードだけを抽出したあと，指定した順序で URL が遷移しているレコードを結合します。これによって，HTTP パケットの URL からユーザーのサイト遷移状況を監視することもできます。

11.5.2　レコードの抽出のデータ処理の流れ

レコードの抽出では，レコードを抽出して，抽出したレコードを結合して新たなレコードを生成するまでに，抽出対象条件の判定，抽出条件の判定，および抽出レコード生成の処理を実施します。レコードの抽出のデータ処理の流れを次の図に示します。

図 11-14　レコードの抽出のデータ処理の流れ



図中の 1.～3.の処理の詳細について，以降で説明します。なお，説明に出てくるアダプター構成定義ファイルのレコード抽出定義のタグや属性の詳細については，「9.11.4　レコード抽出定義」を参照してください。

(1)　抽出対象条件の判定

入力されたレコードに対して，抽出対象条件の判定をします。

抽出対象条件とは，抽出対象のレコードを取捨選択するための条件の一つです。抽出対象条件は，レコード抽出定義の targetrecord タグで指定します。

抽出対象条件には，レコード条件とフィールド条件を指定します。

- **レコード条件**
  record タグで，抽出対象のレコード名を指定します。
- **フィールド条件**
  field タグで，抽出対象のレコードのフィールド名，比較演算記号，および条件値を指定します。

例えば，レコード名が R1，フィールド名が F1，フィールド値が TARO という条件を指定したい場合には，レコード条件のレコード名には「source="R1"」，フィールド条件には「source="F1" condition="eq" value="TARO"」と指定します。

抽出対象条件の判定では，レコード抽出定義で定義した抽出対象条件に従って，レコード条件の判定，およびフィールド条件の判定が行われます。抽出対象条件の判定の流れを次に示します。

1. **レコード条件の判定**
   入力されたレコードのレコード名が，レコード条件に指定したレコード名と一致するかどうかを判定します。
   - **レコード条件に一致したレコード**
     フィールド条件の判定の処理へ進みます。
   - **レコード条件に一致しなかったレコード**
     一致しなかったレコードは，そのまま出力されて，次のコールバックに渡されます。
2. **フィールド条件の判定**
   レコード条件に一致したレコードのフィールド値が，フィールド条件に指定したフィールド値と一致するかどうかを判定します。
   - **フィールド条件に一致したレコード**
     抽出対象条件が成立して，抽出条件の判定の処理へ進みます。抽出条件の判定の処理については，「(2)　抽出条件の判定」を参照してください。
   - **フィールド条件に一致しなかったレコード**
     一致しなかったレコードは，そこで破棄されます。

(2)　抽出条件の判定

抽出対象条件が成立したレコードに対して，抽出条件の判定をします。

抽出条件とは，抽出対象のレコードを取捨選択するための条件の一つです。抽出条件は，レコード抽出定義の extraction タグで指定します。

抽出条件には，レコードバッファーに保持するレコードの上限値や，レコード間条件を指定します。**レコード間条件**には，レコードの入力順序と，レコード間でフィールド値を比較するためのフィールド値を指定します。レコード間条件は，targets タグで指定します。

抽出条件の判定では，レコード抽出定義で定義した抽出条件に従って，レコード間条件の判定が行われます。レコード間条件に一致したレコードは，抽出条件が成立するまでレコードバッファーに保持されます。抽出条件の判定の流れを次に示します。

1. **レコード間条件の判定**
   抽出対象条件が成立したレコードに対して，レコード間条件で指定した次の内容と一致するかどうかを判定します。
   - レコードの入力順序が一致するかどうか。※

- レコード間のフィールド値が一致するかどうか。※

注※

レコード内の時刻情報が同一である複数のレコードが入力された場合，抽出条件はレコードの入力順序で判定します。

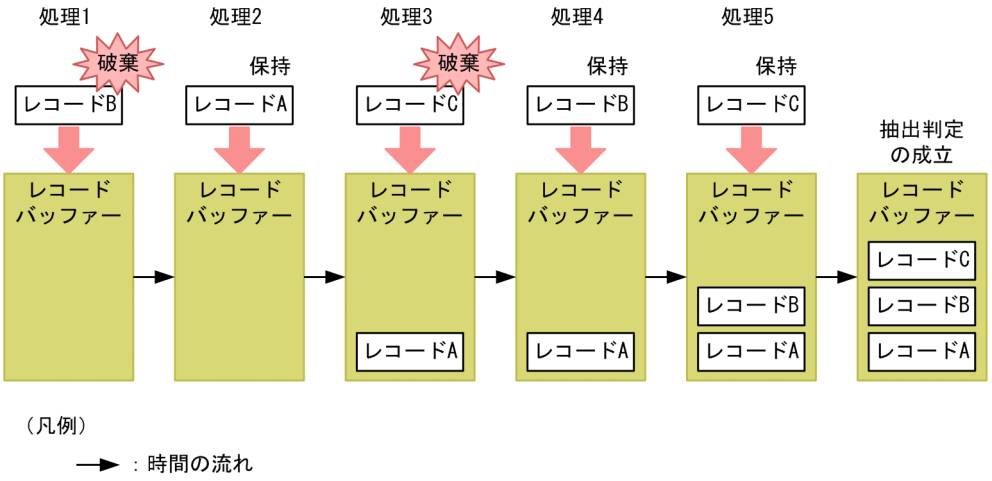
これらの条件に一致したレコードはレコードバッファーに保持され，一致しなかったレコードはそこで破棄されます。

2. レコードバッファーに保持

レコード間条件に一致したレコードは，レコードバッファーに保持され，レコード間条件で指定したすべてのレコードがそろって抽出条件が成立すると，抽出レコード生成の処理へ進みます。抽出レコード生成の処理については，「(3)　抽出レコード生成」を参照してください。

レコードの入力順序を指定している場合のレコード保持と破棄の流れの例を次の図に示します。

図 11-15　レコードの入力順序を指定している場合のレコード保持と破棄の流れの例



この例では，レコードの入力順序をレコード A→レコード B→レコード C と指定しています。このため，入力されたレコードは次のように保持，または破棄されます。

- 処理 1：レコードバッファー内にレコードがない状態で入力されたレコード B は，入力順序に一致しないため破棄されます。
- 処理 2：レコードバッファー内にレコードがない状態で入力されたレコード A は，入力順序に一致するため保持されます。
- 処理 3：レコード A のあとに入力されたレコード C は，入力順序に一致しないため破棄されます。
- 処理 4：レコード A のあとに入力されたレコード B は，入力順序に一致するため保持されます。
- 処理 5：レコード A とレコード B のあとに入力されたレコード C は，入力順序に一致するため保持されます。レコード A→レコード B→レコード C の順序でレコードが入力され，すべてのレコードがそろうため，ここで抽出条件が成立します。

レコードを破棄した場合，破棄したレコード数は，アダプタートレースに出力されます。アダプタートレースについては，「6.3.3　アダプタートレースの詳細」を参照してください。

抽出条件の判定では，レコード内の時刻情報が逆転した場合や，レコードバッファーに保持しているレコードがタイムアウトした場合などにもレコードを破棄します。それぞれの内容について，次に説明します。

- **レコード内の時刻情報が逆転した場合**

  レコード内の時刻情報の逆転が発生したレコードが入力された場合には，レコードを破棄します。

- **レコードバッファーに保持しているレコードがタイムアウトした場合**

  レコードの抽出では，レコードバッファーに保持しているレコードがタイムアウトする条件として，レコードの生存時間を指定できます。レコードの生存時間は，extraction タグの timelimit 属性で指定します。

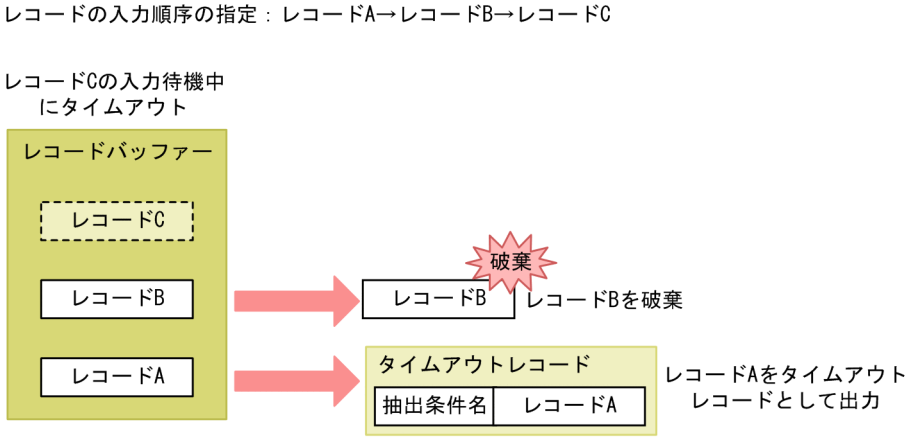
  指定した生存時間が経過しても抽出条件が成立していない場合には，タイムアウトと見なして，レコードバッファーに保持されていたレコードを破棄します。

  このとき，extractions タグの timeout 属性でタイムアウトレコードの出力を指定していると，レコードバッファーに保持されていたレコードのうち，最初のレコードだけを出力できます。

  タイムアウト時に出力されるレコードのことを**タイムアウトレコード**といます。タイムアウトレコードには，extraction タグの name 属性で指定した抽出条件名が入力されたフィールドが追加されます。

  タイムアウト時のレコードの処理の流れを次の図に示します。

  図 11-16　タイムアウト時のレコードの処理の流れ



  この例の条件は次のとおりです。

  - レコード A，レコード B，レコード C の三つがそろうと，抽出条件をすべて満たして，抽出レコード生成の処理へ進むことができる。
  - タイムアウトした場合には，タイムアウトレコードを出力する。
  - レコード C の入力待機中にタイムアウトした。

  このため，レコード B は破棄され，レコード A はタイムアウトレコードとして出力されます。

- **レコードバッファーに保持するレコード数の上限値を超えた場合**

  レコードの抽出では，レコードバッファーに保持するレコード数を管理するために，保持するレコード数の上限値を指定できます。保持するレコード数の上限値は，extractions タグの size 属性で指定します。上限値を超えた場合には，レコードバッファーに保持しているすべてのレコードを破棄します。

- **同一レコードが入力された場合**

  レコードの抽出では，同一の抽出対象条件を満たし，かつレコード間条件に指定したフィールドの値が等しいレコードは，同一レコードと見なします。同一レコードが入力された場合には，extractions タグの samerecord 属性で指定した内容に従って，次の処理が行われます。

  - **samerecord 属性で overwrite を指定した場合**

同一レコードと見なされた既存のレコードが，レコードバッファーのどの位置にあるかによって処理が異なります。

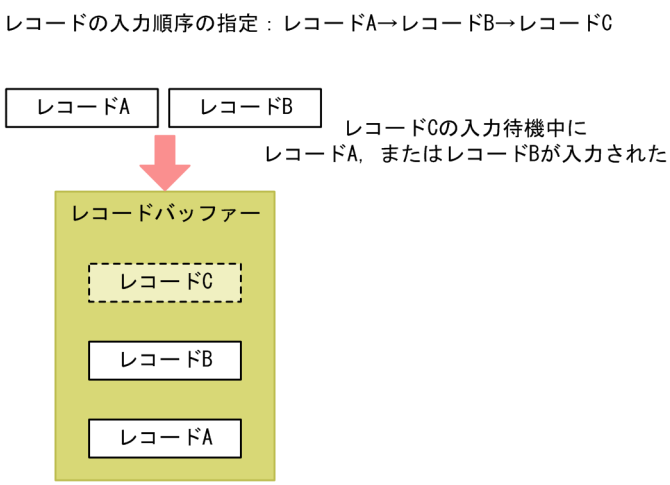
最も後ろのレコードの場合，入力されたレコードで，レコードバッファーに保持していた既存のレコードを上書きします。ほかのレコードの場合，レコードバッファーに保持していた既存のレコードをすべて破棄し，入力されたレコードを保持します。

- samerecord 属性で delete を指定した場合

  入力されたレコードを破棄します。

次の図に示す例を使用して，同一レコードが入力された場合の処理について説明します。

図 11-17　同一レコードが入力された場合の処理



この例の条件は次のとおりです。

- レコード A，レコード B，レコード C の三つがそろうと，抽出条件をすべて満たして，抽出レコード生成の処理へ進むことができる。
- samerecord 属性で overwrite を指定している。
- レコード C の入力待機中に，レコード A，またはレコード B が入力された。

