HITACHI
Inspire the Next

Hitachi Streaming Data Platform
概説

3000-3-E22-31

前書き

■ 著作権

■ 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

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

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

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

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 Daisuke Okajima and Kohsuke Kawaguchi (http://relaxngcc.sf.net/).

This product includes software developed by Andy Clark.

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

■ 発行

2020 年 7 月

変更内容

変更内容(3000-3-E22-31)　Hitachi Streaming Data Platform 03-30，Hitachi Streaming Data Platform software development kit 03-30

追加・変更内容	変更個所
記述不良を見直した。	－

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，Hitachi Streaming Data Platform （Streaming Data Platform）の概要や前提知識について説明したものです。Streaming Data Platform 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の習得を目的としています。

■ 対象製品

- P-9W64-9F31 Hitachi Streaming Data Platform 03-30 （適用 OS：Red Hat(R) Enterprise Linux(R) Server 6（64-bit x86_64），Red Hat(R) Enterprise Linux(R) Server 7（64-bit x86_64））
- P-9W64-9K31 Hitachi Streaming Data Platform software development kit 03-30 （適用 OS：Red Hat(R) Enterprise Linux(R) Server 6（64-bit x86_64），Red Hat(R) Enterprise Linux(R) Server 7（64-bit x86_64））

■ 対象読者

このマニュアルは，ソリューションデベロッパー，およびインテグレーションデベロッパーを対象にしています。

■ このマニュアルで使用する記号

このマニュアルで使用する記号を次に示します。

記号	意味
太字	本文中では次を表します。 　キーボードのキー，パラメーター名，プロパティ名，ハードウェアラベル，ハードウェアボタン，ハードウェアスイッチ 手順中では次を表します。 　ユーザーインタフェース
斜体	可変値，強調，呼び出しを示します。
＜ ＞	可変値（斜体で変数を表現しきれない場合に使用）
[ ]	任意の値であることを示します。
{ }	必要な値，または期待される値を示します。
| （ストローク）	複数の項目または引数から選択することを示します。
… （点線）	この記号の直前に示された項目を繰り返して指定できることを示します。

■ このマニュアルで使用する略語

このマニュアルで使用する製品名の表記

このマニュアルでは，製品名を次のように表記しています。

表記	製品名
HSDP	Hitachi Streaming Data Platform
Streaming Data Platform	
SDP	
HSDP software development kit	Hitachi Streaming Data Platform software development kit
Streaming Data Platform software development kit	
SDP SDK	
JavaVM	Java Virtual Machine
Red Hat Enterprise Linux Server	Red Hat(R) Enterprise Linux(R) Server

このマニュアルで使用する英略語

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	正式名称
AP	Application Program
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BOM	Byte Order Mark
CPU	Central Processing Unit
CQL	Continuous Query Language
CSV	Comma Separated Value
EADs	Hitachi Elastic Application Data Store
EUC	Extended UNIX Code
EUC-JP	Extended UNIX Code Packed Format for Japanese
G1GC	Garbage First Garbage Collection
GC	Garbage Collection
GCC	GNU Compiler Collection

英略語	正式名称
GMT	Greenwich Mean Time
HTTP	Hyper Text Transfer Protocol
ID	Identifier
IP	Internet Protocol
JDBC	Java Database Connectivity
JIS	Japanese Industrial Standards
JRE	Java Runtime Environment
JST	Japan Standard Time
JVM	Java Virtual Machine
NIC	Network Interface Card
OS	Operating System
SJIS	Shift JIS
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP/IP	User Datagram Protocol/Internet Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF	UCS Transformation Format
VM	Virtual Machine
VTL	Velocity Template Language
XML	Extensible Markup Language

■ このマニュアルで使用する KB（キロバイト）などの単位表記

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。

■ 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。

- 『Hitachi Streaming Data Platform 入門』 (3000-3-E21)
- 『Hitachi Streaming Data Platform システム構築ガイド』 (3000-3-E23)
- 『Hitachi Streaming Data Platform アプリケーション開発ガイド』 (3000-3-E24)
- 『Hitachi Streaming Data Platform メッセージ』 (3000-3-E25)

目次

1 Streaming Data Platform とは

Streaming Data Platform は，リアルタイムに出力され続ける多大なデータを分析できるストリームデータ処理を実現するための製品です。この章では，Streaming Data Platform の概要および特長について説明します。またこの章では，現行のワークフローへの Streaming Data Platform の導入例と，Streaming Data Platform の設定と稼働に必要なシステム構成についても説明します。

1.1 「今」を分析するデータ処理システム

社会インフラの変化によって，携帯電話，IC カード，家電製品など，身の回りにあるさまざまなものが多大な情報を持つようになっています。その結果，データ処理システムが扱う情報量は日々増え続けています。そして，これらの情報の中には，迅速に集計・分析することで新たな価値を生むものがあります。リアルタイムに出力され続ける多大な「今」の情報の中から新たな価値を生み出すことは，データ処理システムへの要求の一つになっています。

Streaming Data Platform は，**ストリームデータ処理**を実現することで，この要求に応えます。ストリームデータ処理によって，連続して発生する多大な時系列順のデータ（ストリームデータ）に対して，発生時からのタイムラグなしでリアルタイムに集計・分析できます。

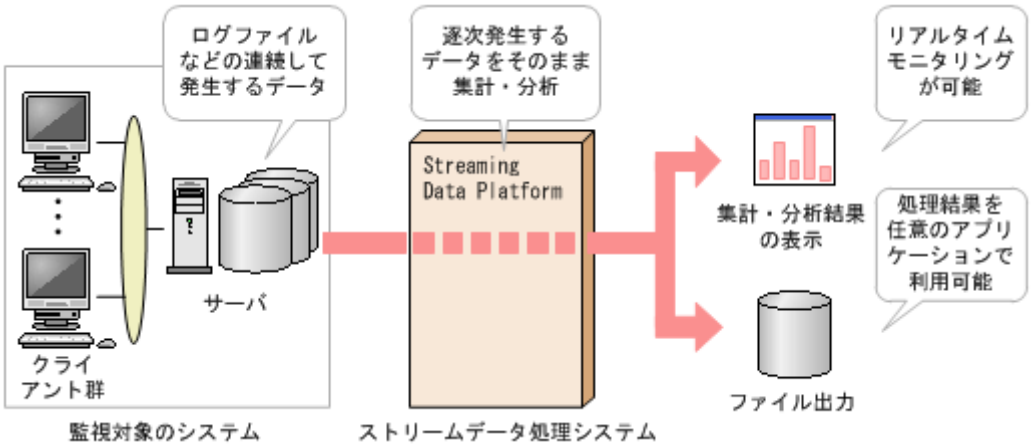
例えば，PC や携帯電話からアクセスできる検索サイトが持つ情報をリアルタイムに集計・分析することで，商品の販売好機がわかります。ある商品がコミュニティサイトなどで話題になり，需要の増加が見込まれる場合，検索サイトでは対象の商品名の検索回数が増加する傾向があります。ストリームデータ処理によって，検索回数をリアルタイムに集計・分析することでそのような商品を特定できます。それによって，販売店側では迅速に商品を追加発注でき，メーカー側では商品の増産を迅速に決断できます。

また，IT システムでは，高効率での運用やコストの削減が求められ，仮想化やクラウドコンピューティングの採用が加速しています。この結果，システム構成の大規模化・複雑化が起き，IT システムの稼働状況は見えにくくなっています。そのため，障害の発生時に，障害の検知や対策に時間が掛かることがあります。しかし，ストリームデータ処理によって多大なデータを集計・分析し続け，システムの稼働状況をリアルタイムにモニタリングすれば，障害発生時に迅速な対応ができます。また，システムの稼働情報の傾向分析や相関分析によってシステムの障害予兆を検知し，障害の発生防止に役立てることができます。

このように，ストリームデータ処理を実現する Streaming Data Platform のデータ処理システムへの導入は，多大な情報を処理するためのアプローチとして適しています。

Streaming Data Platform を利用したストリームデータ処理の概要を次の図に示します。

図 1-1　Streaming Data Platform を利用したストリームデータ処理の概要



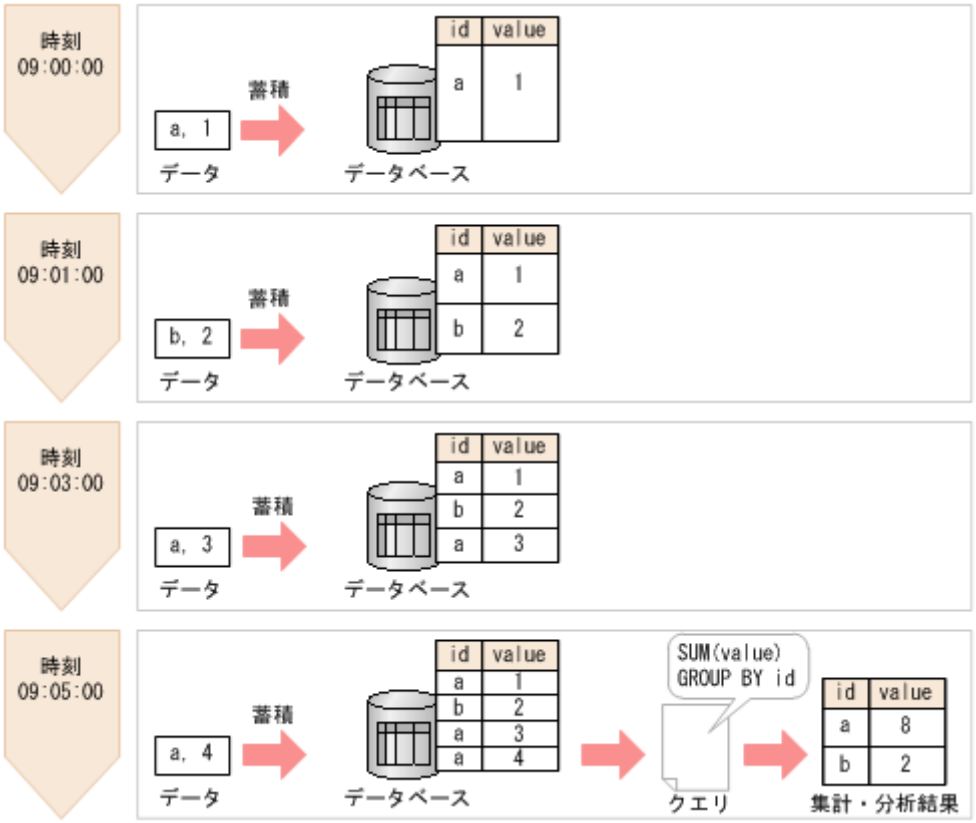
Streaming Data Platform を導入したストリームデータ処理システムでは，連続して発生するデータをそのまま集計・分析の対象にできます。

例えば，ストリームデータ処理システムでシステムの稼働を監視する場合，サーバが出力するログファイルや，ネットワーク上の HTTP パケットなどを集計・分析できます。この処理結果をファイルに出力することで，システムの稼働状況をリアルタイムにモニタリングできます。これによって，システムの障害発生時に迅速に対処でき，運用効率や保守効率を向上できます。また，処理結果をファイルに出力すれば，任意のアプリケーションで利用できます。

ストリームデータ処理が，どのようにしてリアルタイムな処理を実現しているのか，従来のストック型データ処理と比較して説明します。

従来のストック型データ処理を次の図に示します。

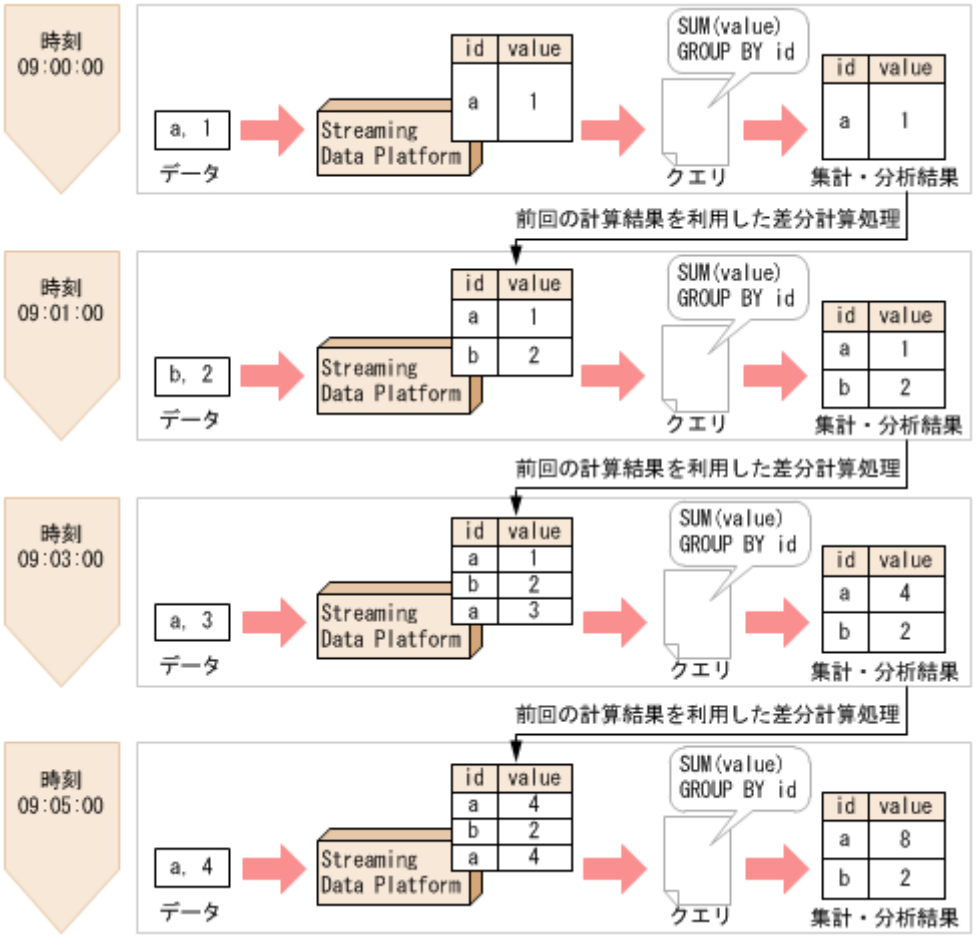
図 1-2　ストック型データ処理



ストック型データ処理を利用したデータ処理では，発生したデータは順次データベースなどに蓄積されます。蓄積されたデータはクエリの発行を契機に処理され，集計・分析結果を得ます。問い合わせを受けてから，あらかじめデータベースに格納されたデータを検索するため，得られたデータの集計・分析結果には，データ発生時からのタイムラグが生じます。図に示した処理では，09:00:00 に発生したデータが，09:05:00 のクエリの発行によって処理され，データ発生時からのタイムラグが生じていることがわかります。

これに対して，ストリームデータ処理を次の図に示します。

図 1-3　ストリームデータ処理



ストリームデータ処理では，クエリ（集計・分析シナリオ）をあらかじめ登録しておき，最小限の計算処理をする差分計算処理を実行します。この計算処理は，データの入力を契機に実行されるため，データの発生からのタイムラグのない，リアルタイムな集計・分析結果が得られます。データの入力を契機に処理が実行されるストリームデータ処理は，データが次々に発生する場合に効果的です。

このように，Streaming Data Platform を導入してストリームデータ処理を実現することで，リアルタイムな集計・分析を行えます。

1.2 Streaming Data Platform の特長

Streaming Data Platform には，次の特長があります。

- 多大な時系列データの高速処理
- プログラミングを必要としない集計・分析シナリオの定義

それぞれの特長について説明します。

1.2.1 多大な時系列データの高速処理

Streaming Data Platform では，**インメモリ処理**と**差分計算処理**を併用することで，多大な時系列データを高速処理しています。

インメモリ処理

インメモリ処理では，ディスクドライブを使用しないで，メモリ上で処理を実行します。

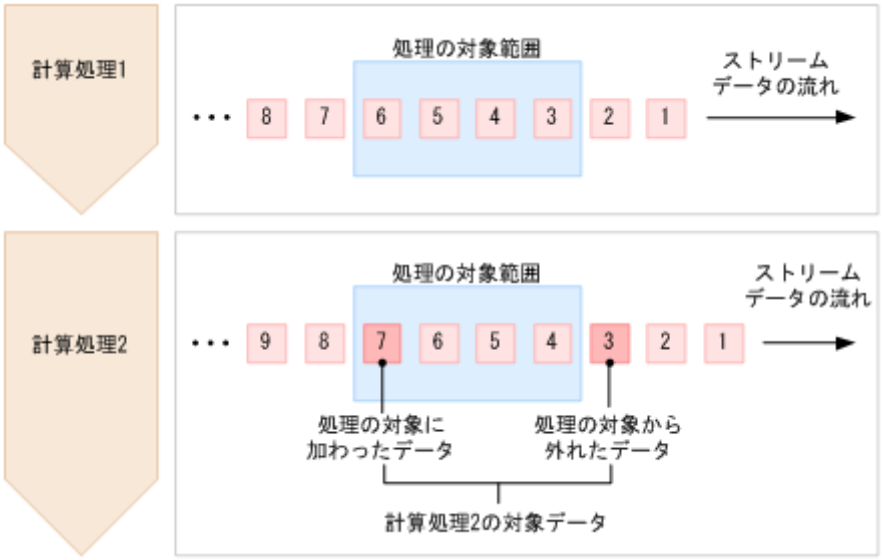
多大なデータを処理する場合，ディスクドライブとの入出力処理に掛かる時間を無視できなくなります。Streaming Data Platform では，このディスクドライブとの入出力処理をなくし，メモリ上でデータを処理することで，データの処理速度を高速化しています。

差分計算処理

差分計算処理では，クエリをあらかじめ登録しておき，データの入力を契機に逐次処理して，処理結果を次の処理に利用します。したがって，次の処理では処理対象のデータ全体を計算する必要がなく，変化量の計算処理だけで済みます。

Streaming Data Platform でのストリームデータの差分計算を次の図に示します。

図 1-4　ストリームデータの差分計算



この図に示すとおり，Streaming Data Platform は，最初に計算処理 1 を実行します。次のストリームデータが到来したとき，計算処理 2 では，計算処理 1 の結果から，対象から外れた 3 番目のデータを引き，新たに対象となった 7 番目のデータを加える処理だけを実行します。このように最小限の計算処理をすることで，データの処理速度を高速化しています。

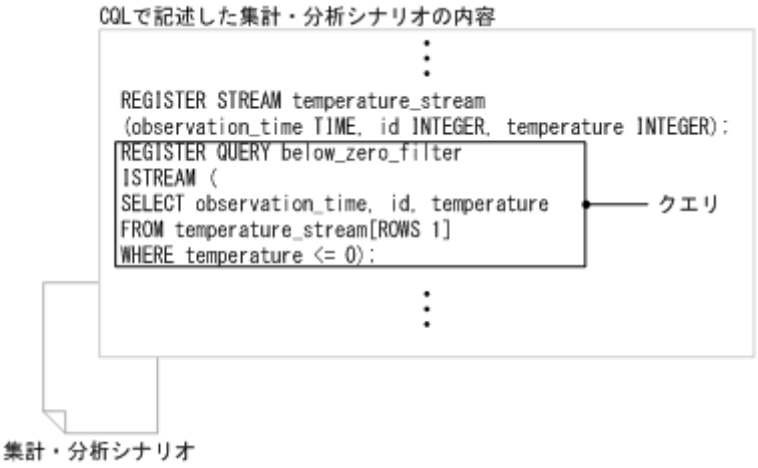
1.2.2　プログラミングを必要としない集計・分析シナリオの定義

ストリームデータ処理の内容は，**集計・分析シナリオ**に定義します。この集計・分析シナリオの定義は，データベースで標準的に扱われる SQL に類似した言語である CQL で記述します。そのため，集計・分析シナリオを定義するときに，分析専用アプリケーションの作成は必要ありません。集計・分析シナリオを変更する場合も，CQL で記述した定義ファイルの書き換えだけで済みます。

CQL で記述するストリームデータ処理の内容をクエリといいます。集計・分析シナリオ内には，複数のクエリが定義できます。

例えば以下の図は，それぞれ ID が与えられた複数の観測所を持つ気温観測システム向けの，CQL で書かれたサマリ分析シナリオです。観測データ中の全氷点下のデータを集計・分析することがクエリの目的です。

図 1-5　CQL を使用した集計・分析シナリオの記述例



CQL は，クエリ言語として汎用的な言語であり，いろいろな処理を記述できます。複数のクエリを組み合わせることで，さまざまな業務に対応する集計・分析シナリオを定義できます。

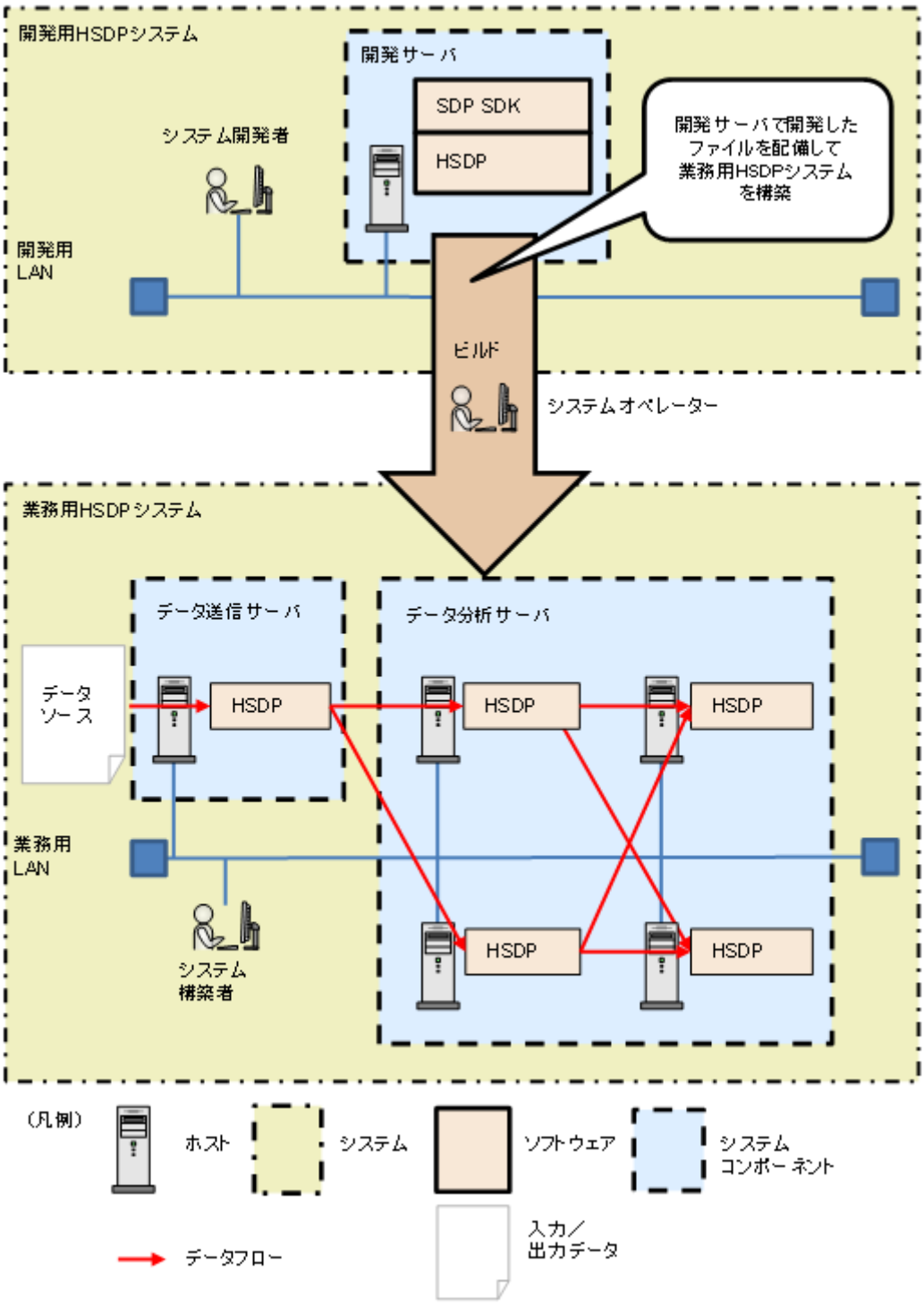
2 HSDP を利用するシステムの概要

この章では，システムコンポーネント，ソフトウェアコンポーネントについて説明します。

2.1 システムコンポーネント

Hitachi Streaming Data Platform では，次々に生成される時系列データ（ストリームデータ）のリアルタイムな処理をメモリ上で実現します。ストリームデータは，CQL によるユーザー定義の分析シナリオに基づいて処理されます。Streaming Data Platform のシステムコンポーネントとは，HSDP を利用するシステムを構成するサーバ，つまり，開発サーバ，データ送信サーバ，およびデータ分析サーバのことを指します。

HSDP システムの例



Streaming Data Platform のシステムコンポーネントは次のとおりです。

表 2-1　Streaming Data Platform のシステムコンポーネント

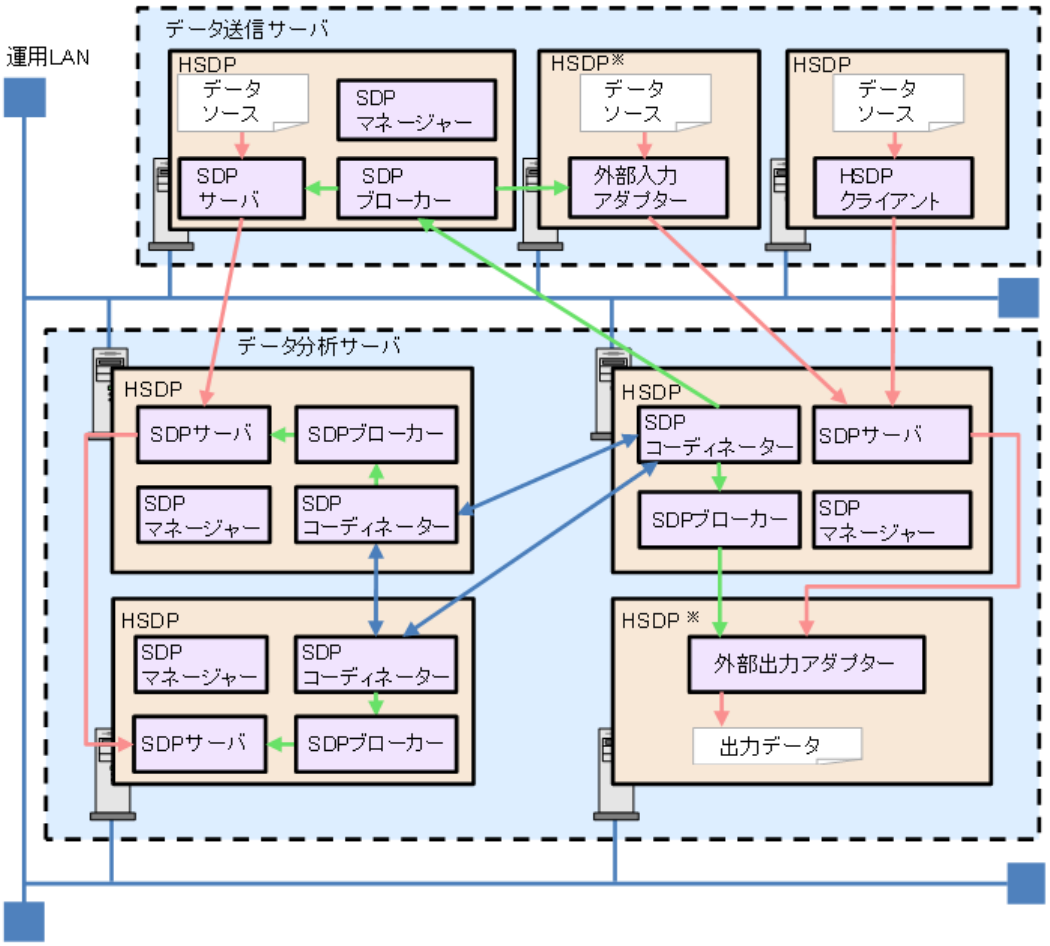
項番	コンポーネント	説明
1	開発サーバ	• 開発サーバには，Streaming Data Platform および Streaming Data Platform software development kit をインストールします。