この条件の場合，レコード A とレコード B のどちらが入力されるかによって，次のように処理が異なります。

- レコード A の場合

  レコードバッファーに保持していた既存のレコード A とレコード B を破棄し，入力されたレコード A を保持します。

- レコード B の場合

  入力されたレコード B で，レコードバッファーに保持していた既存のレコード B を上書きします。

(3)　抽出レコード生成

抽出レコード生成では，抽出条件が成立したレコードを結合して，抽出レコードを生成します。抽出レコード生成の流れを次に示します。

1. 結合するレコードの選択

   抽出条件が成立したレコードの中から，抽出レコードとして結合するレコードを選択します。結合するレコードは，select タグの source 属性で指定します。

2. 抽出レコードの生成

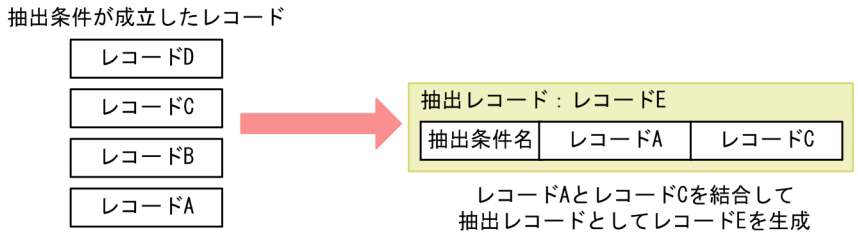
手順 1.で選択したレコードを結合して，抽出レコードを生成します。

生成される抽出レコードには，extraction タグの name 属性で指定した抽出条件名が入力されたフィールドが追加されます。

- 抽出条件名が入力されるフィールド名：「condition」
- ほかのフィールド：「<抽出対象レコード名>_<フィールド名>」

抽出レコードの例を次の図に示します。

図 11-18　抽出レコードの例



この例では，レコード A とレコード C を結合して，抽出レコードとして，新たなレコード E が生成されます。

11.5.3　レコードの抽出の設定

レコードの抽出について設定が必要なファイルは，アダプター構成定義ファイルです。アダプター構成定義ファイルの入力アダプター定義で，次の CB 定義に設定します。

- **編集用 CB 定義**
  レコード抽出定義に，レコードを抽出するための抽出対象条件と抽出条件などを定義します。定義の詳細については，「9.9.3　編集用 CB 定義」，および「9.11.4　レコード抽出定義」を参照してください。

11.6　ファイルへの出力

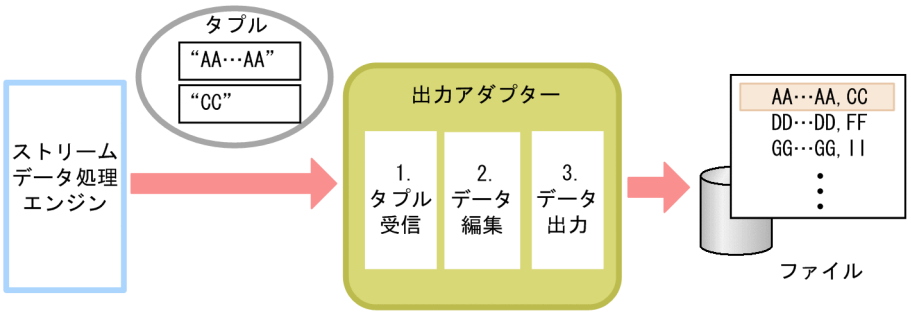
ここでは，ファイルへの出力の概要，データ処理の流れ，および定義ファイルでの設定内容について説明します。

11.6.1　ファイルへの出力の概要

ファイルへの出力では，ストリームデータ処理エンジンで処理したデータを，出力形式に合わせてデータを変換してから出力します。ファイルへの出力の処理は出力アダプターで実施します。

ファイルへの出力の概要を次の図に示します。

図 11-19　ファイルへの出力の概要



1. タプル受信

   ストリームデータ処理エンジンで処理されたストリームデータの集計・分析結果のタプルを受信します。

2. データ編集

   受信したデータのマッピングおよびフォーマット変換をして，データを編集します。

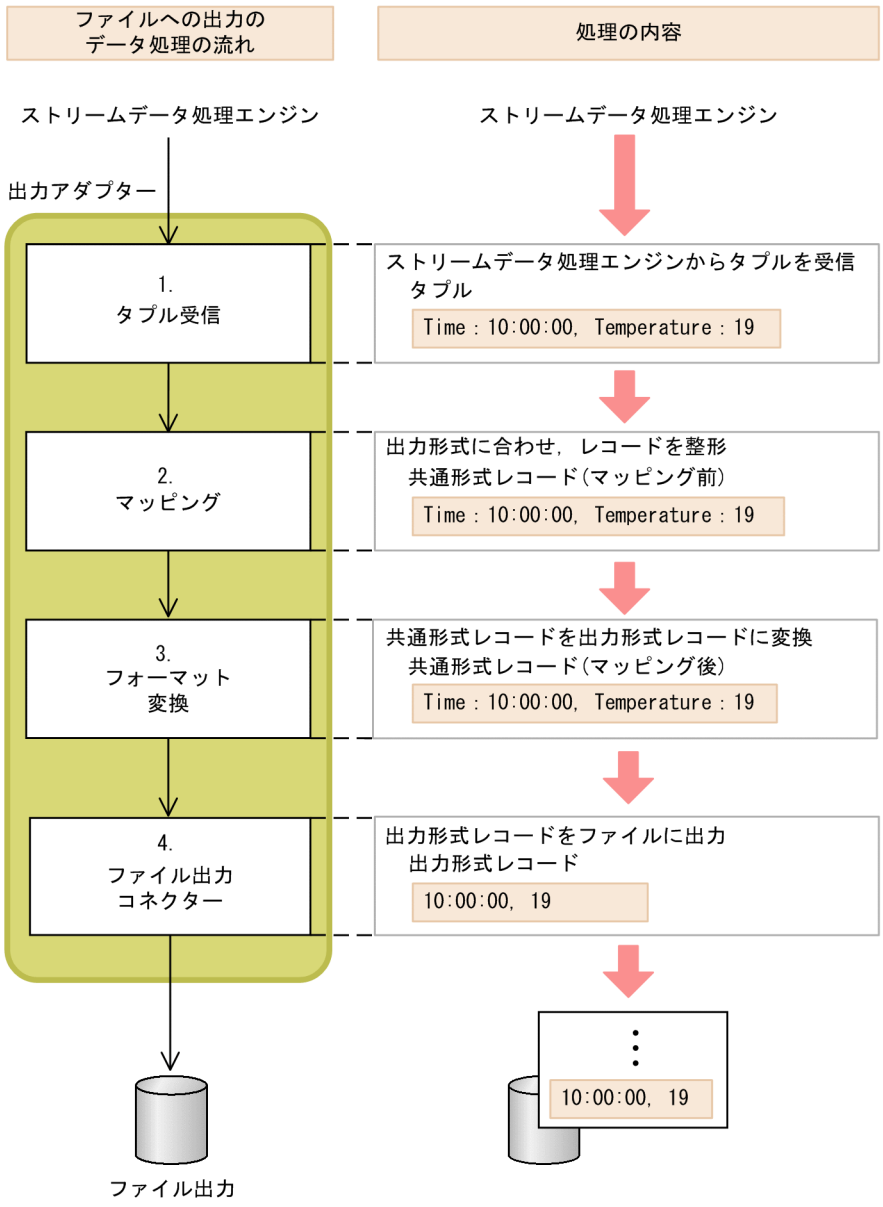
3. データ出力

   ファイル出力コネクターで，編集したデータを出力形式に合わせたデータに変換して出力します。

11.6.2　ファイルへの出力のデータ処理の流れ

出力アダプターでのファイルへの出力のデータ処理の流れと処理の内容を次の図に示します。

図 11-20　ファイルへの出力のデータ処理の流れと処理の内容



出力アダプターで扱うデータ形式について，「(1)　出力アダプターで扱うデータ形式」で説明します。また，各処理の詳細について，「(2)　タプル受信」以降で説明します。

(1)　出力アダプターで扱うデータ形式

出力アダプターで扱うデータ形式には，共通形式レコードおよび出力形式レコードがあります。共通形式レコードについては，入力アダプターで扱う共通形式レコードと同じです。共通形式レコードについては，「11.2.2(1)　入力アダプターで扱うデータ形式」の共通形式レコードの説明を参照してください。ここでは出力形式レコードについて説明します。

出力形式レコード

出力ファイルに出力する，行単位のデータのことです。出力アダプターでは，1 行のデータを 1 出力形式レコードとして扱います。

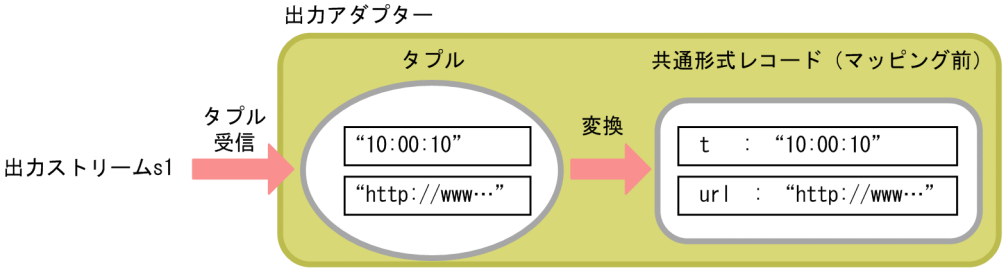
(2)　タプル受信

タプル受信についての定義は，アダプター構成定義ファイルの出力ストリーム定義に定義します。

タプル受信では，出力ストリームから送信されたタプルを受信し，共通形式レコードに変換します。

出力アダプターでの，出力ストリームからのタプル受信の例を次の図に示します。この例では，出力ストリーム s1 からタプルを受信します。

図 11-21　出力アダプターでのタプル受信の例



(3)　マッピング

マッピングについての定義は，アダプター構成定義ファイルのマッピング定義で定義します。

マッピングには，**レコードとストリーム間のマッピング**，および**レコード間のマッピング**があります。それぞれのマッピングの概要を次の表に示します。

表 11-3　マッピングの概要

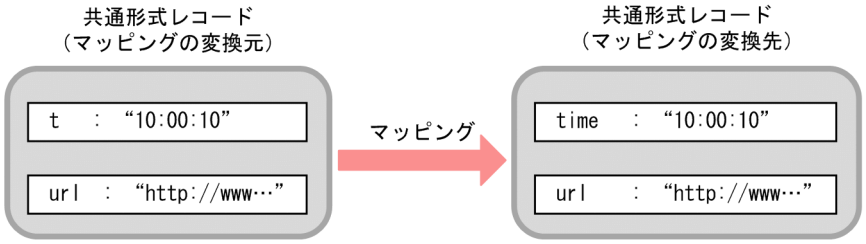
項番	マッピングの種類	説明
1	レコードとストリーム間のマッピング	タプル受信で変換された共通形式レコード（マッピングの変換元）を，出力形式に従った共通形式レコード（マッピングの変換先）に変換します。 レコードとストリーム間のマッピングは，タプル受信のあとに必ず実施します。
2	レコード間のマッピング	レコードとストリーム間のマッピングで変換された共通形式レコードを編集し，マッピングの変換先の共通形式レコードに変換します。 レコード間のマッピングは，レコードとストリーム間のマッピングのあと，かつフォーマット変換の前に，必要に応じて実施します。 マッピングの変換元の共通形式レコードのフィールド名を変更したい場合や，複数のフィールドから次のコールバックの処理に不要なフィールドを削除したい場合などに使用できます。 また，組み込み関数※を使用して，マッピングの変換元の共通形式レコードから文字列や時刻を取得し，取得した文字列や時刻をマッピングの変換先の共通形式レコードに反映することもできます。 なお，レコード間のマッピングは複数定義できます。

注※

レコード間のマッピングで使用できる組み込み関数は，アダプター構成定義ファイルの map タグの function 属性で指定します。function 属性で指定できる組み込み関数については，「9.11.2　マッピング定義」を参照してください。

出力アダプターでのマッピングの例を次の図に示します。この例では，t フィールドと url フィールドを，アダプターの出力形式に従って time フィールドと url フィールドに変換し，マッピングの変換先の共通形式レコードに変換します。

図 11-22　出力アダプターでのレコードとストリーム間マッピングの例



参考

レコードとストリーム間のマッピングのあとには，レコード間のマッピングおよびレコードのフィルタリングが順不同で実施できます。必要に応じて実施してください。

レコードのフィルタリングについては，「11.4　レコードのフィルタリング」を参照してください。

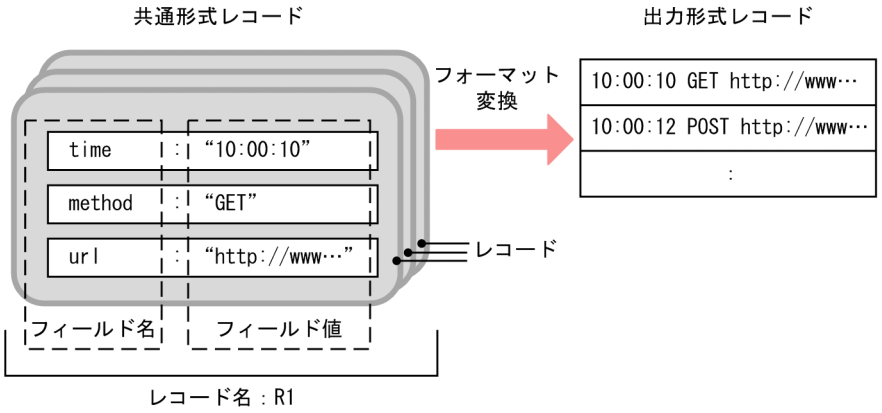
(4)　フォーマット変換

フォーマット変換についての定義は，アダプター構成定義ファイルのフォーマット変換定義で定義します。

フォーマット変換では，共通形式レコードを出力形式レコードに変換します。

共通形式レコードから出力形式レコードへのフォーマット変換の例を次の図に示します。この例では，TIME 型と文字列型のフィールド値を持つ共通形式レコードから，空白で区切られた三つのフィールドで構成された出力形式レコードにフォーマット変換されます。

図 11-23　共通形式レコードから出力形式レコードへのフォーマット変換の例



この図の場合の共通形式レコードのレコード構成と指定するフォーマット変換定義のタグを次の表に示します。

表 11-4　指定する定義ファイルとレコード構成

レコード構成	タグ
レコード名：R1 レコード構成：($_time) △ ($_method) △ ($_url)	record タグ（レコード定義）
フィールド名：time，型：TIME	field タグ（フィールド定義）
フィールド名：method，型：STRING	
フィールド名：url，型：STRING	

（凡例）

△：半角スペース

変換できるデータ型および共通形式レコードのレコード構成の設定については，「9.11.1　フォーマット変換定義」を参照してください。

(5)　ファイル出力コネクターによるファイルへの出力

ファイル出力についての定義は，アダプター構成定義ファイルのファイル出力コネクター定義で定義します。

ファイル出力コネクターでは，入力された出力形式レコードを，定義された出力ディレクトリに出力します。

ファイル出力コネクターで出力できるファイルのファイル構成を次の表に示します。

ファイル構成	説明
ラップアラウンド	ファイル出力コネクター定義に指定した出力ファイル生成規則に従ってファイルを生成します。 最大ファイル数に達すると，最初に起動したファイルを上書きして書き込みます。
非ラップアラウンド	ファイル出力コネクター定義に指定した出力ファイル生成規則に従ってファイルを生成します。 最大ファイル数に達すると，出力アダプターでレコードの出力を停止し，レコードを破棄します。

11.6.3　ファイルへの出力の設定

ファイルへの出力について設定が必要な定義ファイルは，アダプター構成定義ファイルです。アダプター構成定義ファイルの出力アダプター定義で，次の CB 定義に設定します。

- **出力用 CB 定義**

  ファイル出力コネクター定義に，出力コネクターの情報を定義します。定義の詳細については，「9.9.2　出力用 CB 定義」，および「9.10.3　ファイル出力コネクター定義」を参照してください。

- **編集用 CB 定義**

  フォーマット変換定義に，出力データのデータ形式を定義します。マッピング定義には，マッピング情報を定義します。定義の詳細については，「9.9.3　編集用 CB 定義」，「9.11.1　フォーマット変換定義」，または「9.11.2　マッピング定義」を参照してください。

- **受信用 CB 定義**

  出力ストリーム定義に，出力アダプターが接続する出力ストリームの情報を定義します。定義の詳細については，「9.9.5　受信用 CB 定義」，および「9.12.2　出力ストリーム定義」を参照してください。

11.7　ダッシュボードへの出力

ここでは，ダッシュボードへの出力の概要，データ処理の流れ，および定義ファイルでの設定内容について説明します。

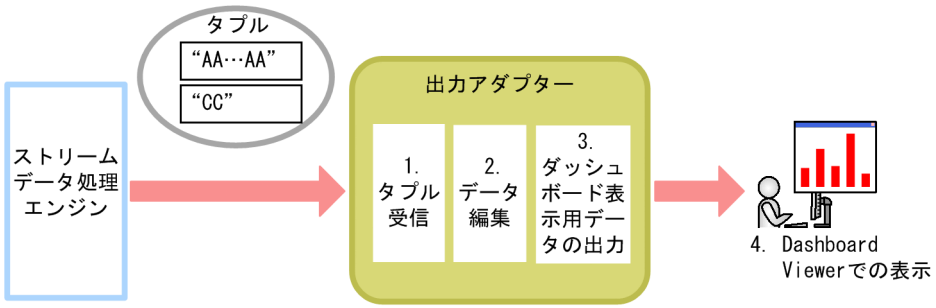
11.7.1　ダッシュボードへの出力の概要

ダッシュボードへの出力では，ストリームデータの集計・分析結果を棒グラフや折れ線グラフの形式でダッシュボードに表示するために，ストリームデータ処理エンジンで処理したデータを，**ダッシュボード表示用データ**に変換して Flex Dashboard の Dashboard Server に送信します。また，Dashboard Server で取得したダッシュボード表示用データを Dashboard Viewer で表示します。

ダッシュボードへの出力の概要を次の図に示します。

なお，Dashboard Viewer でストリームデータの集計・分析結果を表示するには，Internet Explorer および Flash Player が必要です。

図 11-24　ダッシュボードへの出力の概要



1. タプル受信

   ストリームデータ処理エンジンで処理されたストリームデータの集計・分析結果のタプルを受信します。

2. データ編集

   受信したデータのマッピングをします。

3. ダッシュボード表示用データの出力

   ダッシュボード出力コネクターで，編集したデータをダッシュボード表示用データに変換して出力します。また，古くなったデータを削除します。

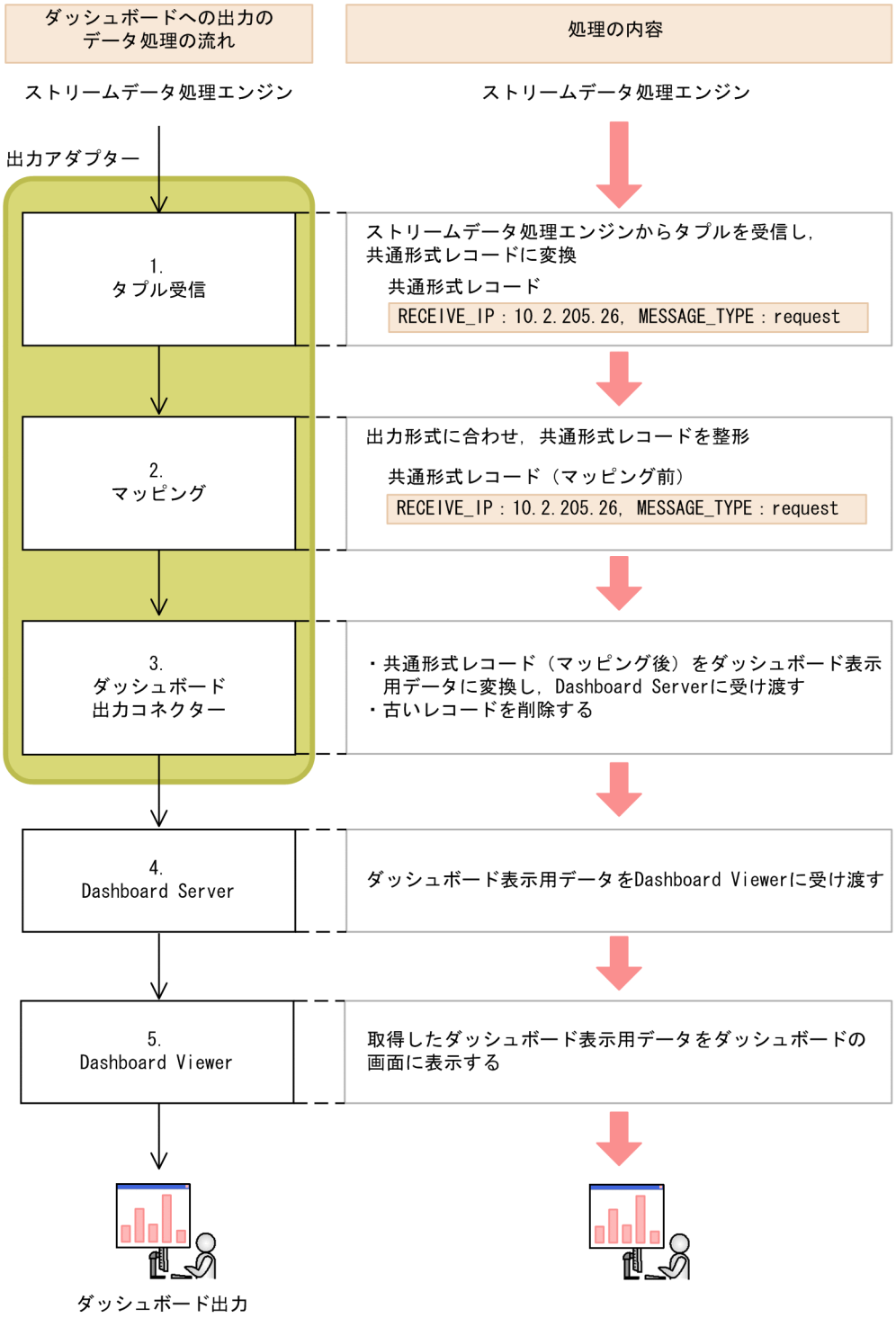
4. Dashboard Viewer での表示

   Dashboard Viewer で，ダッシュボード出力コネクターから出力されたダッシュボード表示用データを表示します。

11.7.2　ダッシュボードへの出力のデータ処理の流れ

ダッシュボードへの出力のデータ処理の流れと処理の内容を次の図に示します。

図 11-25　ダッシュボードへの出力のデータ処理の流れと処理の内容



ここでは，図中のダッシュボード出力コネクターでのダッシュボード表示用データの出力から Dashboard Viewer でのダッシュボード表示用データの表示までのデータ処理の流れについて説明します。タプル受信，およびマッピングについては，「11.6.2(2)　タプル受信」，および「11.6.2(3)　マッピング」を参照してください。また，出力アダプターで扱うデータ形式については，「11.6.2(1)　出力アダプターで扱うデータ形式」を参照してください。

(1)　ダッシュボード出力コネクターでのダッシュボード表示用データへの変換

ダッシュボード出力コネクターでのダッシュボード表示用データへの変換についての定義は，アダプター構成定義ファイルのダッシュボード出力コネクター定義で定義します。

ダッシュボード出力コネクターでは，マッピング後の共通形式レコードを取得し，ダッシュボード出力コネクター定義に従ってダッシュボード表示用データに変換して，**RMI 通信用領域**に出力します。RMI 通信用領域に出力したダッシュボード表示用データは，Dashboard Server から取得できます。

(2)　ダッシュボード出力コネクターでのレコードの削除

ダッシュボード出力コネクターでのレコードの削除についての定義は，アダプター構成定義ファイルのダッシュボード出力コネクター定義に定義します。

ダッシュボード出力コネクターで取得したダッシュボード表示用データがダッシュボード出力コネクターのメモリを圧迫しないように，ダッシュボード出力コネクター定義に定義したレコードの削除条件に従って，ダッシュボード出力コネクターのレコード保持領域からダッシュボード表示用データを削除します。

なお，ダッシュボード出力コネクターでダッシュボード表示用データを削除した場合，削除した日時，ダッシュボード表示用データのレコードの個数がアダプタートレースに出力されます。

レコードの削除には，次の方法があります。

レコード保持期間による削除

レコードの保持期間による削除では，ダッシュボード出力コネクターに新しいレコードが追加されたとき，レコード保持領域内のレコードのうち，基準時刻から保持期間を引いた時刻よりもレコードに設定された時刻情報が古いレコードが削除されます。例えば，基準時刻が 10:00:10，保持期間が 5 秒，レコードに設定された時刻が 10:00:04 だった場合，そのレコードは削除されます。

基準時刻から保持期間を引いた時刻とレコードに設定された時刻情報が同じ場合，レコードは削除されません。

なお，レコードの保持期間による削除は，レコード保持領域の各レコードについて，レコードに設定された時刻情報がレコードの到着順に昇順になっているものとして動作します。昇順になっていない場合，基準時刻から保持期間を引いた時刻よりもレコードに設定された時刻情報が新しいレコードのあとに到着した，基準時刻から保持期間を引いた時刻よりも古いレコードは削除されません。

なお，時刻の比較をする際の時刻の精度はミリ秒です。ミリ秒より小さい精度は切り捨てられます。

レコードの保持期間による削除は，ダッシュボード出力コネクター定義の RecordHoldTime タグの，DateReference 属性，RecordTime 属性，および DateFieldPosition 属性に指定します。指定する属性の詳細については，「9.10.4　ダッシュボード出力コネクター定義」を参照してください。

最大レコード保持数による削除

最大レコード保持数による削除では，ダッシュボード出力コネクターに新しいレコードが追加された時，レコード保持領域内のレコード数が最大レコード保持数よりも大きい場合に，レコード数が最大レコード保持数と等しくなるまでレコードを削除します。このときレコードは，レコード保持領域内に追加された順番で古いレコードから削除されます。

最大レコード保持数による削除は，ダッシュボード出力コネクター定義の DashboardOutputConnectorDefinition タグの MaxNum 属性に指定します。属性の詳細については，「9.10.4　ダッシュボード出力コネクター定義」を参照してください。

取得済みレコードの削除

取得済みレコードの削除では，Dashboard Viewer が一つの Dashboard Server からダッシュボード表示用データを取得する場合に，取得済みのレコードを削除します。

取得済みレコードの削除は，ダッシュボード出力コネクター定義の DashboardOutputConnectorDefinition タグの ReadRecordRemoveFlag 属性に指定します。属性の詳細については，「9.10.4　ダッシュボード出力コネクター定義」を参照してください。