項番	コンポーネント	説明
1	開発サーバ	• このサーバは分析シナリオ用の開発環境を提供します。また，SDP システムで使用されるストリームデータを送受信するアダプターの開発環境も提供します。 • システム開発者は，Streaming Data Platform software development kit で提供される API およびツールを使用して，分析シナリオやアダプターを開発し，テストできます。
2	データ送信サーバ	• データ送信サーバには，Streaming Data Platform をインストールします。 • このサーバは，データソースからデータ分析サーバにストリームデータを出力します。 • Streaming Data Platform は，ストリームデータの出力形式として，標準でテキストファイルや TCP などをサポートしています。 • システム構築者は，Streaming Data Platform software development kit の API を利用してアダプターを作成することで，標準提供出力形式以外の任意の形式でデータを出力できます。
3	データ分析サーバ	• データ分析サーバには，Streaming Data Platform をインストールします。 • このサーバは，データ送信サーバから受信したストリームデータを（ユーザーが開発した分析シナリオに基づいて）処理し，処理したストリームデータを出力します。 • Streaming Data Platform が標準でサポートするテキストファイルや TCP などの出力形式，または Streaming Data Platform software development kit の API を利用してユーザーが作成したアダプターによる任意の出力形式で，分析結果を出力できます。 • データ分析サーバは，処理したストリームデータをほかのデータ分析サーバに送信することもできます。したがって，システム構築者は，複数のデータ分析サーバを接続することで拡張性のあるシステムを構築できます。

2.2　ソフトウェアコンポーネント

HSDP を利用したシステムでの，業務システムおよび開発システムそれぞれのソフトウェアコンポーネントを次に示します。

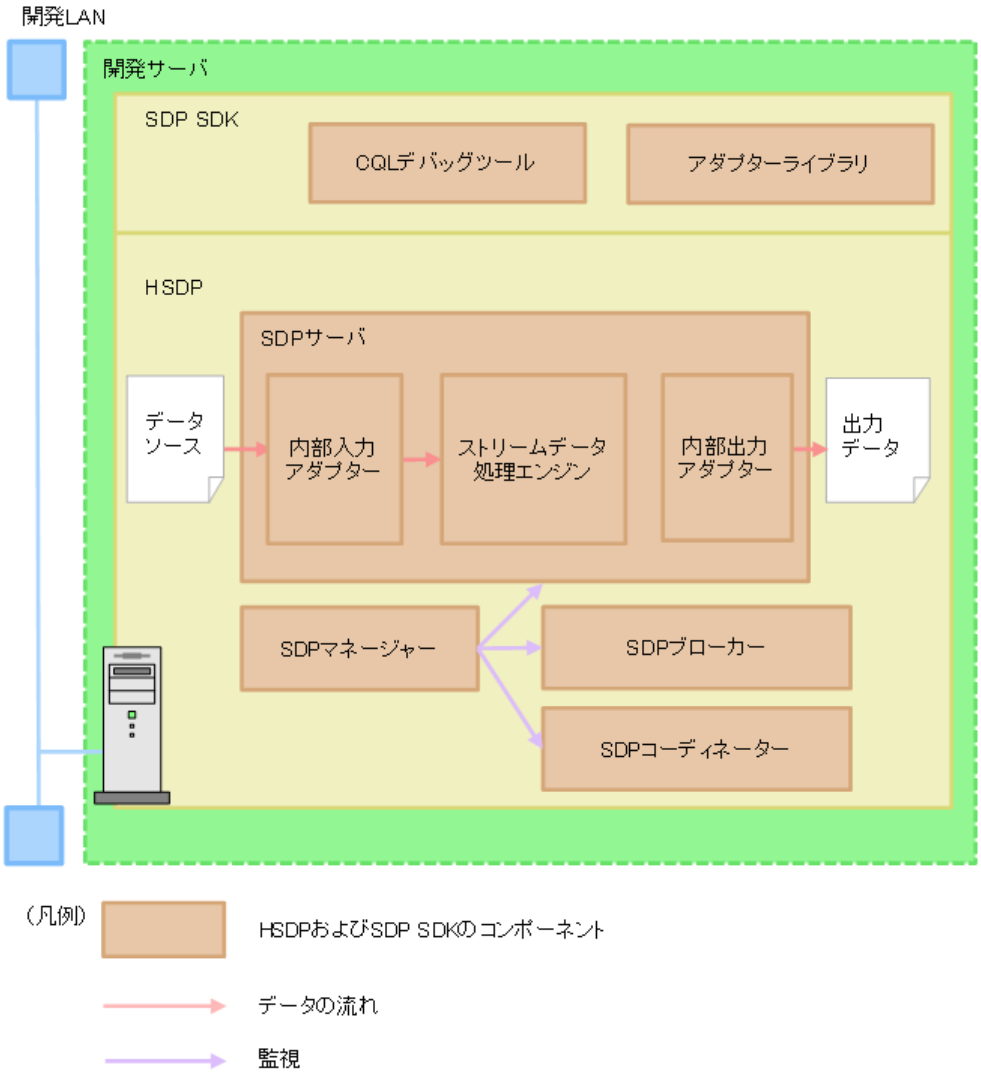
業務システムの HSDP コンポーネント



（凡例）
- ストリームデータの出力･入力宛先情報のデータフロー
- SDPサーバの稼働情報の多重化
- データの流れ

注※　外部アダプターは，HSDPがインストールされていないサーバでもデータを送受信できます。

開発システムの HSDP および SDP SDK コンポーネント



HSDP システムの Streaming Data Platform および Streaming Data Platform software development kit のコンポーネントおよび機能は，次のとおりです。

表 2-2　Streaming Data Platform のコンポーネントと機能

項番	コンポーネント	説明
1	SDP サーバ	• SDP サーバは，ストリームデータを受信，処理，および出力します。 • SDP サーバは，ストリームデータ処理エンジンとストリームデータの入力および出力に使用される内部アダプターで構成されます。
2	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，（CQL を使用して）ユーザーによって定義された分析シナリオに基づいてストリームデータを処理します。
3	内部アダプター	内部アダプターは，SDP サーバとインプロセス連携で動作し，ストリームデータを SDP サーバから入出力するアダプターです。内部アダプターは，内部入力アダプターと内部出力アダプターの 2 つに分類されます。内部入力アダプターと内部出力アダプターの詳細は，それぞれ「7.2　内部入力アダプター」と「7.3　内部出力アダプター」を参照してください。

項番	コンポーネント	説明
4	外部アダプター	外部アダプターは，HSDP が提供する外部アダプターライブラリを利用してユーザーが作成する，任意のデータを HSDP に送信するためのプログラムです。外部アダプターの詳細は「8. 外部アダプター」を参照ください。
5	SDP ブローカー	• SDP ブローカーは，外部アダプター，およびカスケーディングアダプターから問い合わせを受け，それらのアダプターの入出力宛先ストリームのアドレス情報を SDP コーディネーターから取得し，それらのアダプターに送信します。 • 問い合わせしたアダプターは，受け取った入出力宛先ストリームのアドレス情報を基に目的のストリームに接続し，ストリームデータを送受信します。 • 外部アダプター，およびカスケーディングアダプターが，SDP ブローカーを仲介して宛先ストリームに接続する方式を，「ブローカー連携」と呼びます。
6	SDP コーディネーター	• SDP コーディネーターは，入出力宛先（のストリーム）の情報など，SDP サーバの稼働情報を管理します。 • SDP コーディネーターは，他のホストの SDP コーディネーターとクラスターを形成することもできます（コーディネーターグループ）。 • クラスターは，SDP サーバの稼働情報を多重化するのに使用されます。
7	SDP マネージャー	• SDP マネージャーは，SDP サーバ，SDP ブローカー，および SDP コーディネーターを制御します。 • SDP マネージャーは，SDP サーバ，SDP ブローカー，または SDP コーディネーターが機能しなくなると稼働情報に基づいて回復させることができます。
8	HSDP クライアント	HSDP クライアントとは，HSDP クライアントライブラリを利用してユーザーが作成する，ファイルの出力および HSDP 上の TCP データ入力コネクターへの高性能（高速）なデータ送信を実行するためのユーザープログラムです 。

表 2-3 Streaming Data Platform software development kit のコンポーネントと機能

項番	コンポーネント	説明
1	CQL デバッグツール	CQL デバッグツールは，分析シナリオをデバッグします。ユーザーはこのツールを操作して，CQL を使用して開発された分析シナリオをテストします。
2	アダプターライブラリ	アダプターライブラリは，ユーザーが外部アダプター，および内部カスタムアダプターを開発するための API モジュールとヘッダーを提供します。
3	HSDP クライアントライブラリ	HSDP クライアントライブラリは，ユーザーが HSDP クライアントを開発するための API モジュール，ヘッダー，コマンドを提供します。

3 ストリームデータ処理

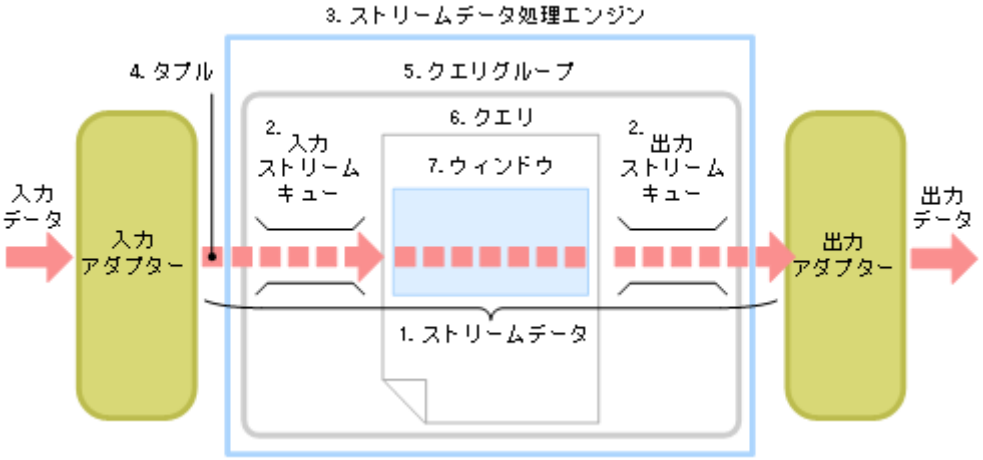
この章では，ストリームデータの処理に使用する，タプル，クエリ，クエリグループ，ウィンドウ，およびストリームデータ処理エンジンについて説明します。

3.1　ストリームデータ処理の概要

ここでは，ストリームデータ処理で使用するソフトウェアコンポーネントについて説明します。

ストリームデータ処理で使用するソフトウェアコンポーネントを次の図に示します。

図 3-1　ストリームデータ処理で使用するソフトウェアコンポーネント



ここでは，図中に示したソフトウェアコンポーネントのうち，次のソフトウェアコンポーネントについて説明します。

1. **ストリームデータ**：連続して発生する多大な時系列順のデータです。
2. **入力ストリームキュー・出力ストリームキュー**：ストリームデータの通路です。
3. **ストリームデータ処理エンジン**：ストリームデータ処理システムで，実際にストリームデータ処理をする部分です。
4. **タプル**：ストリームデータを構成する，値と時刻を併せ持つデータです。
5. **クエリグループ**：ストリームデータ処理での集計・分析シナリオです。業務の目的ごとに作成します。
6. **クエリ**：ストリームデータ処理の内容です。CQL で記述します。
7. **ウィンドウ**：ストリームデータ処理の対象範囲です。ストリームデータのうち，ウィンドウで囲まれた範囲が処理の対象となります。クエリ中に定義します。

3.1.1　ストリームデータ

ストリームデータは，連続して発生する多大な時系列順のデータです。

ストリームデータは，CQL で定義するストリームデータの型（**ストリーム**）に従って流れ，入力ストリームキューを通過して，クエリで処理されます。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。

3.1.2　入力ストリームキュー・出力ストリームキュー

入力ストリームキューは，入力となるストリームデータが通過するための通路です。ストリームを定義するための CQL をクエリ中に記述することで作成されます。

出力ストリームキューは，ストリームデータ処理エンジンでの処理結果（ストリームデータ）を出力するための通路です。出力ストリームキューは，クエリでの処理結果をストリームデータとして出力するための CQL をクエリ中に記述することで作成されます。