(3)　Dashboard Server でのダッシュボード表示用データの取得

Dashboard Server でのダッシュボード表示用データの取得についての定義は，アダプターグループ定義に定義します。

Dashboard Server では，ダッシュボード出力コネクターの RMI 通信用領域にアクセスして，出力されたダッシュボード表示用データを取得します。

なお，Dashboard Server が RMI 通信用領域にアクセスするためには，RMI サーバのポート番号を指定します。RMI サーバのポート番号は，アダプターグループ定義のインプロセスグループ定義，または RMI グループ定義の dashboardPortNo 属性に指定します。指定する属性の詳細については，「9.7.1　インプロセスグループ定義」，または「9.7.2　RMI グループ定義」を参照してください。

(4)　Dashboard Viewer でのダッシュボード表示用データの表示

Dashboard Viewer は，Dashboard Server からダッシュボード表示用データを取得し，ダッシュボード画面に表示します。

クライアントは，Web ブラウザから URL にアクセスして Dashboard Viewer をダウンロードすることで，ダッシュボード表示用データをストリームデータの集計・分析結果として閲覧します。

ストリームデータの集計・分析結果をダッシュボードで表示する手順については，「4.5　ダッシュボードでの分析結果の表示」を参照してください。

11.7.3　ダッシュボードへの出力の設定

ダッシュボードへの出力で設定が必要な定義ファイルは，アダプター構成定義ファイルです。アダプター構成定義ファイルのアダプターグループ定義と，出力アダプター定義の編集用 CB 定義および出力用 CB 定義に設定します。

- **アダプターグループ定義**

  インプロセスグループ定義，または RMI グループ定義に RMI サーバのポート番号を指定します。定義の詳細については，「9.7.1　インプロセスグループ定義」，または「9.7.2　RMI グループ定義」を参照してください。

- **編集用 CB 定義**

  マッピング定義に，マッピング情報を定義します。定義の詳細については，「9.9.3　編集用 CB 定義」，および「9.11.2　マッピング定義」を参照してください。

- **出力用 CB 定義**

  ダッシュボード出力コネクター定義に，ダッシュボード出力コネクターの情報を設定します。定義の詳細については，「9.9.2　出力用 CB 定義」，および「9.10.4　ダッシュボード出力コネクター定義」を参照してください。

11.8　タプルのタイムスタンプの調整

ここでは，タイムスタンプ調整の適用範囲，調整する時刻の範囲，時刻の調整方法や設定などについて説明します。

11.8.1　タイムスタンプ調整の適用範囲

データソースモードの場合，入力ストリームには，タプルに設定されている時刻を基に昇順でタプルが入力される必要があります。しかし，複数の入力アダプターからタプルを送信する場合などに，タプルに設定された時刻の順序が逆転して SDP サーバに到着してしまうことがあります。この場合，時刻が逆転したタプルは SDP サーバに破棄されます。

このような場合に，**タイムスタンプ調整機能**を使用することで，タプルに設定された時刻の順序が逆転して SDP サーバに到着したタプルも，昇順に並べ替えて入力ストリームに入力できます。

タイムスタンプ調整機能は，SDP サーバの時刻制御方式にデータソースモードを使用している場合に使用できます。また，タイムスタンプ調整機能は入力ストリーム単位で動作します。

11.8.2　調整する時刻の範囲

タイムスタンプ調整機能では，**調整する時刻の範囲**内のタイムスタンプを持つタプルを対象にタプルの時刻調整をします。調整する時刻の範囲は，**基準時刻**と**調整する時刻の幅**から決定します。基準時刻からさかのぼって，調整する時刻の幅の範囲が，タイムスタンプ調整機能で調整する範囲となります。基準時刻が遷移すると調整する時刻の範囲も遷移します。

基準時刻

タプルの時刻を調整するときに基準となる時刻です。クエリグループの起動後に，入力アダプターが入力ストリームに対して送信したタプルの中で最も新しい時刻情報を持つタプルの時刻が，この入力ストリームの基準時間として設定されます。また，新たに最新の時刻情報を持つタプルが到着すると，基準時刻はその時刻へ遷移します。

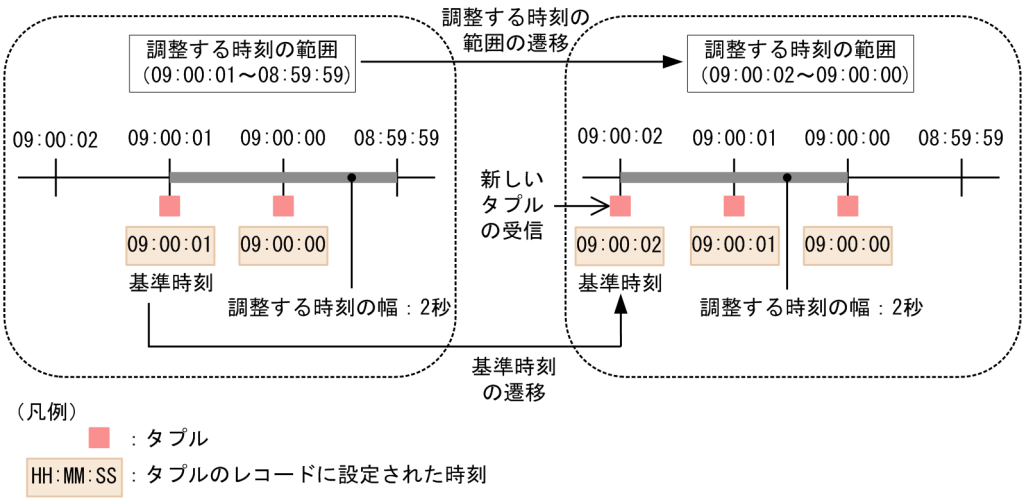
調整する時刻の幅

時刻調整する時間の範囲です。**時刻単位**と**時刻調整範囲**で決まります。時刻単位は，「秒」，「ミリ秒」，「マイクロ秒」から選択できます。時刻調整範囲は，選択した時刻単位での時間の範囲です。

時刻単位と時刻調整範囲は，システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，またはストリーム用プロパティファイルの stream.timestampAccuracy パラメーターで指定します。stream.timestampAccuracy パラメーターについては，「8.6　システムコンフィグプロパティファイル（system_config.properties）」を参照してください。

調整する時刻の幅として時刻単位を「秒」，時刻調整範囲を「2」とした場合の，調整する時刻の範囲の遷移の例を次の図に示します。

図 11-26　調整する時刻の範囲の遷移例



この例では，SDP サーバに到着しているタプルの中で最新の時刻情報を持つタプルの時刻である 09:00:01 が基準時刻となります。また，調整する時刻の幅が 2 秒のため，調整する時刻の範囲は，09:00:01～08:59:59 となります。

そのあと，入力アダプターから送信された新しいタプルが到着し，設定されている時刻情報が 09:00:02 と最新の時刻のため，基準時刻が遷移します。また，それに伴って，調整する時刻の範囲も 09:00:02～09:00:00 に遷移します。

11.8.3　時刻の調整方法

タイムスタンプ調整機能では次の流れでタプルの時刻を調整します。

1. タイムスタンプの設定

   SDP サーバに到着したタプルに対し，タプルのレコードに設定された時刻情報から，指定された時刻単位よりも小さい単位を切り捨てた値をタプルのタイムスタンプとして設定します。例えば，時刻単位に「秒」を指定した場合，ミリ秒単位以下が切り捨てられた値が，タイムスタンプとして設定されます。

2. タプルの保留

   調整する時刻の範囲内のタイムスタンプを持つタプルをタイムスタンプ調整機能内で保留します。

   調整する時刻の範囲よりも過去のタイムスタンプを持つタプルの場合，破棄します。

3. 入力ストリームへの入力

   保留したタプルのうち，調整する時刻の範囲の遷移によって範囲外となったタプルを入力ストリームに入力します。

タプルの保留および入力ストリームへの入力では，基準時刻やタプルのレコードに設定された時刻，指定する時刻単位などによってタプルの時刻の調整方法，および入力ストリームへの入力順序が異なります。ここでは，タプルの時刻の調整方法，および入力ストリームへの入力順序について説明します。

(1)　タプルの時刻の調整方法

ここでは，次の場合に分けて，タプルの時刻の調整方法について説明します。

- クエリグループ開始後の最初のタプルの場合
- タプルのレコードに設定された時刻が基準時刻よりも未来の場合

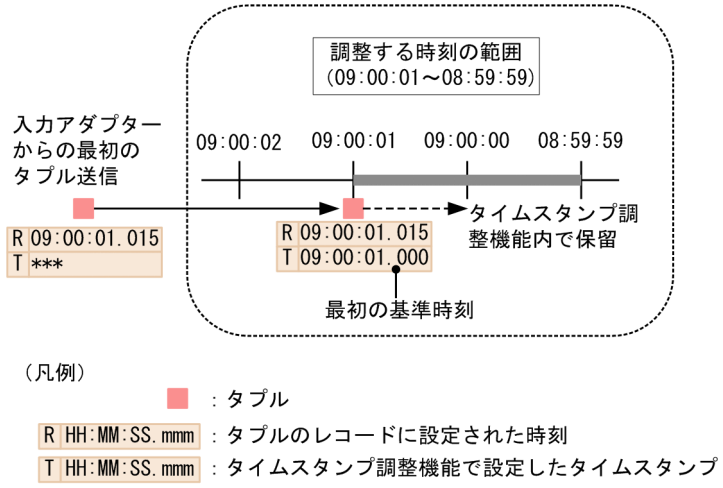
- タプルのレコードに設定された時刻が調整する時刻の範囲内の場合
- タプルのレコードに設定された時刻が調整する時刻範囲より過去の場合

クエリグループ開始後の最初のタプルの場合

入力アダプターから送信されたタプルのレコードに設定された時刻を基準時刻として設定します。

クエリグループ開始後の最初のタプルの例を次の図に示します。なお，この例では，時刻単位を「秒」，調整する時刻の幅に「2」を指定した場合の処理について説明します。

図 11-27　クエリグループ開始後の最初のタプルの例



入力アダプターから最初のタプル（レコードに設定された時刻は「09:00:01:015」）が送信されると，タイムスタンプ調整機能では基準時刻を「09:00:01」に設定します。

この場合の時刻の調整範囲は，「09:00:01～08:59:59」となります。また，最初のタプルは「09:00:01」のタイムスタンプを持つタプルとしてタイムスタンプ調整機能内で保留されます。

タプルのレコードに設定された時刻が基準時刻よりも未来の場合

入力アダプターから送信されたタプルのレコードに設定された時刻が基準時刻よりも未来の時刻の場合，次の順序でタプルの時刻を調整します。

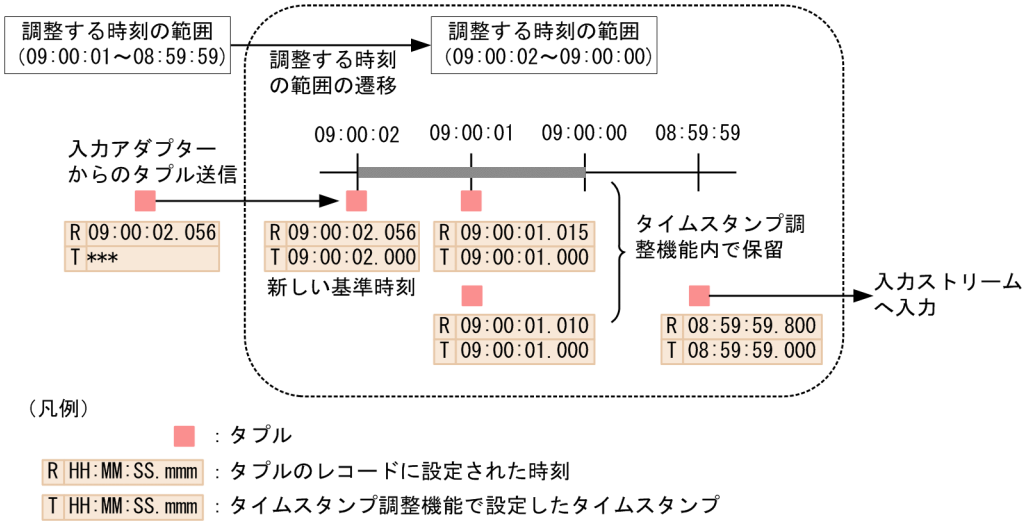
1. 送信されたタプルのレコードに設定された時刻情報から，新しい基準時刻を設定します。これに伴って，調整する時刻範囲が遷移します。

   また，送信されたタプルはタイムスタンプ調整機能内で保留されます。

2. 以前の基準時刻で調整する時刻範囲内だった保留タプルがある場合は，基準時刻の遷移で調整する時刻範囲外となったタプルを，タイムスタンプが古い順に入力ストリームへ入力します。なお，同一時刻のタプルが複数ある場合は，SDP サーバに到着した順序で入力ストリームへ入力します。

タプルのレコードに設定された時刻が基準時刻よりも未来の例を次の図に示します。なお，この例では，時刻単位を「秒」，調整する時刻の幅に「2」を指定した場合の処理について説明します。

図 11-28　タプルのレコードに設定された時刻が基準時刻よりも未来の例



基準時刻が「09:00:01」，調整する時刻範囲が「09:00:01～08:59:59」の場合に，入力アダプターから新しいタプル（レコードに設定された時刻は 09:00:02:056）が送信されると，新しいタプルの時刻が基準時刻よりも未来の時刻のため，基準時刻は「09:00:02」に遷移します。これに伴って，時刻の調整範囲は「09:00:02～09:00:00」に遷移します。