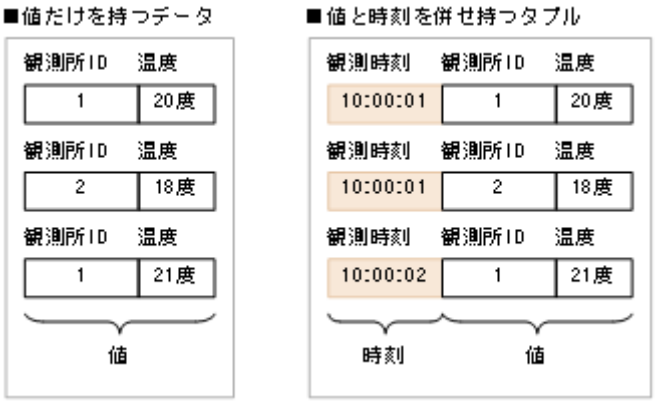
なお，入力ストリームキューを通過するストリームデータの型を**入力ストリーム**といい，出力ストリームキューを通過するストリームデータの型を**出力ストリーム**といいます。

3.1.3　タプル

タプルは，ストリームデータを構成する，値と時刻（**タイムスタンプ**）を併せ持ったデータです。

例として，観測所 1（観測所 ID：1）と観測所 2（観測所 ID：2）でそれぞれ気温を観測した場合について，値だけを持つデータと，値と時刻を併せ持つタプルを比較した図を次に示します。

図 3-2　値だけを持つデータと，値と時刻を併せ持つタプルの比較



この図に示すとおり，観測所ごとの温度情報だけを扱うのではなく，観測時刻というタイムスタンプをタプルに設定することで，ストリームデータ処理の対象にできます。

タプルのタイムスタンプの設定には，タプルがストリームデータ処理エンジンに到着した時刻のタイムスタンプを設定する**サーバモード**と，データの発生した時点の時刻でタイムスタンプを設定する**データソースモード**があります。ログ解析処理など，データソース上の時刻情報の時系列順にストリームデータ処理をしたい場合は，データソースモードを使用してください。

3.1.4　クエリ

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定義したものです。クエリは，**クエリ定義ファイル**内に CQL で記述します。クエリ定義ファイルについては，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

クエリでは，次の 4 種類の演算処理を定義します。

- ストリームデータから分析対象のデータを抽出する**ウィンドウ演算**
- 抽出したデータに処理を実行する**関係演算**
- 処理の結果をストリーム化して出力する**ストリーム化演算**
- ストリームデータを処理して別のストリームデータに変換する**ストリーム間演算**

これらの演算処理の関係を次の図に示します。

図 3-3　クエリでの処理の関係

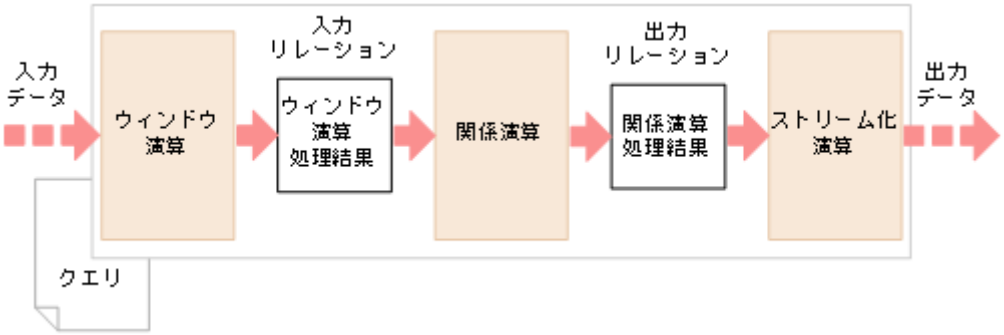
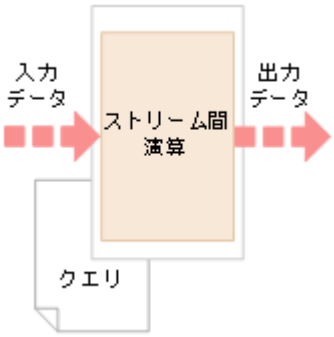


図 3-4　ストリーム間演算



ウィンドウ演算は，ストリームデータからウィンドウで一定の範囲内のデータを抽出するための演算です。このとき生成されるデータ（タプルの集合）を**入力リレーション**といいます。

関係演算は，ウィンドウ演算で抽出されたデータを処理するための演算です。このとき生成されるタプルの集合を**出力リレーション**といいます。

ストリーム化演算は，関係演算で処理されたデータをストリームデータに変換して出力するための演算です。

これに対して，**ストリーム間演算**は，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を実行し，別のストリームデータに変換します。ストリーム間演算では，入出力がストリームデータであること以外は特に規定がなく，入力されたストリームデータに対して実行する処理は任意です。ストリーム間演算関数の処理ロジックを，ユーザーが Java または C 言語で記述して作成するクラスファイルにメソッドとして実装することで，任意の処理を実行できます。

ウィンドウ演算，関係演算，およびストリーム化演算の組み合わせで区間集計（一定区間（時間）の発生データを定期的に集計する）などの処理を実現するのは難しいものでした。今では，区間集計は，ストリーム間演算で実現できます。

なお，ストリーム間演算を使用する場合は，CQL でストリーム間演算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『Hitachi Streaming Data Platform アプリケーション開発ガイド』を参照してください。

それぞれの演算については，「3.2.2　操作系 CQL によるストリームデータの演算処理の指定」を参照してください。

ストリームデータ処理は，ストリームデータ処理エンジンに登録したクエリ定義ファイルの内容に従って実行されます。クエリ定義ファイルの内容については「3.2　CQL を使用したストリームデータ処理の実行」を参照してください。

3.1.5　クエリグループ

クエリグループは，あらかじめユーザーが作成するストリームデータの集計・分析シナリオです。入力ストリームキュー（入力ストリーム），出力ストリームキュー（出力ストリーム），およびクエリで構成されます。

クエリグループは，業務の目的ごとに作成して登録します。また，複数のクエリグループの登録もできます。

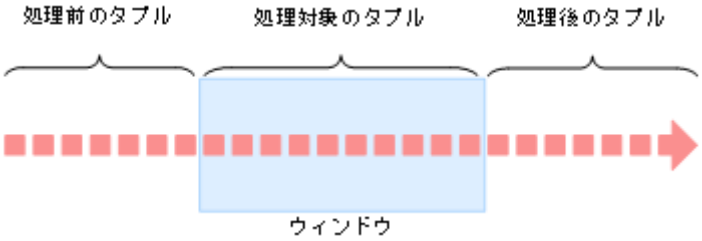
3.1.6　ウィンドウ

ウィンドウは，ストリームデータに対し，集計・分析をするために設定する範囲です。クエリ中に定義します。

データを集計・分析するには，対象とする範囲を明確にする必要があります。ストリームデータの場合も，あらかじめ一定の範囲を決め，範囲内のデータを処理の対象にする必要があります。

ストリームデータとウィンドウの関係を次の図に示します。

図 3-5　ストリームデータとウィンドウの関係



この図に示すウィンドウで囲まれた範囲のストリームデータ（タプル）が，一時的にメモリ上に格納され，処理の対象になります。

ウィンドウでは，時間やタプルの個数などで，処理の対象となるストリームデータの範囲を指定できます。ウィンドウの指定については，「3.2.2　操作系 CQL によるストリームデータの演算処理の指定」を参照してください。

3.1.7　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，ストリームデータ処理を実行する Streaming Data Platform の中心的な要素です。あらかじめ登録したクエリの定義内容に従い，入力アダプターから送信されてきたストリームデータをリアルタイムに処理します。処理結果は，出力アダプターに送られます。

ストリームデータ処理エンジンには，CQL エンジンと acceleration CQL エンジンの 2 種類があります。

通常は CQL エンジンで動作しますが，C 言語の外部定義関数を使うクエリグループの場合は，acceleration CQL エンジンが動作します。CQL エンジンは Java エンジンとも呼ばれます。

CQL エンジンで動作するクエリグループのストリームを Java ストリーム，acceleration CQL エンジンで動作するクエリグループのストリームを C ストリームと呼びます。

3.2 CQL を使用したストリームデータ処理の実行

ストリームデータ処理は，システムに登録したクエリ定義ファイルに従って実行されます。クエリ定義ファイルには，ストリームデータの型であるSTREAM およびクエリを CQL で記述します。これらの CQL による命令を **CQL 文**といいます。

クエリ定義ファイルに記述するために使用する CQL には，次の 2 種類があります。

- 定義系 CQL
  ストリームおよびクエリを定義するために使用する CQL
- 操作系 CQL
  ストリームデータの演算処理で使用する CQL

ここでは，定義系 CQL によるストリームおよびクエリの定義方法，ならびに操作系 CQL によるストリームデータの演算処理の指定方法について説明します。

CQL については，マニュアル『Hitachi Streaming Data Platform アプリケーション開発ガイド』を参照してください。

なお，CQL 文は，あらかじめ意味の与えられているキーワードと，キーワードに続けて指定する項目で構成されます。このキーワードと指定項目の組を**句**といいます。例えば，REGISTER STREAM *ストリームの名称*のように，REGISTER STREAM というキーワードと*ストリームの名称*という指定項目を合わせたものを REGISTER STREAM 句といいます。

3.2.1 定義系 CQL によるストリームおよびクエリの定義

ストリームおよびクエリの定義に使用する CQL を**定義系 CQL** といいます。定義系 CQL には，次の 3 種類があります。

- REGISTER STREAM 句
- REGISTER QUERY 句
- REGISTER QUERY_ATTRIBUTE 句

3.2.2 操作系 CQL によるストリームデータの演算処理の指定

操作系 CQL には，次の 4 種類があります。

- ウィンドウ演算
- 関係演算
- ストリーム化演算

- ストリーム間演算

3.2.3 外部定義関数

外部定義関数には，Java の外部定義関数と C 言語の外部定義関数があります。

- Java の外部定義関数

  外部定義関数の処理ロジックを Java で実装します。Java で実装した外部定義関数は，CQL エンジンで処理されます。利用できる Java の外部定義関数は，外部定義集合関数，外部定義スカラ関数，および外部定義ストリーム間演算関数です。

- C 言語の外部定義関数

  外部定義関数の処理ロジックを C 言語で実装します。C 言語で実装した外部定義関数は，acceleration CQL エンジンで処理されます。利用できる C 言語の外部定義関数は，外部定義ストリーム間演算関数です。

  C 言語の外部定義関数を開発するには，C 言語の外部定義関数のライブラリによって提供されるヘッダーを含める必要があります。C 言語の外部定義関数のライブラリでは，構造体と関数が提供されます。

4 SDP サーバ

この章では，SDP サーバの概要について説明します。

4.1 SDP サーバの概要

SDP サーバとは，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です。

SDP サーバにクエリグループを登録することで（複数登録可能），クエリグループに記載されたシナリオに従い，ストリームデータを集計・分析します。

運用ディレクトリ 1 つにつき，SDP サーバは 1 つ起動できます。複数の SDP サーバを起動する場合は，運用ディレクトリを複数作成します。

5 SDP ブローカーと SDP コーディネーター

この章では，SDP ブローカーと SDP コーディネーターについて説明します。

5.1 SDP ブローカーと SDP コーディネーターの概要

SDP ブローカーと SDP コーディネーターは，外部アダプターおよびカスケーディングアダプターが，データ送信ストリームまたは受信宛先ストリーム（入力ストリームまたは出力ストリーム）に接続するために使用する機能を提供します。

SDP コーディネーター

SDP コーディネーターは，接続できるストリームが登録されている SDP サーバの稼働情報（該当する SDP サーバが稼働しているホストのホスト名，該当する SDP サーバに登録されているストリーム名など）を管理します。

SDP ブローカー

SDP ブローカーは，接続先ストリームの場所を特定し，接続して，外部アダプターおよびカスケーディングアダプターに情報を渡すために必要となる情報を（SDP コーディネーターから）探すための機能を提供します。

最大で 1 つずつの SDP ブローカーと SDP コーディネーターを 1 つのホスト上で実行できます。

SDP ブローカーのプロセスを開始するかどうかを，SDP マネージャー定義ファイルの`hsdp_broker` プロパティで指定します。また，自ホストで SDP コーディネーターのプロセスを開始するかどうかを，SDP マネージャー定義ファイルの`hsdp_coordinator` プロパティで指定します。SDP マネージャー定義ファイルの`hsdp_broker` プロパティと`hsdp_coordinator` プロパティに`true` を指定すると，`hsdpmanager -start` コマンド実行時に SDP ブローカー，および SDP コーディネーターが起動して，自ホストを含めたコーディネーターグループを作成できます。

メモ

SDP マネージャー定義ファイルの`hsdp_broker` プロパティと`hsdp_coordinator` プロパティには，同じ値（`true` または`false` のどちらか）を指定してください。異なる値を指定した場合，SDP ブローカーと SDP コーディネーターは正常に動作しないおそれがあります。

SDP サーバ単独で動作する機能を使用したい場合や，ソリューションデベロッパーが分析シナリオ（CQL）を開発，検証したい場合などに，`hsdp_broker` プロパティと`hsdp_coordinator` プロパティに`false` を指定します。`false` を指定した場合は，次の動作となります。