また，新しいタプルは「09:00:02」のタイムスタンプを持つタプルとしてタイムスタンプ調整機能内で保留されます。

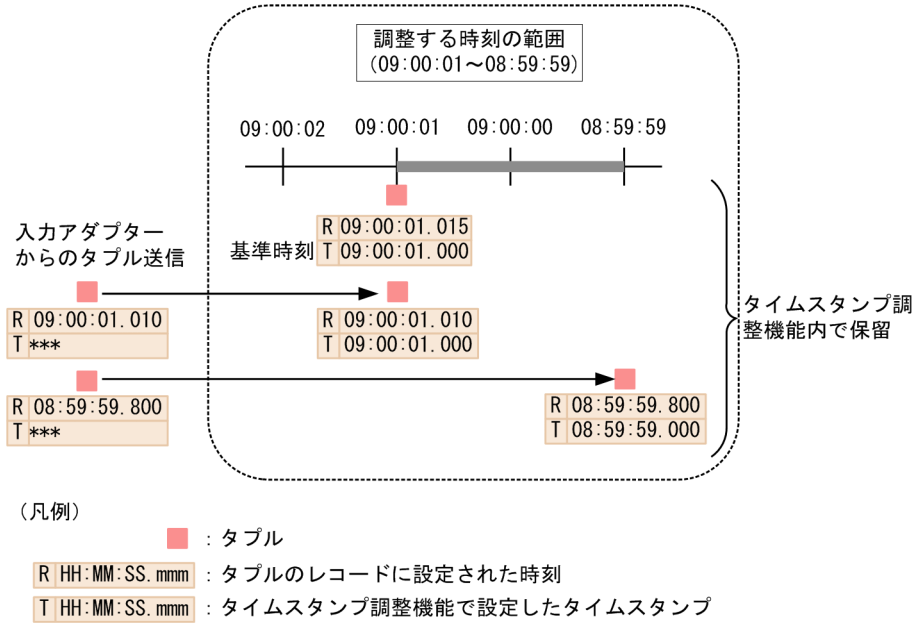
時刻の調整範囲の遷移によって調整範囲から外れる「08:59:59」のタイムスタンプを持つタプルは入力ストリームに入力されます。

タプルのレコードに設定された時刻が調整する時刻の範囲内の場合

入力アダプターから送信されたタプルのレコードに設定された時刻が調整する時刻の範囲内の場合，送信されたタプルをタイムスタンプ調整機能内で保留します。

タプルのレコードに設定された時刻が調整する時刻の範囲内の例を次の図に示します。なお，この例では，時刻単位を「秒」，調整する時刻の幅に「2」を指定した場合の処理について説明します。

図 11-29　タプルのレコードに設定された時刻が調整する時刻の範囲内の例



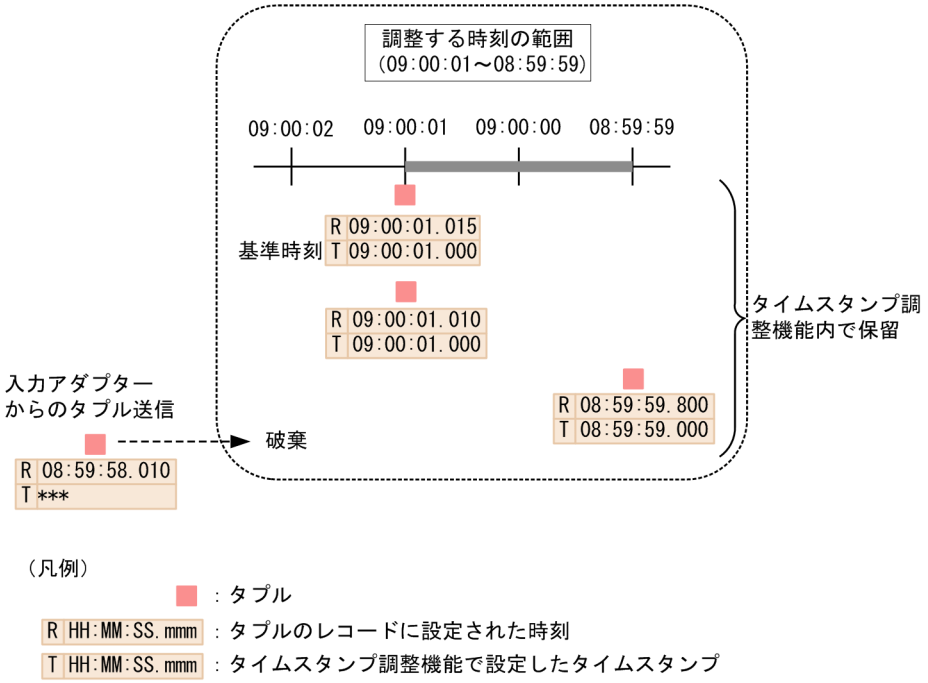
基準時刻が「09:00:01」，調整する時刻範囲が「09:00:01～08:59:59」の場合に，入力アダプターから新しいタプル（レコードに設定された時刻は 09:00:01:010）が送信されると，タプルのレコードに設定された時刻が調整する時刻の範囲内のため，タイムスタンプ調整機能内で「09:00:01」のタイムスタンプを持つタプルとして保留されます。さらに，入力アダプターからタプルのレコードに設定された時刻が「08:59:59.800」のタプルが送信された場合も同様に，タイムスタンプ調整機能内で「08:59:59」のタイムスタンプを持つタプルとして保留されます。

タプルのレコードに設定された時刻が調整する時刻範囲より過去の場合

入力アダプターから送信されたタプルのレコードに設定された時刻が，調整する時刻範囲よりも過去の場合，タプルを破棄して，タプルを破棄したことを示すログをタプルログに出力します。

タプルのレコードに設定された時刻が調整する時刻範囲より過去の例を次の図に示します。なお，この例では，時刻単位を「秒」，調整する時刻の幅に「2」を指定した場合の処理について説明します。

図 11-30　タプルのレコードに設定された時刻が調整する時刻範囲より過去の例



基準時刻が「09:00:01」，調整する時刻範囲が「09:00:01～08:59:59」の場合に，入力アダプターから新しいタプル（レコードに設定された時刻は 08:59:58:010）が送信されると，タプルのレコードに設定された時刻が調整する時刻の範囲よりも過去のため，タプルは破棄されます。タプルログには，タプルが破棄されたことを示すログが出力されます。

(2)　入力ストリームへの入力順序

タイムスタンプ調整機能によって時刻を調整されたタプルは，タイムスタンプ調整機能で設定されたタイムスタンプの昇順で入力ストリームに入力されます。また，同じタイムスタンプを持つタプルが複数あった場合は，SDP サーバに到着した順序で入力ストリームに入力されます。

このとき，タイムスタンプ調整機能で調整する時刻単位と範囲が異なると，同じストリームデータの場合でも，入力ストリームへのタプルの入力順序が異なります。

ここでは，調整する時刻単位と範囲の組み合わせの例ごとに，タイムスタンプ調整機能で設定されるタプルのタイムスタンプと，入力ストリームへのタプルの入力順序について説明します。

ここで説明する例では，次の表に示す A から E のタプルが，入力アダプターから送信されたとします。

タプル	入力アダプターからの送信順	タプルのレコードに設定された時刻
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

時刻単位に「秒」，調整する時刻の範囲に「1」を指定した場合

時刻単位に「秒」，調整する時刻の範囲に「1」を指定した場合の，タイムスタンプ調整機能で設定するタプルのタイムスタンプ，および入力ストリームへのタプルの入力順序を次の表に示します。

なお，時刻単位に「秒」を指定した場合，「ミリ秒」以下が切り捨てられた値がタイムスタンプとして設定されます。

タプル	入力順	タイムスタンプ調整機能で設定したタイムスタンプ
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

この例の場合，すべてのタプルが時刻調整範囲内となるため，すべてのタプルが入力ストリームに入力されます。

時刻単位に「ミリ秒」，調整する時刻の範囲に「999」を指定した場合の例

時刻単位に「ミリ秒」，調整する時刻の範囲に「999」を指定した場合の，タイムスタンプ調整機能で設定するタプルのタイムスタンプ，および入力ストリームへのタプルの入力順序を次の表に示します。

時刻単位に「ミリ秒」を指定した場合「マイクロ秒」以下が切り捨てられた値がタイムスタンプとして設定されます。

タプル	入力順	タイムスタンプ調整機能で設定したタイムスタンプ
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

この例の場合，すべてのタプルが時刻調整範囲内となるため，すべてのタプルが入力ストリームに入力されます。

時刻単位に「マイクロ秒」，調整する時刻の範囲に「999」を指した場合の例

時刻単位に「マイクロ秒」，調整する時刻の範囲に「999」を指定した場合の，タイムスタンプ調整機能で設定するタプルのタイムスタンプ，および入力ストリームへのタプルの入力順序を次の表に示します。

時刻単位に「マイクロ秒」を指定した場合「ナノ秒」以下が切り捨てられた値がタイムスタンプとして設定されます。

タプル	入力順	タイムスタンプ調整機能で設定したタイムスタンプ
A	1	2009/03/01 12:15:22:345678000
B	破棄	設定されません。
C	2	2009/03/01 12:15:23:123456000
D	破棄	設定されません。

タプル	入力順	タイムスタンプ調整機能で設定したタイムスタンプ
E	3	2009/03/01 12:15:24:123456000

この例の場合，タプル A のタイムスタンプが基準時刻となったあと，タプル B はこの基準時刻よりも過去の時刻情報を持ち，調整する時刻の範囲外となるため破棄されます。同様に，タプル D はタプル C よりも過去の時刻情報を持ち，調整する時刻の範囲外となるため破棄されます。

時刻単位に「秒」，調整する時刻の範囲に「0」を指定した場合

時刻単位に「秒」，調整する時刻の範囲に「0」を指定した場合の，タイムスタンプ調整機能で設定するタプルのタイムスタンプ，および入力ストリームへのタプルの入力順序を次の表に示します。

なお，時刻単位に「秒」を指定した場合，「ミリ秒」以下が切り捨てられた値がタイムスタンプとして設定されます。

タプル	入力順	タイムスタンプ調整機能で設定したタイムスタンプ
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	破棄	設定されません。
E	4	2009/03/01 12:15:23:000000000

この例の場合，基準時刻だけが時刻調整範囲となります。タプル C が送信されるとタプル C のタイムスタンプが基準時刻となり，その基準時刻より過去の時刻情報を持つタプル D は破棄されます。

11.8.4　タプル入力完了後の処理

タイムスタンプ調整機能では，入力アダプターから送信されたタプルのレコードに設定された時刻情報が進むことで基準時刻を遷移し，タイムスタンプ調整機能内で保留していたタプルを入力ストリームに入力します。そのため，入力アダプターからのタプルの入力が完了すると，タイムスタンプ調整機能の基準時刻も更新されないため，保留しているタプルは入力ストリームに入力されません。

タプルの入力完了後に，タイムスタンプ調整機能で保留しているタプルを入力ストリームに入力するには，StreamInput オブジェクトの putEnd メソッドを使用します。カスタムアダプターの場合，データ送信 AP で putEnd メソッドを発行して，保留していたタプルを入力ストリームに入力します。標準提供アダプターの場合は，タプル入力完了後に putEnd メソッドが発行されます。

putEnd メソッドを発行した場合のタイムスタンプ調整機能の状態は次のように遷移します。

1. 新たなタプルの受け付けを停止

   タイムスタンプ調整機能で，新たなタプルの受け付けができなくなります。

   クエリグループ内のすべての入力ストリームに putEnd メソッドが発行された時点で，受け付け再開の準備をします。

2. 新たなタプルの受け付けを開始

   タプルの受け付け再開の準備がすべて完了すると，タイムスタンプ調整機能で新たなタプルの受け付けを再開します。受け付け再開時のタイムスタンプ調整機能は，クエリグループ開始直後と同じ状態となります。

また，StreamInput オブジェクトの isStarted メソッドを使用することで，タイムスタンプ調整機能の状態を確認することができます。タイムスタンプ調整機能が 1.の状態のときに isStarted メソッドを発行する

と，false（新たなタプルの受け付けを開始していない状態）が返されます。タイムスタンプ調整機能が 2. の状態のときに isStarted メソッドを発行すると，true（新たなタプルの受け付けを開始している状態）が返されます。

11.8.5　クエリグループの停止閉塞時の動作

タイムスタンプ調整機能では，クエリグループの正常停止時，強制停止時，または閉塞時でタプルに対する動作が異なります。

クエリグループの状態ごとのタイムスタンプ調整機能のタプルに対する動作を次の表に示します。

表 11-5　クエリグループの停止閉塞時のタイムスタンプ調整機能の動作

項番	クエリグループの状態	タイムスタンプ調整機能の動作		
		新たなタプルの受け付けの停止	全保留タプルの入力ストリームへの入力	全保留タプルの破棄
1	正常停止	○	○	×
2	強制停止	○	×	○
3	閉塞	○	×	○

（凡例）

○：動作します。

×：動作しません。

11.8.6　タプルの保留期限

タイムスタンプ調整機能では，タプルの時刻調整のためにタイムスタンプ調整機能内でタプルを保留しています。しかし，大量のタプルを保留しているとメモリが枯渇するおそれがあるため，タイムスタンプ調整機能でのタプルの保留数の上限を設定してシステム全体のリソース使用量の安定化を図ります。