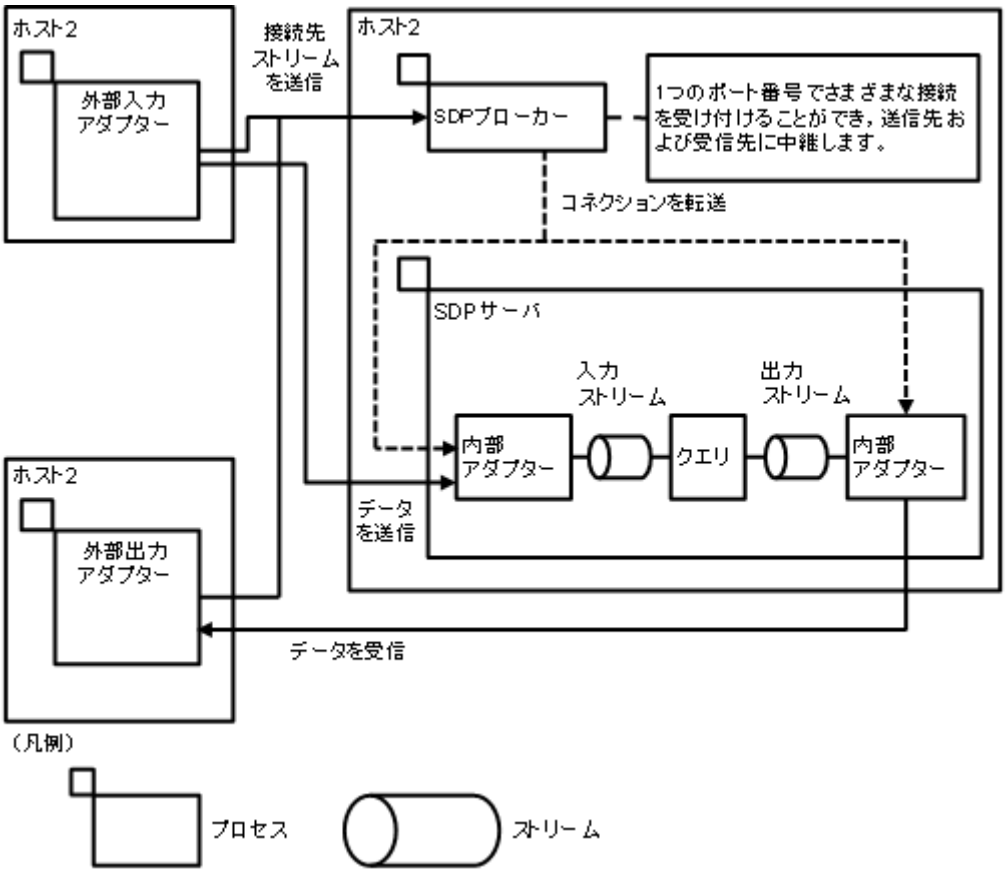
- `hsdpmanager -start` コマンド実行時に SDP ブローカーは起動しません。そのため，外部アダプターは自ホストのブローカーに接続できません。必ず他ホストで起動している SDP ブローカーを指定してください。
- コーディネーターグループに自ホストを指定しているかどうかに関係なく，`hsdpmanager -start` コマンド実行時に SDP コーディネーターを起動しません。そのため，自ホストを含めたコーディネーターグループを作成できません。

- 自ホストの SDP サーバにクエリグループを登録した場合も，SDP コーディネーターには情報が登録されません。そのため，他ホストから自ホストへのストリーム情報は検索されません。

TCP ポートを統合する

SDP ブローカーには，ローカルホストのストリームにデータを送受信する内部アダプターに，自身と外部アダプターやカスケーディングアダプターとの TCP 通信によるコネクションを転送する機能があります。

この機能を使用することで，SDP ブローカーは，外部アダプターまたはカスケーディングアダプターと，内部アダプターの間でのコネクションを中継し，1 つのポート番号を使用してホスト上のさまざまなストリームへの接続を受け付けられます。

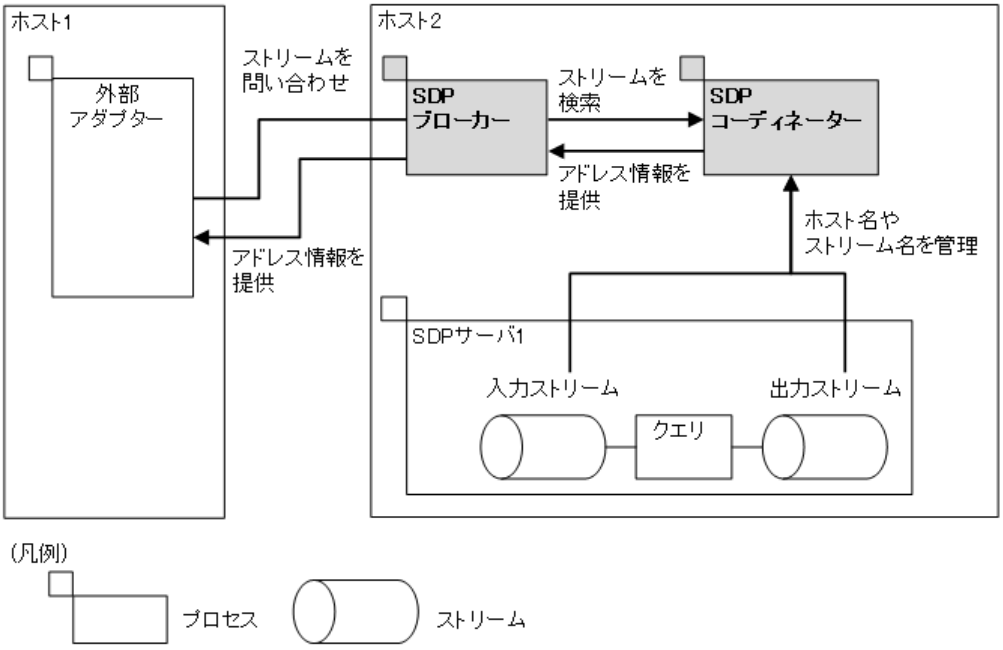


5.2 ストリーム検出機能

SDP コーディネーターは，接続できるストリームが登録されている SDP サーバの稼働情報（該当する SDP サーバが稼働しているホストのホスト名，該当する SDP サーバに登録されているストリーム名など）を管理します。SDP ブローカーは，外部アダプター，およびカスケーディングアダプターから問い合わせを受けると，SDP コーディネーターを利用して接続先ストリームの場所を特定し，問い合わせ元アダプターと接続先ストリームの接続を仲介する機能を提供します。

ストリームを検出する

外部アダプター，およびカスケーディングアダプターが SDP ブローカーを利用してストリームに接続する場合，SDP ブローカーへの問い合わせに必要な情報は，目的のストリームのサーバクラスター名，クエリグループ名，ストリーム名です。これらの情報を基に SDP ブローカーと SDP コーディネーターがそのストリームの実際のアドレスを解決するため，そのストリームが別の SDP サーバに再登録された場合でも，オペレーターは外部アダプター，およびカスケーディングアダプターの設定を変更しないで，同じ設定のままそれらのアダプターの実行を続けることができます。



SDP ブローカー

SDP ブローカーは，外部アダプター，およびカスケーディングアダプターから問い合わせを受けると，SDP コーディネーターを利用して接続先ストリームの場所を特定し，問い合わせ元アダプターにアドレス情報を提供します。このアドレス情報を基に，外部アダプター，およびカスケーディングアダプターは別の SDP サーバ，および外部アダプターに接続してストリームデータを送受信します。

SDP コーディネーター

SDP コーディネーターは，ホスト名や登録されているストリーム名など，SDP サーバの稼働情報を管理します。SDP コーディネーターは，ほかのホストの SDP コーディネーターとクラスター（コーディネーターグループ）を形成し，SDP サーバの稼働情報を多重化することもできます。

SDP コーディネーターで管理される情報

SDP ブローカーは，SDP サーバに登録されているクエリグループを開始すると同時に，次の表に示す情報を SDP コーディネーターに登録します。SDP サーバからクエリグループが削除されると，SDP ブローカーは削除したクエリグループに対応する登録情報を SDP コーディネーターから削除します。コーディネーターグループが設定されている場合，情報が登録されるか削除されると，現在の登録情報が，コーディネーターグループのすべての SDP コーディネーターに直ちに共有されます。

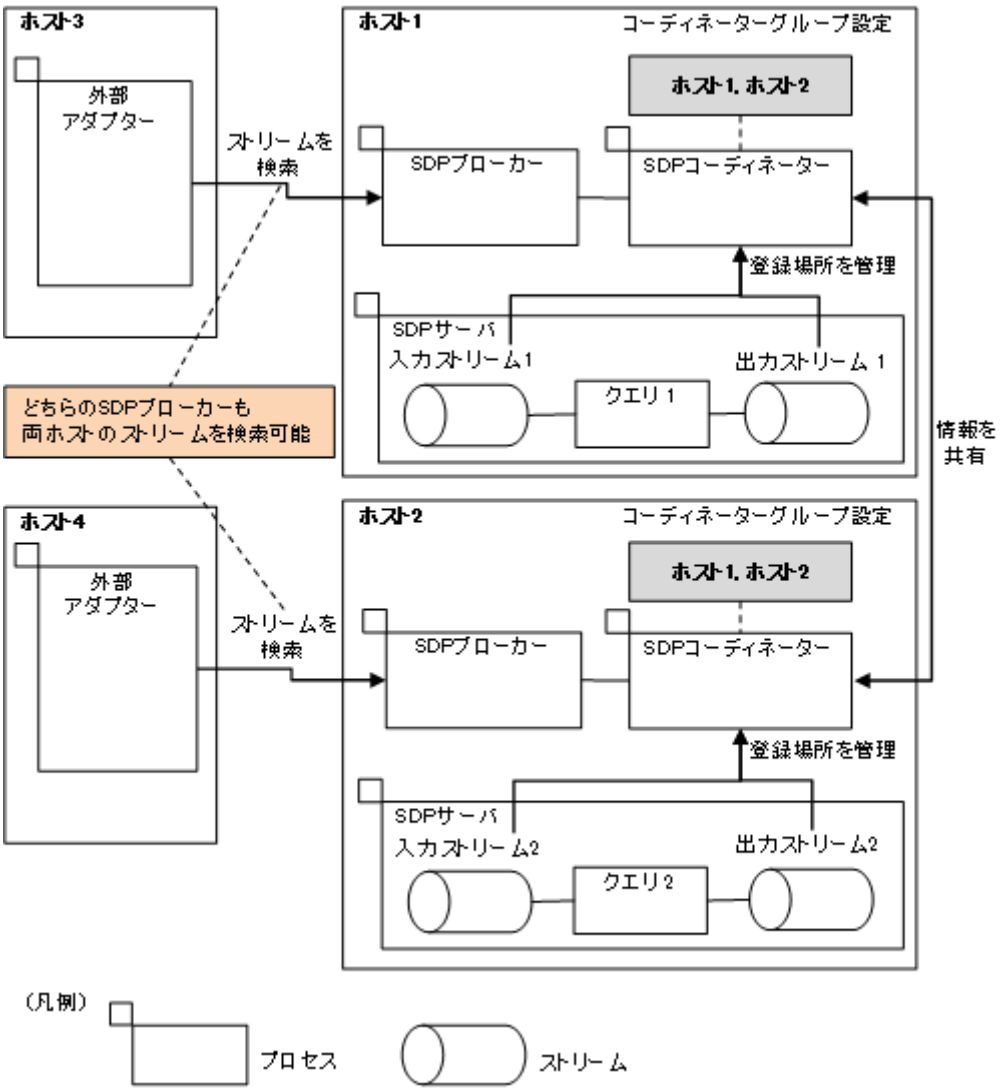
表 5-1　SDP コーディネーターで管理される情報

No.	項目	説明
1	ホスト	接続先ストリームが登録される HSDP システムのホスト名または IP アドレス
2	HSDP 運用ディレクトリ	接続先ストリームが登録される SDP サーバの運用ディレクトリの絶対パス
3	サーバクラスター名	サーバが属するサーバクラスターの名前
4	サーバ名	SDP サーバの名前
5	クエリグループ名	接続先ストリームが定義されるクエリグループの名前
6	文字コード情報	接続先ストリームが文字列を処理する際の文字コード情報
7	ストリーム名	接続先ストリームの名前
8	TCP 接続ポート	TCP ポート
9	RMI 通信ポート	RMI ポート
10	ストリームタイプ	入力／出力
11	タイムスタンプモード	接続先ストリームのタイムスタンプモード
12	負荷分散方式（ディスパッチタイプ）	接続先ストリームへのデータの負荷分散方式を記述するプロパティ情報
13	スキーマ情報	接続先ストリームのスキーマ情報

5.3 コーディネーターグループ

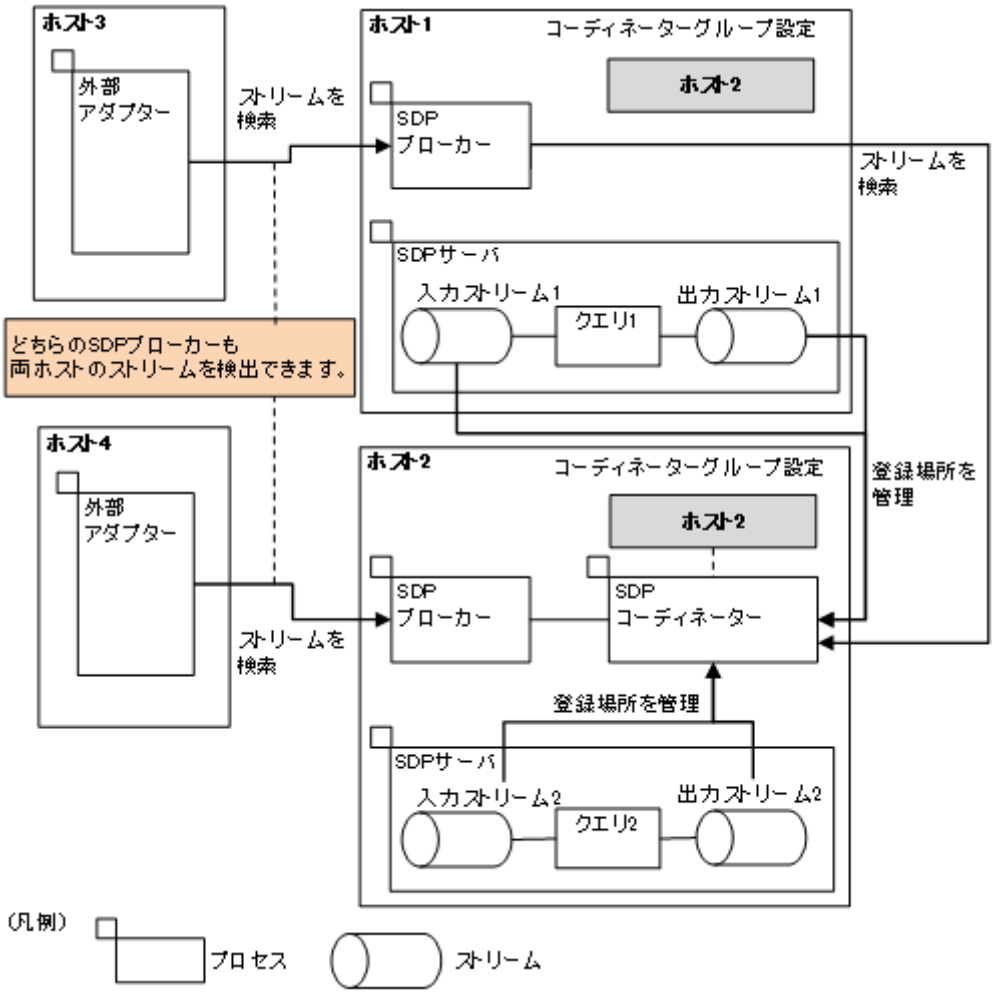
コーディネーターグループとは，SDP サーバの稼働情報を共有する SDP コーディネーターの集合です。SDP コーディネーターは，複数のホスト間でクエリグループとストリームの接続先情報を共有できます。このような情報を共有する SDP コーディネーターは，hsdpsetup コマンドの-chosts オプションを使用して，コーディネーターグループに設定されます。詳細については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