タイムスタンプ調整機能では，入力アダプターから新たなタプルが送信されるときに，時刻調整をしてタプルの保留数の上限値をチェックします。タプルの保留数が上限に達している場合は，新たなタプルの保留はしないで，クエリグループを閉塞します。クエリグループが閉塞すると，保留していたタプルはすべて破棄されます。

タプルの保留数の上限値はシステムコンフィグプロパティファイル，クエリグループ用プロパティファイル，またはストリーム用プロパティファイルの stream.maxKeepTupleCount パラメーターで設定します。stream.maxKeepTupleCount パラメーターの詳細については，「8.6　システムコンフィグプロパティファイル（system_config.properties）」を参照してください。

11.8.7　フィルタリングによるタプルの選択

タイムスタンプ調整機能でタプルを保留するときに，あらかじめ定義しておいた条件演算式に従ってタプルのレコードの値をフィルタリングすることで，タプルを取捨選択してタプルの保留量を削減できます。

タイムスタンプ調整機能でタプルのフィルタリングをするかどうかは，クエリグループ用プロパティファイルまたはストリーム用プロパティファイルの stream.filterMode パラメーターで設定します。また，フィルタリングの条件演算式は，stream.filterCondition パラメーターで設定します。stream.filterMode パラメーターおよび stream.filterCondition パラメーターの詳細については，「8.6　システムコンフィグプロパティファイル（system_config.properties）」を参照してください。

11.8.8　タイムスタンプ調整の設定

タイムスタンプ調整機能について設定が必要な定義ファイルは次のファイルです。

- システムコンフィグプロパティファイル，クエリグループ用プロパティファイル，またはストリーム用プロパティファイル
  - stream.timestampAccuracy パラメーター
    タイムスタンプ調整機能で調整する時刻単位と時刻調整範囲を指定します。
  - stream.maxKeepTupleCount パラメーター
    タイムスタンプ調整機能で保留できるタプルの上限値を指定します。
- クエリグループ用プロパティファイルまたはストリーム用プロパティファイル
  - stream.filterMode パラメーター
    タイムスタンプ調整機能で，タプルのフィルタリングをするかどうかを指定します。
  - stream.filterCondition パラメーター
    タイムスタンプ調整機能で，タプルのフィルタリングをする場合の条件演算式を指定します。

12 Flex Dashboard の設定内容の詳細

この章では，ストリームデータ処理エンジンで処理したデータを Flex Dashboard に出力する場合の，設定内容の詳細について説明します。

12.1　この章の構成

この章では，ストリームデータの集計・分析結果をダッシュボードに出力して表示する場合に必要な設定，画面の構成，および定義ファイルの記述例について説明します。

ダッシュボードに出力する場合，Flex Dashboard の Dashboard Server および Dashboard Viewer の設定が必要です。この章で説明する Flex Dashboard の設定，定義ファイルの記述例，および参照先を次の表に示します。

表 12-1　Flex Dashboard の設定および参照先

項番	Flex Dashboard の設定	参照先
1	Dashboard Server のサーバ内設定ファイル（usrconf.properties）	12.2
2	Dashboard Viewer の画面編集用ファイル	12.3
3	ダッシュボード出力の定義例	12.4

12.2　Dashboard Server のサーバ内設定ファイル（usrconf.properties）

Dashboard Server を使用するには，Dashboard Server のサーバ内設定ファイル（usrconf.properties）を設定します。Dashboard Server のサーバ内設定ファイルの作成方法，設定できるキー，および記述例について説明します。

ファイルの作成方法

サーバ内設定ファイルは，インストール時に提供されるサンプルファイルを使用して作成することをお勧めします。サンプルファイルは，次のディレクトリに格納されていますので，コピーして編集してください。

＜インストールディレクトリ＞¥samples¥httppacket¥web¥containers¥uCSDPAF_Server¥usrconf¥

サンプルファイルでは，REFRESH_INTERVAL キーと RETRY_NUM キー以外は編集しないでください。

作成したサーバ内設定ファイルは，次のディレクトリに格納してください。

V01-50 より前

＜インストールディレクトリ＞¥psb¥CC¥web¥containers¥uCSDPAF_Server¥usrconf¥

V01-50 以降

＜インストールディレクトリ＞¥psb¥CC¥server¥usrconf¥ejb¥uCSDPAF_Server¥

設定できるキー

サーバ内設定ファイルで設定できるキーを次の表に示します。

表 12-2　サーバ内設定ファイルで設定できるキー

項番	キー名称	内容	デフォルト値
1	REFRESH_INTERVAL	Dashboard Server がダッシュボード出力コネクターにアクセスして，ダッシュボード表示用データを更新する間隔（単位：ミリ秒）を 0～600000 の整数値で指定します。	2000
2	RETRY_NUM	ダッシュボード表示用データの更新に失敗した場合の更新のリトライ回数（単位：回）を 0～100 の整数値で指定します。	0

サーバ内設定ファイルは，Java のプロパティ形式（<key>=<value>の形式）で記述してください。

記述例

サーバ内設定ファイルの設定例を次に示します。

```
REFRESH_INTERVAL=2000
RETRY_NUM=0
```

この例では，更新間隔を 2000 ミリ秒，リトライ回数を 0 回と指定しています。

12.3　Dashboard Viewer の画面編集用ファイル

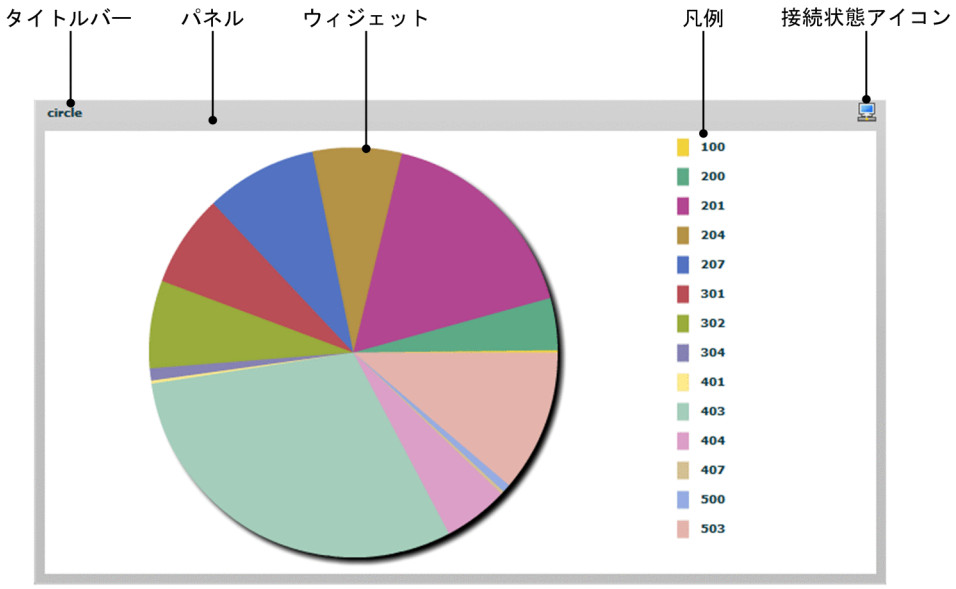
Dashboard Viewer で表示するダッシュボード画面では，Dashboard Viewer の画面編集用ファイルを編集して，グラフの表示レイアウトなどを変更できます。

ここでは，Dashboard Viewer の画面編集用ファイルで変更できる Dashboard Viewer の画面構成，Dashboard Viewer の画面編集用ファイルの詳細，および Dashboard Viewer でのダッシュボード表示用データの型の扱いについて説明します。

12.3.1　Dashboard Viewer の画面構成

Dashboard Viewer の画面編集用ファイルで編集できる Dashboard Viewer の画面構成を次の図に示します。なお，Dashboard Viewer が表示できる最大表示サイズは，1,920 ピクセル×1,200 ピクセルです。

図 12–1　Dashboard Viewer の画面構成



タイトルバー

ウィジェットのタイトルを表示する領域です。

パネル

ウィジェットやタイトルバーを表示するための領域です。ウィジェットの背面に表示されます。

ウィジェット

ストリームデータ処理の分析結果を指定した形式で表示したものです。ウィジェットの種別には，棒グラフ，円グラフ，散布図，折れ線グラフ，および表があります。なお，キャンバスに表示できるウィジェットは一つだけです。

ウィジェットごとのデータの表示順，およびウィジェットから別の URL への遷移方法を次に示します。

データの表示順

- 棒グラフ，円グラフ，および折れ線グラフ

  横軸のデータが昇順にソートされて表示されます。

- 表

  1 列目のデータが昇順にソートされて表示されます。

カラムヘッダーをクリックすると，クリックしたカラムヘッダーのカラムでソートします。クリックするごとに昇順，降順の順でソート規則が変わります。ソート対象のカラムに同一のデータがある場合，同一データ間の順番は順不同になります。

URL への遷移方法

ウィジェット上でダブルクリックすると，Dashboard Viewer の画面編集用ファイルに設定した遷移先 URL の情報が，新しいウィンドウで表示されます。ウィジェットごとの遷移先 URL への遷移方法を次の表に示します。

項番	ウィジェットの種類	遷移方法
1	棒グラフ	• 棒グラフの棒上をダブルクリックする。 • 凡例を表示している場合に，凡例をシングルクリックする。
2	円グラフ	• 円グラフの扇形上をクリックする。 • 凡例を表示している場合に，凡例をシングルクリックする。
3	散布図	凡例を表示している場合に，凡例をシングルクリックする。
4	折れ線グラフ	凡例を表示している場合に，凡例をシングルクリックする。
5	表	表の行をダブルクリックする。

凡例

ウィジェットの凡例です。Dashboard Viewer の画面編集用ファイルで，凡例表示を ON にした場合だけ表示されます。なお，凡例を表示する領域内に表示しきれない場合は，表示が欠けることがあります。

接続状態アイコン

ウィジェットの接続状態を示すアイコンです。接続状態アイコンを次の表に示します。

項番	アイコン	説明	内容
1		接続成功	ウィジェットのデータ取得に成功している場合に表示されます。
2		接続中	Dashboard Viewer の起動時，または接続を再開してデータを受信するまでの間に表示されます。
3		接続中止	ウィジェットのデータ取得を中止した場合に表示されます。
4		接続失敗	データ取得に失敗している場合に表示されます。
5		エラー発生	Dashboard Viewer の画面編集用ファイルの定義と受信データの間で不整合が発生している場合に表示されます。
6		未接続	Dashboard Viewer の画面編集用ファイルの定義に，必ず設定する必要がある，接続に関する項目が設定されていない場合に表示されます。

項番 1，または 2 のときにアイコンをクリックすると，ウィジェットのデータの取得を中止します。このとき，画面のデータはアイコンをクリックしたときの状態となります。

項番 3，または 4 のときにアイコンをクリックすると，接続をリトライします。このとき，アイコンは項番 2 になります。

12.3.2　Dashboard Viewer の画面編集用ファイルの詳細

ここでは，Dashboard Viewer の画面編集用ファイルの作成方法，記述形式，および定義の詳細について説明します。また，ダッシュボード表示用データを Dashboard Viewer で表示するときのデータの型の扱いについても説明します。

(1)　ファイルの作成方法

Dashboard Viewer の画面編集用ファイルのファイル名は，＜画面名＞.xml です。＜画面名＞には，16 文字までの半角英数字で，任意の画面名を指定してください。

なお，Dashboard Viewer の画面編集用ファイルは，インストール時に提供されるサンプルファイルを使用して作成することをお勧めします。次のサンプルファイルをコピーして編集してください。

＜インストールディレクトリ＞¥samples¥httppacket¥web¥containers¥uCSDPAF_Server¥layout¥dashboard.xml

作成した Dashboard Viewer の画面編集用ファイルは，次のディレクトリに格納してください。

V01-50 より前の場合

＜インストールディレクトリ＞¥psb¥CC¥web¥containers¥uCSDPAF_Server¥layout¥※

V01-50 以降の場合

＜インストールディレクトリ＞¥psb¥CC¥server¥public¥ejb¥uCSDPAF_Server¥layout¥※

注※

layout ディレクトリはユーザーが作成する必要があります。

Dashboard Viewer の画面編集用ファイルを作成するときの注意事項を次に示します。

- Dashboard Viewer の画面編集用ファイルの文字コードは，UTF-8 である必要があります。
- タグ内に記載した値の先頭と末尾にある半角空白，タブおよび改行は無視されます。
- 「(3)　定義の詳細」で説明されていないタグは，すべて無視されます。
- タグを省略した場合，デフォルトの値となります。

(2)　記述形式

Dashboard Viewer の画面編集用ファイルの記述形式の例を示します。

```
<dashboardDisplay>
  <connectionSetting>
    <serverName>localhost</serverName>
    <portNo>20421</portNo>
  </connectionSetting>
  <widgetGroup>
    <widget>
      <kind>table</kind>
      <dataSetting>
        <interval>5000</interval>
        <data1>
          <dataName>InprocessAPTest/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)　定義の詳細

dashboardDisplay タグ（全体情報の定義）

ダッシュボード画面の全体情報を定義します。

connectionSetting タグ（接続先サーバの定義）

接続先のサーバの情報を定義します。

serverName タグ

接続先の SDP サーバのサーバ名を 0～255 文字の半角英数字で指定します。デフォルト値は空文字です。空文字の場合，サーバに接続しません。

portNo タグ

接続先の SDP サーバのポート番号を 0～65,535 文字の半角整数で指定します。デフォルト値は空文字です。空文字の場合，サーバに接続しません。

widgetGroup タグ（ウィジェット全体の定義）

ウィジェット全体の情報を定義します。

widget タグ（ウィジェットの定義）

ウィジェットを定義します。

kind タグ

表示するウィジェットの種別を指定します。デフォルト値は，table です。指定できる値を次に示します。

- bar
  縦方向の棒グラフです。
- circle
  円グラフです。
- x-y
  散布図です。
- trend
  折れ線グラフです。
- table
  表です。

widgetSetting タグ（ウィジェットのパネルの定義）

ウィジェットのパネルの情報を定義します。

title タグ

ウィジェットのタイトルを，日本語を含む 0～32 文字の文字列で指定します。デフォルト値は空文字です。

border タグ（ウィジェットの枠線の定義）

ウィジェットの枠線を定義します。

Color タグ

ウィジェットの枠線の色を，RGB で指定できる色で指定します。デフォルト値は，192,192,192（灰色）です。

bg タグ（ウィジェットの背景の定義）

ウィジェットの背景を定義します。

なお，widget タグ下の kind タグが table の場合，表の背景に背景色および背景画像は表示されません。

Color1 タグ

ウィジェットの背景色を，RGB で指定できる色で指定します。デフォルト値は，255,255,255（白）です。

bgImage タグ

ウィジェットの背景に設定する画像ファイルの URL を，http://から始まる 0～255 文字の半角英数字で指定します。デフォルト値は空文字です。

指定できる画像ファイルは GIF 形式および JPG 形式です。指定した画像に動画が含まれる場合は，動作は保障されません。ウィジェットと画像のサイズが異なる場合，画像を拡大または縮小しないで，ウィジェットの中央地点と画像の中央地点を合わせた位置に表示します。

空文字を指定した場合，および画像が表示できない場合は，背景色が設定されます。

dataSetting タグ（ウィジェットのデータ設定の定義）

ウィジェットのデータ設定の情報を定義します。

interval タグ

Dashboard Viewer の表示の更新間隔，および Dashboard Viewer が Dashboard Server からデータを取得する間隔を 1～60000 の半角整数（単位：ミリ秒）で指定します。デフォルト値は 3000 です。

なお，起動直後や再接続直後は，データが表示されるまでに更新間隔の 2 倍程度の時間が掛かります。

data1 タグ～data5 タグ（データ系列の定義）

表示するデータを定義します。各データ系列は最大で 1,024 個保持できます。1,024 個を超えた場合，Dashboard Server に早く到着した順に破棄されます。

widget タグ下の kind タグが bar，circle，table の場合，data2 タグ～data5 タグの値は利用されません。

legendName タグ

表示するデータの凡例名を 0～30 文字の日本語を含む文字列で指定します。デフォルト値は空文字です。

なお，widget タグ下の kind タグが bar，circle，または table の場合，凡例名は表示されません。

dataName タグ

Dashboard Server から取得したいデータの接続名を 0～255 文字の半角英数字で指定します。デフォルト値は空文字です。

空文字を指定した場合，データ接続はしません。入力した接続名が存在しない場合は，エラーが発生します。

Dashboard Viewer 起動後，指定した接続名が存在しないか接続に失敗した場合は，接続状態アイコンが接続失敗のアイコンになります。

column1 タグ～column2 タグ

Dashboard Server から取得したデータの中から，どのカラムを表示するかを 0～255 文字の半角英数字で指定します。デフォルト値は空文字です。

widget タグ下の kind タグの値によって，指定する内容が異なります。

- bar

  column1 タグには横軸（ID）を指定します。column2 タグには縦軸（値）を指定します。

- circle

  column1 タグには扇形（ID）を指定します。column2 タグには扇形の大きさ（値）を指定します。

- x-y

  column1 タグには横軸（x 軸の値）を指定します。column2 タグには縦軸（y 軸の値）を指定します。

- trend

  column1 タグには横軸（時刻）を指定します。column2 タグには縦軸（値）を指定します。

- table

  値を使用しません。

kind タグが bar，circle，x-y，trend の場合，column1 タグまたは column2 タグに空文字を指定したときは，データは表示されません。

Dashboard Viewer 起動後に初めてデータを取得できた場合，データの中に指定したカラム名が存在しないときは，エラーメッセージ KFSP46954-E が出力されます。適切ではない型のときは，エラーメッセージ KFSP46955-E が出力されます。

color タグ

widget タグ下の kind タグが x-y，または trend の場合に，データを表示する色を RGB で指定できる色で指定します。デフォルト値は，0,0,0（黒）です。

filter タグ（フィルターの定義）

取得したデータに対するフィルターの情報を定義します。フィルターは各データに一つだけ設定できます。

columnName タグ

データにフィルターをかけるカラム名を 0～255 文字の半角英数字で指定します。デフォルト値は空文字です。

condition タグで定義するフィルターの条件を満たしていないデータを非表示にします。空文字の場合，データを非表示にしません。

Dashboard Viewer 起動後に初めてデータを取得できた場合，データの中に指定したカラム名が存在しないときは，エラーメッセージ KFSP46954-E が出力されます。適切ではない型のときは，エラーメッセージ KFSP46955-E が出力されます。

condition タグ

カラムの値と value タグで指定する値の間の条件を指定します。デフォルト値は eq です。

columnName タグに指定した値の型ごとに指定できる値が異なります。指定できる値を次に示します。

- columnName タグに文字列型（STRING）のカラム名を指定した場合

  eq（=）または ne（!=）のどちらかを指定できます。

- columnName タグに整数型（BYTE，SHORT，INT，LONG のどれか）のカラム名を指定した場合

lt（>），gt（<），le（>=），ge（<=），eq（=），ne（!=）のどれかを指定できます。

Dashboard Viewer 起動後に初めてデータを取得できた場合，上記の条件で指定できない値を指定したときは，エラーメッセージ KFSP46961-E が出力されます。

value タグ

比較する値を指定します。

columnName タグに指定した値の型ごとに指定できる値が異なります。指定できる値を次に示します。

- columnName タグに文字列型（STRING）のカラム名を指定した場合

  0～255 文字の半角英数字で指定します。

- columnName タグに整数型（BYTE，SHORT，INT，LONG のどれか）のカラム名を指定した場合

  32bit 整数型の範囲（-2147483648～2147483647）で指定します。

Dashboard Viewer 起動後に初めてデータを取得できた場合，上記の条件で指定できない値を指定したときは，エラーメッセージ KFSP46961-E が出力されます。

value タグの値は省略できません。

dataDisplay タグ（データ表示の定義）

ウィジェット内のデータ表示の情報を定義します。

threshold タグ（しきい値の定義）

表示しているデータに対するしきい値を定義します。

columnName タグ

しきい値の設定をするカラム名を 0～255 文字の半角英数字で指定します。デフォルト値は空文字です。

widget タグ下の kind タグが table の場合に，指定したしきい値を満たす行のすべての文字を赤く表示します。空文字を指定した場合，強調表示をしません。

Dashboard Viewer 起動後に初めてデータを取得できた場合，表示されるテーブルデータの中に，指定したカラム名が存在しないときは，エラーメッセージ KFSP46954-E が出力されます。適切でない型のときは，エラーメッセージ KFSP46955-E が出力されます。

condition タグ

カラムの値と value タグで指定した値の間の条件を指定します。デフォルト値は eq です。

columnName タグに指定した値の型ごとに指定できる値が異なります。指定できる値を次に示します。

- columnName タグに文字列型（STRING）のカラム名を指定した場合

  eq（=）または ne（!=）のどちらかを指定できます。

- columnName タグに整数型（BYTE，SHORT，INT，LONG のどれか）のカラム名を指定した場合

  lt（>），gt（<），le（>=），ge（<=），eq（=），ne（!=）のどれかを指定できます。

この条件に合わない値を指定した場合，エラーメッセージ KFSP46961-E が出力されます。

value タグ

比較する値を指定します。

columnName タグに指定した値の型ごとに指定できる値が異なります。指定できる値を次に示します。

- columnName タグに文字列型（STRING）のカラム名を指定した場合

0～255 文字の半角英数字で指定します。

- columnName タグに整数型（BYTE，SHORT，INT，LONG のどれか）のカラム名を指定した場合

  32bit 整数型の範囲（-2147483648～2147483647）で指定します。

この条件に合わない値を指定した場合，エラーメッセージ KFSP46961-E が出力されます。

value タグの値は省略できません。

xAxis タグ（x 軸の定義）

x 軸の設定情報を定義します。

max タグ

widget タグ下の kind タグが x-y の場合に，表示しているデータの x 軸方向の表示について，最大値を空文字，または 32bit の整数型の範囲（-2147483648～2147483647）で指定します。デフォルト値は空文字です。

最大値（max タグの値）<=最小値（min タグの値）となる値が指定された場合は，エラーメッセージ KFSP46961-E が出力されます。

最大値を空文字で指定した場合，自動でスケール変更されます。

なお，widget タグ下の kind タグが x-y 以外の場合，x 軸の最大値の設定はしません。

min タグ

widget タグ下の kind タグが x-y の場合に，表示しているデータの x 軸方向の表示について，最小値を空文字，または 32bit の整数型の範囲（-2147483648～2147483647）で指定します。デフォルト値は空文字です。

最大値（max タグの値）<=最小値（min タグの値）となる値が指定された場合は，エラーメッセージ KFSP46961-E が出力されます。

最小値を空文字で指定した場合，自動でスケール変更されます。

なお，widget タグ下の kind タグが x-y 以外の場合，x 軸の最小値の設定はしません。

yAxis タグ（y 軸の定義）

y 軸の設定情報を定義します。

max タグ

widget タグ下の kind タグが x-y または trend の場合に，表示しているデータの y 軸方向の表示について，最大値を空文字，または 32bit 整数型の範囲（-2147483648～2147483647）で指定します。デフォルト値は空文字です。

最大値（max タグの値）<=最小値（min タグの値）となる値が指定された場合は，エラーメッセージ KFSP46961-E が出力されます。

最大値を空文字で指定した場合，自動でスケール変更されます。

なお，widget タグ下の kind タグが x-y または trend 以外の場合，y 軸の最大値の設定はしません。

min タグ

widget タグ下の kind タグが x-y または trend の場合に，表示しているデータの y 軸方向の表示について，最小値を空文字，または 32bit 整数型の範囲（-2147483648～2147483647）で指定します。デフォルト値は空文字です。

最大値（max タグの値）<=最小値（min タグの値）となる値が指定された場合は，エラーメッセージ KFSP46961-E が出力されます。

最小値に空文字を指定した場合，自動でスケール変更されます。

なお，widget タグ下の kind タグが x-y または trend 以外の場合，y 軸の最小値の設定はしません。

xTimeRange タグ（時間軸の定義）

x 軸（時間軸）の設定情報を定義します。

range タグ

widget タグ下の kind タグが trend の場合に，表示しているデータの x 軸の表示時間を 1～86400（単位：秒）の半角整数で指定します。1 画面に入る表示時間を指定してください。デフォルト値は 300 です。

なお，widget タグ下の kind タグが trend 以外の場合，x 軸について表示時間の設定はしません。

SliderFlag タグ

widget タグ下の kind タグが trend の場合に，時間軸（x 軸）のスライダーを表示するかどうかを指定します。デフォルト値は OFF です。

指定できる値（半角大文字）を次に示します。

- ON
  スライダーを表示します。
- OFF
  スライダーを表示しません。

なお，widget タグ下の kind タグが trend 以外の場合，時間軸（x 軸）のスライダー表示の設定はしません。

tableProperties タグ（表の定義）

表の設定情報を定義します。

columnHeaderList タグ（カラムヘッダーの定義）

widget タグ下の kind タグが table の場合に，表示するカラムヘッダーの情報を定義します。

column タグ

表示するカラムについて，カラムごとの情報を定義します。

タグの数が 16 個を超える場合，エラーメッセージ KFSP46966-E が出力されます。

actualName タグ

受信データに含まれるカラムのうち，データを表示したいカラム名を 0～255 文字の半角英数字で指定します。デフォルト値は空文字です。

空文字を指定した場合，ソートはしません。

Dashboard Viewer 起動後に初めてデータを取得できた場合，データの中に指定したカラム名が存在しないときは，エラーメッセージ KFSP46954-E が出力されます。

displayName タグ

カラムヘッダーに表示するカラム名を 0～32 文字の日本語を含む文字列で指定します。デフォルト値は空文字です。

legend タグ（凡例の定義）

凡例の情報を定義します。

displayFlag タグ

グラフの右側に凡例を表示するかどうかを指定します。デフォルト値は OFF です。

指定できる値（半角大文字）を次に示します。

- ON
  凡例を表示します。
- OFF

凡例を表示しません。

widget タグ下の kind タグに指定する値によって，表示される凡例の内容が異なります。

- bar
  横軸に表示された名前の一覧を表示します。
- circle
  円グラフ内の扇形が持つ名前の一覧を表示します。
- x-y
  data1 タグ～data5 タグごとに legendName タグで指定した値を表示します。
- trend
  data1 タグ～data5 タグごとに legendName タグで指定した値を表示します。
- table
  ON を指定しても凡例は表示されません。

width タグ

凡例の大きさを 1～2048 の半角整数（単位：ピクセル），または 1～100 の半角整数と%で指定します。半角整数と%で指定した場合，ウィジェットに対する凡例の大きさを割合で指定します。半角整数と%で指定した結果，凡例を表示しきれない場合は，表示しきれるサイズに変更されます。

デフォルト値は 25%です。

jumpToURL タグ（URL 遷移の定義）

URL 遷移の情報を定義します。

URL タグ

遷移先の URL を，http://から始まる 0～128 文字の半角英数字で指定します。デフォルト値は空文字です。

空文字を指定した場合，URL 遷移をしません。

parameter1 タグ

選択したデータが持つ情報のうち，URL 遷移先に渡す引数 1 の引数名を 0～16 文字の半角英数字で指定します。デフォルト値は空文字です。

空文字を指定した場合，URL 遷移先に情報を渡しません。

value1 タグ

選択したデータが持つ情報のうち，URL 遷移先に渡す引数 1 の値を 0～16 文字の半角英数字で指定します。デフォルト値は空文字です。

widget タグ下の kind タグに指定する値によって，指定する値（URL 遷移先に渡せる値）が異なります。

- bar
  ID（横軸）の値を指定します。
- circle
  ID（横軸）の値を指定します。
- x-y
  legend の値を指定します。
- trend
  legend の値を指定します。
- table

column1～column16 の値を指定します。

widget タグ下の kind タグに table を指定し，Dashboard Viewer 起動後に初めてデータを取得できた場合，column[n]の[n]の値が表示されているカラム数以上のときは，エラーメッセージ KFSP46961-E が出力されます。また，column[n]に対して，[n]番目のカラムの型が文字列型でないときは，エラーメッセージ KFSP46955-E が出力されます。

(4)　Dashboard Viewer での型の扱い

ダッシュボード表示用データを Dashboard Viewer で表示するとき，Dashboard Viewer では，そのデータは次の表に示す型で扱われます。

項番	ダッシュボード出力コネクター定義で指定できる型	Dashboard Viewer での型の扱い	値の範囲
1	BYTE	整数	-2^{31}～$(2^{31})-1$
2	SHORT		
3	INT		
4	LONG	整数	-2^{53}～2^{53}※1
5	FLOAT	倍精度実数※2	1.79769313486231×(10^{308})～-1.79769313486231×(10^{308})
6	DOUBLE		
7	BIG_DECIMAL	文字列型	0～1,024 文字※3
8	STRING		
9	DATE	時刻型※4	上限※1：1970 年 1 月 1 日 0 時 0 分 0.000 秒 下限※1：9999 年 12 月 31 日 23 時 59 分 59.999 秒
10	TIME		
11	TIMESTAMP		

注※1

下限を下回る値を受信した場合は，その値は下限の値に置き換えられます。上限を上回る値を受信した場合は，その値は上限の値に置き換えられます。

注※2

値の範囲は，IEEE 754 に準拠します。ダッシュボード出力コネクターと Dashboard Viewer 間で，値の丸めは発生しません。

注※3

上限を上回る長さの文字列を受信した場合，その文字列は上限の長さで打ち切られます。2 バイト文字は 1 文字として扱われます。

注※4

TIMESTAMP 型の表示形式を次に示します。

YYYY-MM-DD hh:mm:ss.SSS

YYYY：年

MM：月

DD：日

hh：時

mm：分

ss：秒

SSS：ミリ秒

12.4　ダッシュボード出力の定義例

ここでは，次の内容をダッシュボードに出力する場合の，アダプター構成定義ファイルの記述例，および記述例の内容を説明します。

- ストリームデータの集計・分析結果の最新データだけを表示する。
- ストリームデータの集計・分析結果の履歴を含むデータを表示する。

ここでは，出力アダプター定義について説明します。入力アダプターについては，「9.13.2　記述例 2」の入力アダプターに関する記述を参照してください。

12.4.1　出力アダプター定義（最新のデータの表示）

棒グラフ，円グラフ，散布図または表で，ストリームデータの集計・分析結果の最新のデータをダッシュボードに表示する場合，クエリ定義ファイルでの rstream の最新の計算結果だけを表示できるようにします。

ここでは，最新のデータを表示する場合の出力アダプター定義の記述例を示したあとに，記述例の内容を説明します。

(1)　記述例