SDP ブローカーは，コーディネーターグループによって共有されるデータを使用して，同じコーディネーターグループを使用するすべてのホスト上のストリームのアドレスを取得できます。外部アダプターとカスケーディングアダプターは，任意のホスト上の SDP ブローカーに問い合わせることで，複数のホストにあるストリームのアドレスを解決し，接続できます。さらに，同じコーディネーターグループを使用するほかのホストの SDP ブローカーを接続先として設定すると，同じストリームに接続できます。



ローカルホストを含まないコーディネーターグループ

ローカルホストを含まないコーディネーターグループを設定できます。この場合，ローカルホストの SDP コーディネーターは起動しません。SDP ブローカーは別のホストの SDP コーディネーターに対して，ローカルホストの情報の取得および保存を実施します。同じコーディネーターグループを使用するすべてのホストのストリームを検索できます。また，SDP ブローカーが別のホストの SDP コーディネーターを使用する場合，最大 1,024 の SDP ブローカー（参照先 SDP コーディネーターのホスト上にある SDP ブローカーを含む）が，情報参照先のコーディネーターグループに接続できます。



情報の多重化

hsdpsetup コマンドの-cmulti オプションを使用して，データの多重度を設定できます。

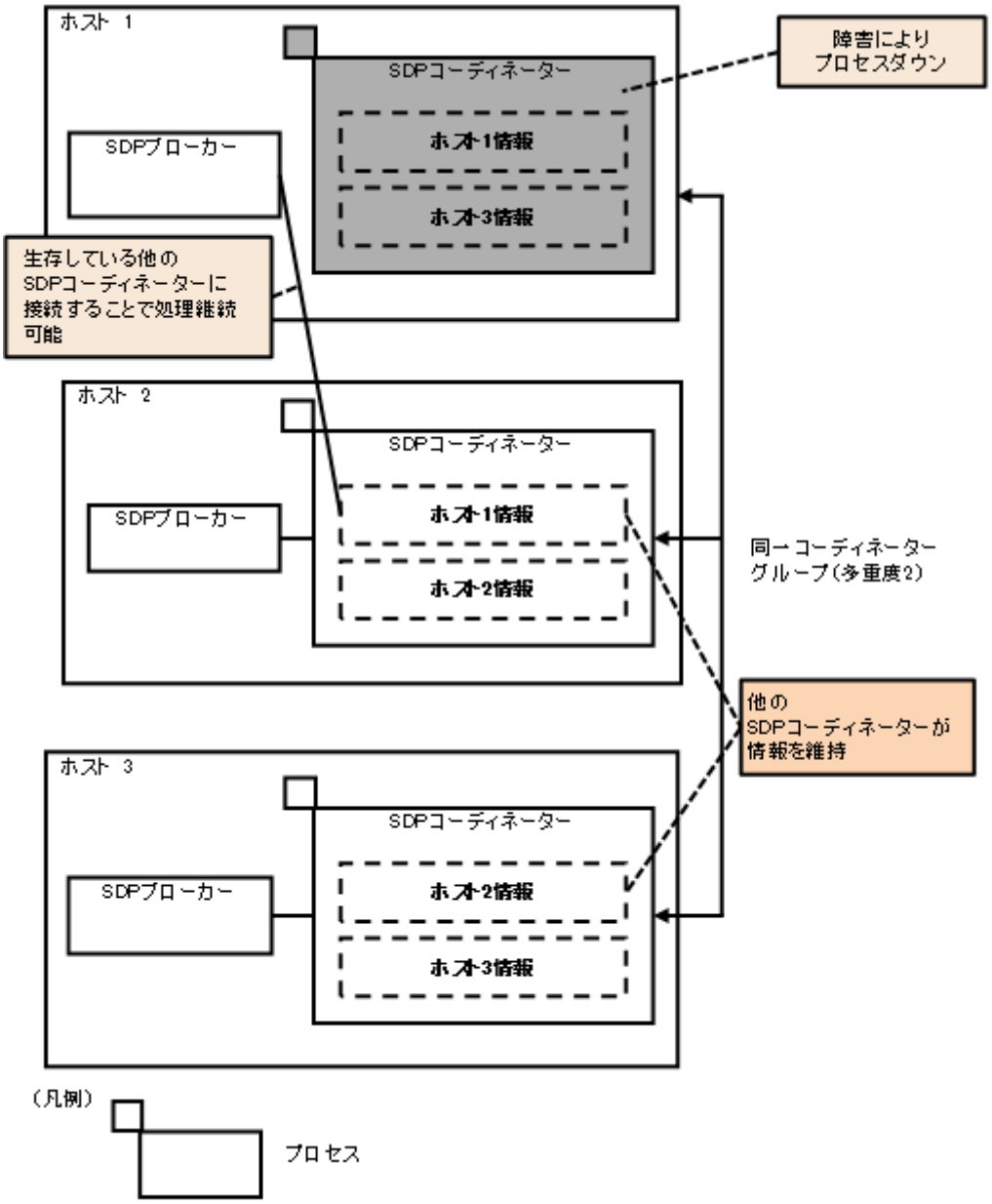
3 つ以上の SDP コーディネーターから成るコーディネーターグループを構成する場合，複数の SDP コーディネーターで同じ情報を重複して保存できます。データの多重度は，hsdpsetup コマンドの-cmulti オプションを使用して設定してください。詳細については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

データの多重度が 2 以上で設定されている場合に，多重度より少ない数の SDP コーディネーターがコーディネーターグループ内で機能しなくなっても，別のホストの SDP コーディネーターを使用して稼働を続

けることができます。機能しなくなった SDP コーディネーターの数が設定された多重度と同じか，多い場合は，すべての SDP コーディネーターを再起動する必要があります。また，SDP コーディネーターが機能しなくなる前にクエリグループが起動され，実行していた場合，クエリグループも再起動する必要があります。

コーディネーターグループ内で 2 つの SDP コーディネーターが実行されていた場合，停止した SDP コーディネーターを再起動することで，コーディネーターグループを元の状態に戻せます。

詳細については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。



6 SDP マネージャー

この章では，SDP サーバ，SDP ブローカー，および SDP コーディネーターを制御する SDP マネージャーについて説明します。

6.1 SDP マネージャーの概要

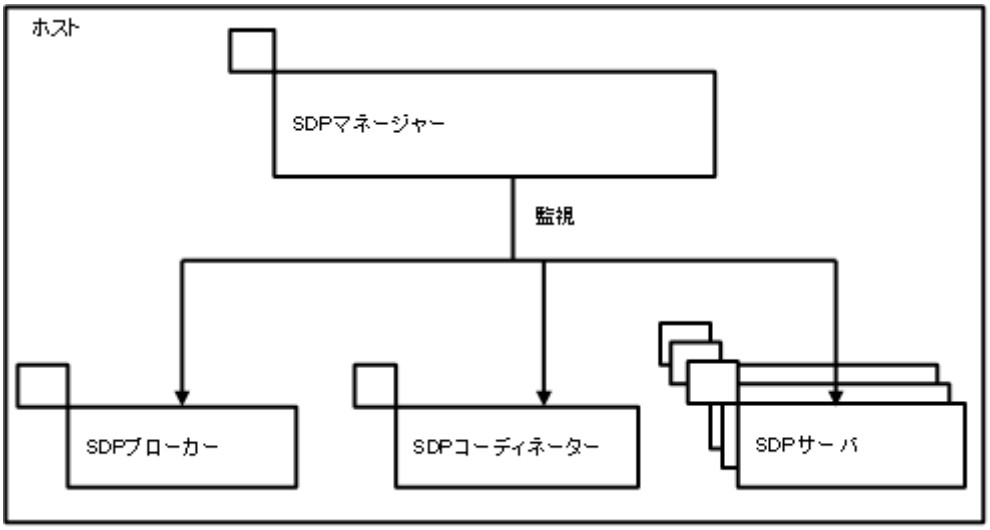
SDP マネージャーは，SDP サーバ，SDP ブローカー，および SDP コーディネーターを制御します。

SDP サーバが機能しなくなると，SDP マネージャーは，SDP コーディネーターが保持する（SDP サーバの）稼働情報に基づいて，その SDP サーバを回復します。また，SDP ブローカー，SDP コーディネーターが異常終了すると，SDP マネージャーはそれらを再起動します。

6.2 SDP マネージャーのログ通知

SDP マネージャーのログ通知機能は，ホストで利用できるさまざまなコンポーネントのプロセスを監視するために使用されます。プロセスダウンを検知すると，ログ通知機能によって，ログファイルにメッセージが出力されます。

プロセス監視



ログ通知機能は，同一ホスト上の次のコンポーネントのプロセスを監視します。

- SDP ブローカー
- SDP コーディネーター
- SDP サーバ

最大で 1 つの SDP マネージャーを 1 つのホスト上で実行できます。なお，SDP マネージャーを起動するかどうかを，SDP マネージャー定義ファイルの`hsdp_manager` プロパティで指定できます。

SDP ブローカー，SDP コーディネーター，および SDP サーバのコンポーネントのプロセスを`hsdpmanager` または`hsdpstart` コマンドを実行して起動できます。

各プロセスを起動すると，各プロセスが稼働中になり，SDP マネージャーはこれらのプロセスの監視を開始します。監視時に，エラーが発生しプロセスが異常終了すると，SDP マネージャーはその異常終了を検知し，SDP マネージャーのログファイルにメッセージを出力します。メッセージはエラーとその後発生した異常終了の詳細で構成されます。SDP マネージャーのログファイルの詳細については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

`hsdpmanager` コマンドを実行した際に SDP マネージャーの起動に失敗した場合，どのコンポーネントのプロセスも監視されません。また，`hsdpmanager` コマンド，`hsdpstart` コマンド，または`hsdpcql` コマンドを実行した際にコンポーネントの起動に失敗した場合，そのコンポーネントは監視されません。

SDP マネージャーは，次の場合に，SDP マネージャーログファイルにメッセージを出力します。

表 6-1　SDP マネージャーログファイルに出力されるメッセージ

状況	メッセージ ID
SDP マネージャーがプロセスの監視を開始しました。	KFHD10000-I
SDP マネージャーがプロセスの停止を検知しました。	KFHD10002-E
SDP マネージャーがプロセスの監視を停止しました。	KFHD10001-I

6.3 SDP マネージャーの再起動機能

SDP マネージャーの再起動機能は，ログ通知に表示される各コンポーネントのプロセスを監視し，異常終了したプロセスを再起動する機能を提供します。

SDP サーバが再起動されると，（SDP サーバがシャットダウンする前に実行していた）クエリグループと内部アダプターも再起動されます。また，SDP マネージャーはシャットダウンされた SDP マネージャー自体のプロセスも再起動させます。再起動機能は，SDP マネージャー定義ファイルの`hsdp_restart` プロパティで有効または無効にできます。SDP マネージャー定義ファイルの詳細については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。SDP マネージャー定義ファイルの`hsdp_cpu_no_list` プロパティに指定された CPU は，再起動されたコンポーネントのプロセスに割り当てられます。

次のどれかの条件を満たしている場合，SDP マネージャーは，特定のコンポーネントのプロセスを再起動しません。

- `hsdpmanager` コマンドを実行した際に SDP マネージャーの起動に失敗した場合，どのコンポーネントのプロセスも（SDP マネージャー自体のプロセスも含め）再起動しません。
- `hsdpmanager` コマンド，`hsdpstart` コマンド，または`hsdpcql` コマンドを実行した際にコンポーネントが起動に失敗した場合，SDP マネージャーはそのコンポーネントのプロセスを再起動しません。
- 再起動設定が無効の場合，SDP マネージャーはログ通知に表示されるどのプロセスも（SDP マネージャー自体のプロセスも含め）再起動しません。
- SDP マネージャー定義ファイルで`hsdp_manager` プロパティに`false` が指定されている場合は，どのコンポーネントのプロセスも（SDP マネージャー自体のプロセスも含め）再起動しません。
- SDP マネージャー定義ファイルで，指定したコンポーネントを使用するかどうかの設定（`hsdp_broker` プロパティ，`hsdp_coordinator` プロパティ）に`false` が指定されている場合は，SDP マネージャーはそのコンポーネントのプロセスを再起動しません。
- 利用している OS の種類によっては，再起動が有効に設定されていても，SDP マネージャーはどのコンポーネントも再起動しません。