```
<adp:OutputAdaptorDefinition name="OutputAdaptor1" charCode="MS932" lineFeed="CR_LF">

    <!-- 受信用CB定義 -->
    <cb:ReceiveCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.ReceiveCBImpl" name="receiver">
        <cb:streamInfo name="QUERY" querygroup="Inprocess_QueryGroupTest"/>
    </cb:ReceiveCBDefinition>

    <!-- データ編集用CB定義 -->
    <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.OutputMappingCBImpl"
name="editor1">
        <!-- マッピング定義 -->
        <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: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>

    <!-- データ編集用CB定義 -->
    <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor2">
        <!-- マッピング定義 -->
        <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.DashboardOutputCBImpl"
name="outputer">
        <!-- ダッシュボード出力コネクタ定義 -->
        <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)　記述例の内容

この記述例では，ストリームデータの集計・分析結果のデータを出力アダプター「OutputAdaptor1」でダッシュボードへ出力します。

出力アダプター「OutputAdaptor1」では，次の表に示す処理を実施します。なお，括弧内は，アダプター構成定義ファイル内での定義名です。

コールバックの種類	コールバックでの処理
受信用コールバック（受信用 CB 定義）	タプル受信（出力ストリーム定義）
編集用コールバック（編集用 CB 定義）	レコードとストリーム間のマッピング（マッピング定義）
	レコード間のマッピング（マッピング定義）
	フォーマット変換（フォーマット変換定義）
送信用コールバック（送信用 CB 定義）	ダッシュボードへの出力（ダッシュボード出力コネクター定義）

出力アダプター「OutputAdaptor1」の各定義での定義内容を次に示します。

- 出力ストリーム定義の内容

  クエリグループ「Inprocess_QueryGroupTest」の出力ストリーム「QUERY」からタプルを受信して，共通形式レコードに変換します。

- マッピング定義の内容

  レコードとストリーム間のマッピングで，出力ストリームの形式に対応した共通形式レコードと，フォーマット変換で入力する共通形式レコードとを関連づけます。

また，レコードの間のマッピングで，rstream の最新の結果がどのレコードかがわかるように，タプルの時刻を取得（map タグの function 属性に getTupleTime を指定）して共通形式レコードに出力します。

- **フォーマット変換定義の内容**

  マッピングで出力された共通形式レコードを出力形式レコードに変換します。

- **ダッシュボード出力コネクター定義の内容**

  フォーマット変換で出力された出力形式レコードに対し，レコード保持期間（RecordHoldTime タグの RecordTime 属性）に 0 を設定し，最新の結果以外のレコードをすべて削除するようにします。

- **アダプター構成定義ファイルで指定するストリーム名**

  アダプター構成定義ファイルで指定するストリーム名は，クエリ定義ファイルで指定した内容と合わせてください。

  - 入力ストリームの場合

    クエリ定義ファイルの register stream 句で指定したストリーム名「STREAM」が，アダプター構成定義ファイルで指定する入力ストリーム名となります。

  - 出力ストリームの場合

    クエリ定義ファイルの register query 句で指定したクエリ名「QUERY」が，アダプター構成定義ファイルで指定する出力ストリーム名となります。

  この記述例の場合のクエリ定義ファイルを次に示します。この定義ファイルでは，送信先 IP アドレス（sendip）ごとに，直近 1 分間の中での送受信時間（subtime）の最大値を求め，その結果を rstream で出力する処理を記述しています。

```
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);
```

  クエリの定義については，マニュアル「uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド」を参照してください。

12.4.2　出力アダプター定義（履歴を含むデータの表示）

折れ線グラフなどで，現時点までの履歴を含む結果をダッシュボードに表示する場合，現時点から過去 n 秒までの rstream の計算結果を表示できるようにします。

この場合，出力アダプター定義のダッシュボード出力コネクター定義では，レコード保持期間（RecordHoldTime タグの RecordTime 属性）に n を設定して，getTupleTime の結果を格納するフィールドに，現時点から過去 n 秒までの結果を保持する設定をします。

これ以外の出力アダプターの記述例，および記述例の内容については，「12.4.1　出力アダプター定義（最新のデータの表示）」を参照してください。

付録

付録 A　各バージョンの変更内容

表 A-1　変更内容(3020-3-V02-10)　uCosminexus Stream Data Platform - Application Framework　01-05

追加・変更内容
リレーションを生成しないで直接ストリームデータに対して演算（ストリーム間演算）を実行できるようにした。 また，CQL で，ユーザーが Java でプログラミングした外部定義関数を使用できるようにした。 ストリームデータに対する演算機能の追加に伴い，操作系 CQL に次の関数を追加した。 • ストリーム間演算関数 • 外部定義ストリーム間演算関数 システムコンフィグプロパティファイル，およびクエリグループ用プロパティファイルに，次のパラメーターを追加した。 • engine.externalStreamFuncVerifyMode また，次の定義ファイルを追加した。 • 外部定義関数定義ファイル
Dashboard Server の起動順序，および停止順序を変更した。
クエリグループの登録後にプロパティファイルの設定値を変更する場合の説明を変更した。
sdpstartinpro コマンドの引数に，インプロセス連携のカスタムアダプターに渡す引数を追加した。また，この引数の注意事項を追加した。
sdptrced コマンドの-time オプションの形式を変更した。

表 A-2　uCosminexus Stream Data Platform - Application Framework　01-01

追加・変更内容
適用 OS に Linux を追加した。
英語版のサンプルファイルを追加した。
メッセージの出力言語を変更できるようにした。
Dashboard Viewer の URL を変更した。
データ型種別が DATE の場合の表記規則で，指定範囲外の値が指定された場合の説明を変更した。
レコード間条件の判定で，レコード内の時刻情報が同一である複数のレコードが入力された場合の説明を追加した。

付録 B　このマニュアルの参考情報

このマニュアルを読むに当たっての参考情報を示します。

付録 B.1　関連マニュアル

関連マニュアルを次に示します。必要に応じてお読みください。

- **ストリームデータ処理基盤　uCosminexus Stream Data Platform - Application Framework 解説（3020-3-V01）**

  Stream Data Platform - AF の概要や前提知識について説明しています。

  Stream Data Platform - AF の特長やシステム構成などの概要，およびシステムを構築・運用するために必要な前提知識を習得したい場合に参照してください。

- **ストリームデータ処理基盤　uCosminexus Stream Data Platform - Application Framework アプリケーション開発ガイド（3020-3-V03）**

  Stream Data Platform- AF でデータを分析するために使用する CQL の記述方法，およびカスタムアダプターなどのアプリケーションを作成する方法について説明しています。

  分析目的に合わせた CQL を記述したり，API でカスタムアダプターなどのアプリケーションを作成したりする場合に参照してください。

- **ストリームデータ処理基盤　uCosminexus Stream Data Platform - Application Framework メッセージ（3020-3-V04）**

  Stream Data Platform- AF が出力するメッセージについて説明しています。

  メッセージが出力された際に，必要に応じて参照してください。

なお，このマニュアルでは，関連マニュアルについて，「ストリームデータ処理基盤」を省略して表記しています。

付録 B.2　このマニュアルでの表記

このマニュアルでは，製品名および Java 関連用語を次のように表記しています。

表記		製品名または Java 関連用語
jar		Java™ Archive
Java		Java™
JavaVM		Java™ Virtual Machine
JDK		Java™ Development Kit
Linux		Linux(R)
Linux	Red Hat Enterprise Linux 5	Red Hat Enterprise Linux(R) 5 Advanced Platform（AMD/Intel 64）
		Red Hat Enterprise Linux(R) 5（AMD/Intel 64）
	Red Hat Enterprise Linux 6	Red Hat Enterprise Linux(R) Server 6（64-bit x86_64）

表記	製品名または Java 関連用語
Stream Data Platform - AF	uCosminexus Stream Data Platform - Application Framework

付録 B.3　英略語

このマニュアルで使用する英略語を次に示します。

英略語	英語での表記
AP	Application Program
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BOM	Byte Order Mark
CB	Callback
CPU	Central Processing Unit
CQL	Continuous Query Language
EUC	Extended UNIX Code
GMT	Greenwich Mean Time
HTTP	Hyper Text Transfer Protocol
IEEE	Institute of Electrical and Electronic Engineers
I/O	Input/Output
IP	Internet Protocol
JIS	Japanese Industrial Standards
RMI	Remote Method Invocation
TCP	Transmission Control Protocol
UCS	Universal multi-octet coded Character Set
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF	UCS Transformation Format
W3C	World Wide Web Consortium
XML	Extensible Markup Language

付録 B.4　KB（キロバイト）などの単位表記について

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

索引

P

Q

R

S

T

U

W

あ

い

う

え

お

か

き

く

こ

さ

し

す

せ

そ

た

ち

て

と

な

に

は

ひ

ふ

へ

ま

め

も

り

れ

ろ