表 6-2 SDP マネージャーの再起動機能の利用可否

前提条件となる OS	バージョン	利用可能かどうか
Red Hat Enterprise Linux Server	6.5 以降	はい
	7.1 以降	はい

メモ

再起動機能を利用できない場合にプロセスが異常終了したときは，ユーザーは`hsdpmanager` コマンドを使用して，手動で SDP マネージャーのプロセスを再起動する必要があります。

SDP マネージャーは，各コンポーネントの再起動中にプロセス間の通信障害が発生した場合，再起動要求をリトライします。リトライの回数と間隔は，SDP マネージャー定義ファイルの，hsdp_retry_times プロパティおよびhsdp_retry_interval プロパティで指定できます。SDP マネージャー定義ファイルの詳細についてはマニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。指定したリトライ回数に達しても通信障害が発生する場合，再起動処理が失敗します。その後，その他の要因も含めて，SDP マネージャーによる監視時にコンポーネントがまだダウンしている場合，SDP マネージャーは再びコンポーネントの再起動処理をします。SDP マネージャーは，SDP マネージャー定義ファイルのhsdp_restart_watch_time プロパティに指定した時間内に 3 回コンポーネントのダウンを検知した場合，コンポーネントの監視を停止し，以降の再起動処理をしません。

SDP コーディネーターが次の両方の条件を満たす場合，SDP マネージャーによって SDP コーディネーターは再起動されません。

- コーディネーターグループが 3 つ以上の SDP コーディネーターから成る場合
- 指定した多重度より多い数または同じ数の SDP コーディネーターが停止した場合

SDP コーディネーターを再起動できない場合は，コーディネーターグループ内で実行しているすべての SDP コーディネーターをhsdpmanager -stop コマンドを使用して停止してください。すべての SDP コーディネーターを停止後に，hsdpmanager -start コマンドを実行して手動で再起動する必要があります。この場合，クエリグループが実行していると，SDP コーディネーターに登録されているストリーム情報は失われます。このため，クエリグループを再起動する必要があります。

SDP マネージャーは，次の場合に，SDP マネージャーログファイルにメッセージを出力します。

表 6-3　SDP マネージャーログファイルに出力されるメッセージ

状況	メッセージ ID
SDP マネージャーがプロセスの停止を検知しました。	KFHD10002-E
SDP マネージャーが自動再起動を開始しました。	KFHD10017-I
SDP マネージャーが自動再起動を正常に実行しました。	KFHD10000-I KFHD10018-I
SDP マネージャーが自動再起動の実行に失敗しました。	KFHD10003-E
SDP マネージャーがhsdp_restart_watch_time で指定した時間内に 3 回プロセスの停止を検知し，プロセスの監視を停止しました。	KFHD10001-I

7 内部アダプター

この章では，内部アダプターについて説明します。

7.1 内部アダプターの概要

内部アダプターとは，SDP サーバと同一プロセスで動作して，ストリームデータを入出力するアダプターです。

SDP サーバと同一プロセスとして動作するプロセスの動作形式を，インプロセス連携と呼びます。そのため，内部アダプターはインプロセス連携アダプターとも呼ばれます。

入力用の内部アダプターを内部入力アダプター，出力用の内部アダプターを内部出力アダプターと呼びます。

また，それぞれのアダプターには HSDP で標準で提供しているアダプターがあり，それを内部標準アダプターと呼びます。

一方，ユーザーが Streaming Data Platform software development kit の API を使用して開発する内部アダプターは，内部カスタムアダプターと呼びます。

内部標準アダプターでは，コールバックと呼ばれる単位で，アダプター内で実行する処理を定義します。

コールバックは，アダプター構成定義ファイルによって定義します。定義方法の詳細については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

内部入力アダプターが実行する処理（コールバック）

- データ入力（入力対象の形式は，TCP データ，CSV ファイル，HTTP パケットなど）
- データ編集（マッピング，レコードの抽出，フォーマット変換など）
- タプル送信（ストリームデータ処理エンジンにデータを出力）

内部出力アダプターが実行する処理（コールバック）

- タプル受信（ストリームデータ処理エンジンからタプルを受信）
- データ編集（マッピング，フィルタリング，フォーマット変換など）
- データ出力（出力対象の形式は，CSV ファイル，ダッシュボード，TCP データなど）

7.2　内部入力アダプター

内部入力アダプターは，ストリームデータを特定の形式で受信し，ストリームデータ処理エンジンにデータを送信します。

内部標準入力アダプターとしてサポートされる形式は，次のとおりです。

- テキストファイル
- HTTP パケット
- TCP データ

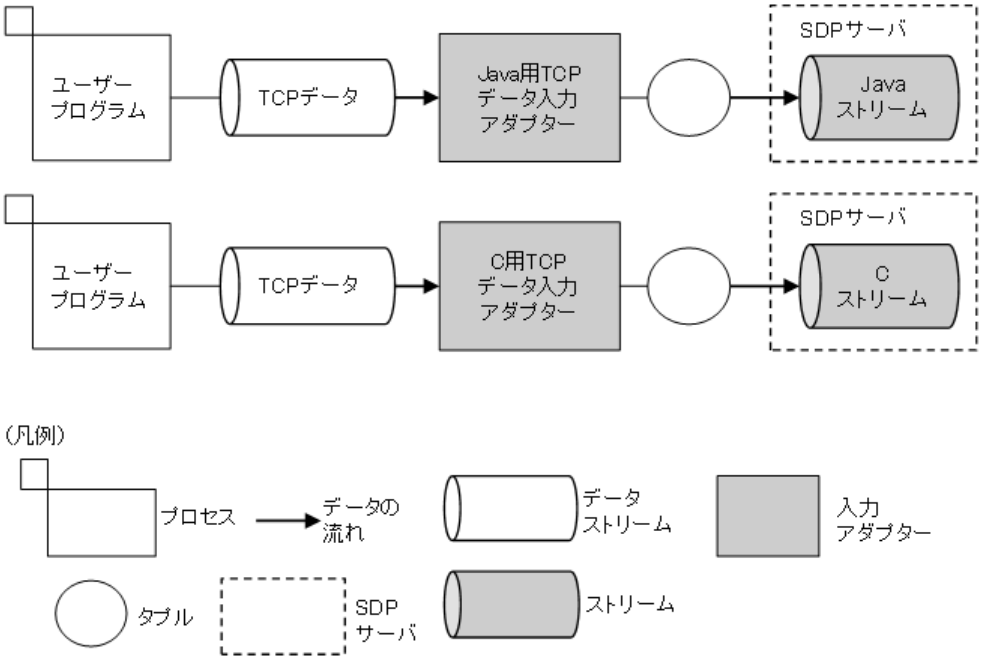
7.2.1　TCP データ入力アダプター

ストリームデータ処理の対象が，TCP データである場合，入力用コールバックとして，TCP データ入力コネクターを使用します。

入力用コールバックに TCP データ入力コネクターを定義したアダプターを，TCP データ入力アダプターと呼びます。

TCP データ入力アダプターは，任意の TCP クライアントを持つユーザープログラム，または外部入力アダプター，HSDP クライアント，カスケーディングアダプターと TCP 接続を確立し，データを受信します。また，受信した TCP データをタプルに変換し，タプルを SDP サーバに送信します。

図 7-1　TCP データの受信とタプルの送信



TCP データ入力アダプター：SDP サーバの Java ストリームまたは C ストリームにタプルを送信します。

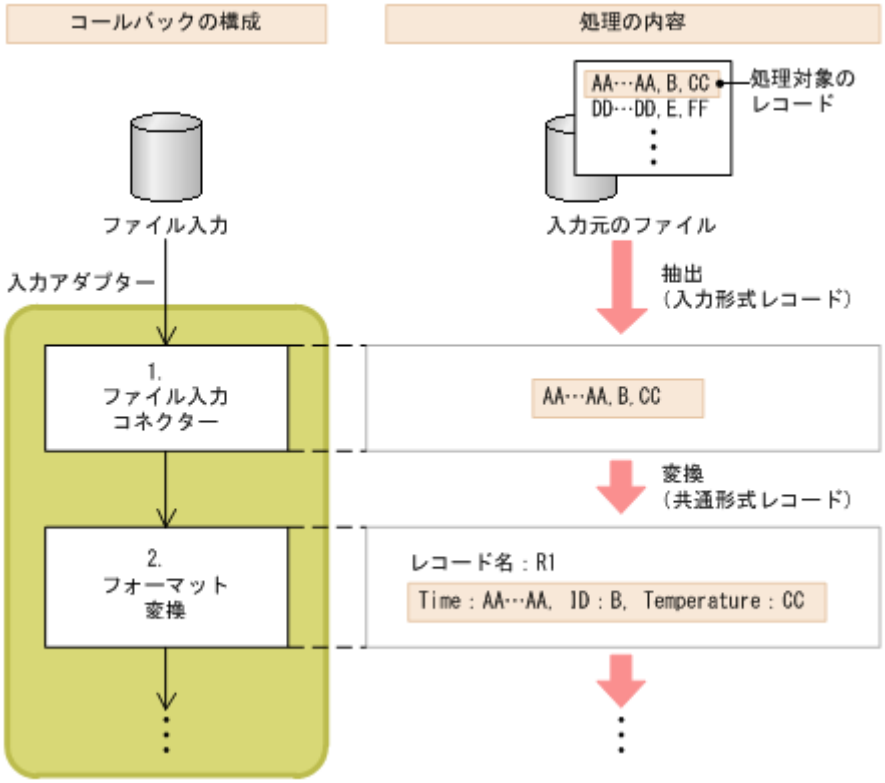
7.2.2　ファイル入力アダプター

ストリームデータ処理の対象が，ログファイルなどファイル形式のデータである場合，入力用コールバックとして，**ファイル入力コネクター**を使用します。

入力用コールバックにファイル入力コネクターを定義したアダプターを，ファイル入力アダプターと呼びます。

ファイル入力コネクターは，入力元のファイルから，処理の対象となるレコードを抽出します。このレコードは，入力形式レコードとして抽出されるため，フォーマット変換で共通形式レコードに変換し，ストリームデータ処理エンジンで扱える形式にする必要があります。ファイル入力でのコールバックの構成と処理の内容を次の図に示します。

図 7-2　ファイルの入力でのコールバックの構成と処理の内容



1. ファイル入力コネクターで，入力されたファイルの 1 行目（レコード）を抽出します。
   このとき，抽出されるレコードは，入力形式レコードです。
2. フォーマット変換をして，入力形式レコードを共通形式レコードに変換します。

> **ヒント**
>
> 入力元のファイルからは，処理の対象となるレコードを一度に複数行抽出することもできます。

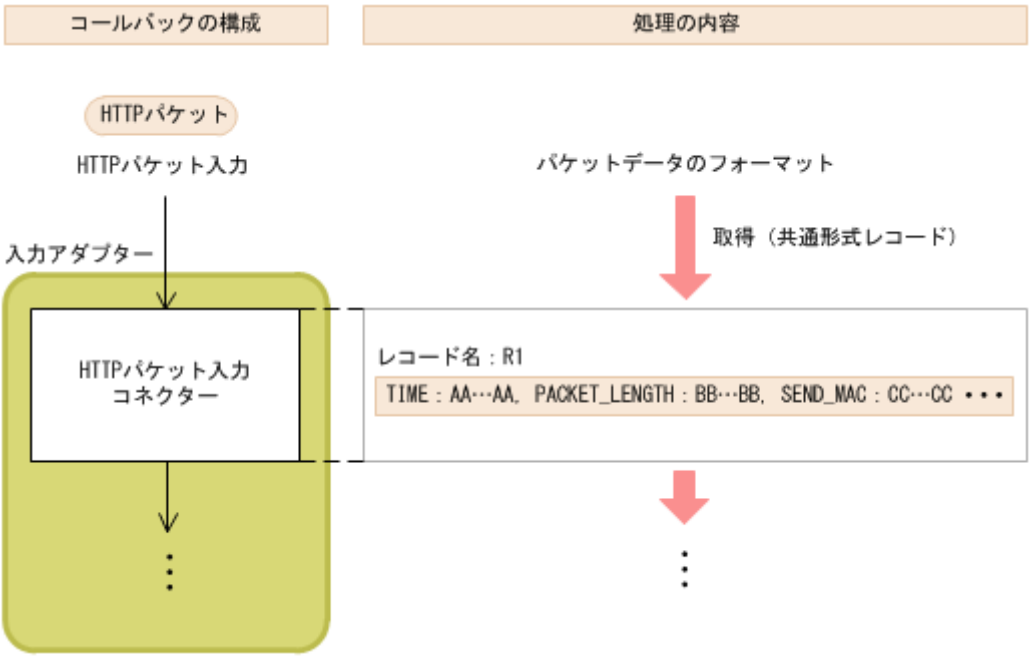
7.2.3　HTTP パケット入力アダプター

ストリームデータ処理の対象が，ネットワーク上の HTTP パケットである場合，入力用コールバックとして，**HTTP パケット入力コネクター**を使用します。

入力用コールバックにファイル入力コネクターを定義したアダプターを，HTTP パケット入力アダプターと呼びます。

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 パケットを取得します。HTTP パケットの入力でのコールバックの構成と処理の内容を次の図に示します。

図 7-3　HTTP パケットの入力でのコールバックの構成と処理の内容



この図に示すとおり，HTTP パケット入力コネクターが取得した HTTP パケットは，HTTP パケット入力コネクター内で共通形式レコードに変換され，ストリームデータ処理システムで扱えるデータ形式になります。

7.3 内部出力アダプター

内部出力アダプターはストリームデータ処理エンジンから処理されたストリームデータを受信し，データを特定の形式で出力します。

内部標準出力アダプターとしてサポートされる形式は，次のとおりです。

- テキストファイル
- TCP（カスケーディングアダプター）
- ダッシュボード

複数の内部出力アダプターを使用して，同一の出力ストリームに接続できます。その場合，出力ストリームに出力されたストリームデータは，接続された複数の内部出力アダプターすべてに複製され，それぞれのアダプターから出力されます。

7.3.1 カスケーディングアダプター

Streaming Data Platform では，内部標準アダプターの 1 つにカスケーディングアダプターを提供しています。カスケーディングアダプターは，クエリグループ間を接続するための TCP データ入力アダプター，または外部出力アダプター宛てに，TCP ソケットを介してデータを送信します。

カスケーディングアダプターは，SDP ブローカー連携によって自動的に生成・起動することができます。

クエリグループ間のデータ送信プロセスに，内部カスタムアダプターは使用できません。

カスケーディングアダプターがデータを受け取るストリーム（出力ストリーム）が Java ストリームの場合，Java 用カスケーディングアダプターを使用し，C ストリームの場合は，C 用カスケーディングアダプターを使用します。Java/C の指定は，アダプター構成定義ファイルの`OutputAdaptorDefinition` タグの `language` パラメータで行います。

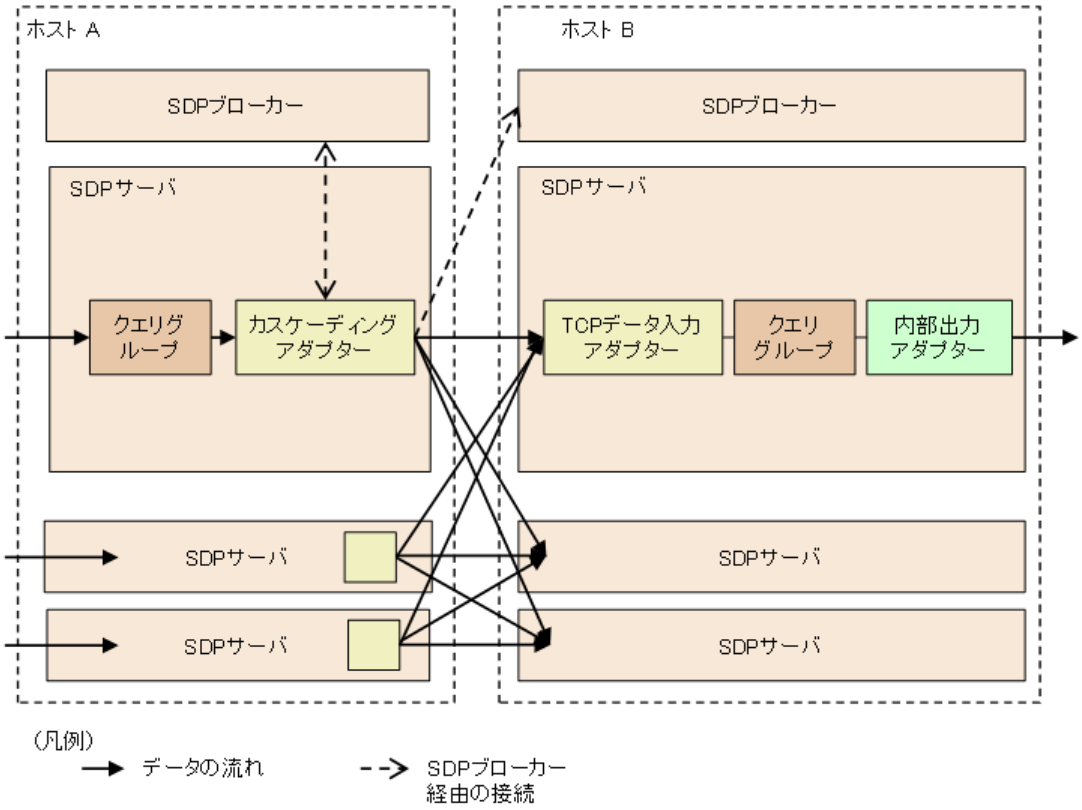
(1) クエリグループへのデータ送信

ほかのクエリグループのストリームへデータを送信する場合，カスケーディングアダプターと TCP データ入力アダプターの接続が必要になります。

次の図は，ホスト A とホスト B 内のクエリグループを接続するカスケーディングアダプターの構成を示しています。カスケーディングアダプターは，ホスト A の出力ストリームからホスト B の入力ストリームに，TCP データ入力アダプターを介してデータを送信します。接続された入力および出力ストリームについての情報は，送信元のクエリグループ用プロパティファイルで定義されます。カスケーディングアダプターは，対応するクエリグループ（この場合，ホスト A のクエリグループ）の起動と同時に自動起動します。起動後，カスケーディングアダプターは，ホスト A の SDP ブローカーに接続するストリームのアドレスを問い合わせて取得し，その後ホスト B の SDP ブローカーを介して接続を確立します。

なお，1 つの SDP サーバに複数のクエリグループを登録し，そのクエリグループ間でデータを送信する場合も，同様にカスケーディングアダプターと TCP データ入力アダプターが必要になります。

図 7-4　クエリグループへのデータ送信

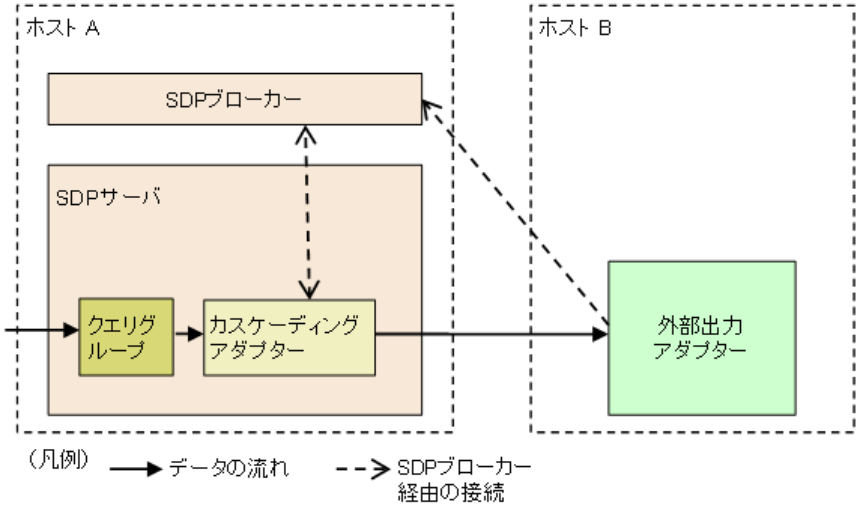


(2)　外部出力アダプターへのデータ送信

次の図は，外部出力アダプターに接続するカスケーディングアダプターの構成を示しています。外部出力アダプターが TCP 接続を要求すると，SDP ブローカー連携によってカスケーディングアダプターが自動起動します。起動後，カスケーディングアダプターは確立された接続を使用して外部出力アダプターにデータを送信します。

この構成では，アダプター構成定義ファイルの作成を省略できます。

図 7-5　外部出力アダプターへのデータ送信

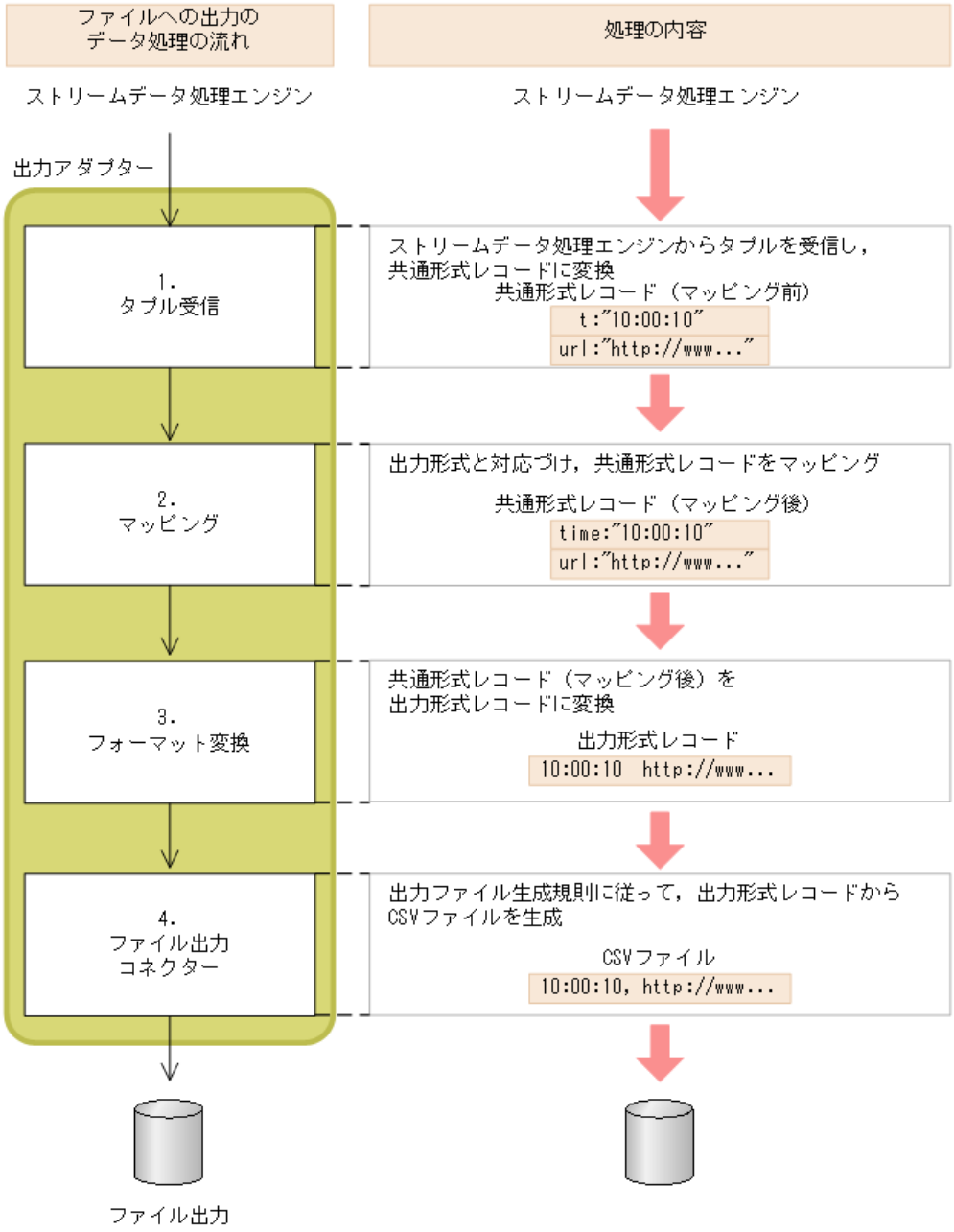


7.3.2　ファイル出力アダプター

ストリームデータ処理の結果をファイルに出力する場合，出力用コールバックとしてファイル出力コネクターを指定します。

ファイル出力コネクターでは，直前のコールバックから出力形式レコードを取得し，CSV データに変換します。ファイルへの出力でのコールバックの構成と処理の内容を次の図に示します。

図 7-6　ファイルの出力でのコールバックの構成と処理の内容

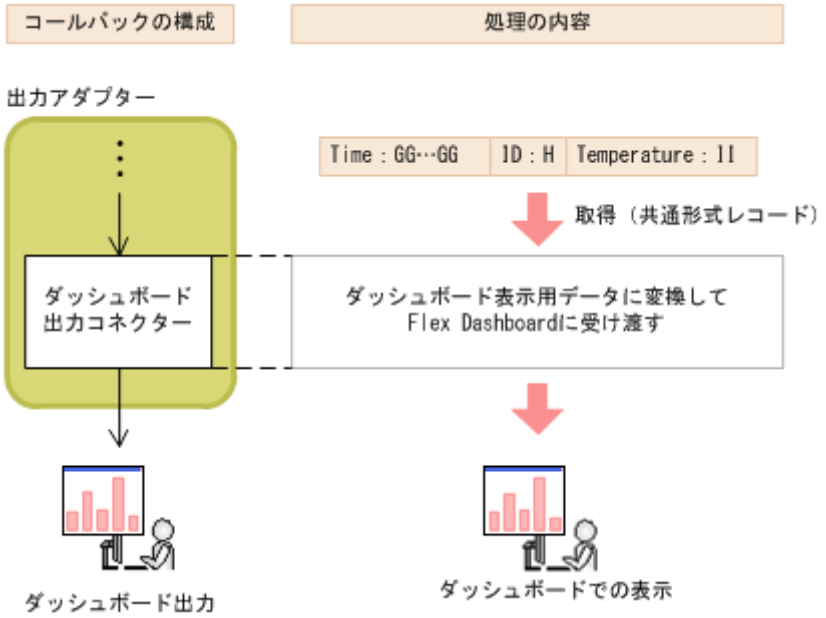


7.3.3　ダッシュボード出力アダプター

ストリームデータ処理の結果をダッシュボードに出力して表示する場合，出力用コールバックとして**ダッシュボード出力コネクター**を指定します。ダッシュボードに出力されたデータは，折れ線グラフや棒グラフで表示できます。

ダッシュボード出力コネクターでは，直前のコールバックから共通形式レコードを取得します。そのあと，ダッシュボード表示用データに変換します。ダッシュボードへの出力でのコールバックの構成と処理の内容を次の図に示します。

図 7-7　ダッシュボードへの出力でのコールバックの構成と処理の内容



7.4　内部アダプターにおけるデータ編集

ここでは，レコードのフィルタリングと抽出について説明します。

ほかにも，内部アダプターでのデータ編集として，レコードのフォーマット変換，およびレコードとストリーム間のマッピングができます。詳細はマニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

7.4.1　レコードのフィルタリング

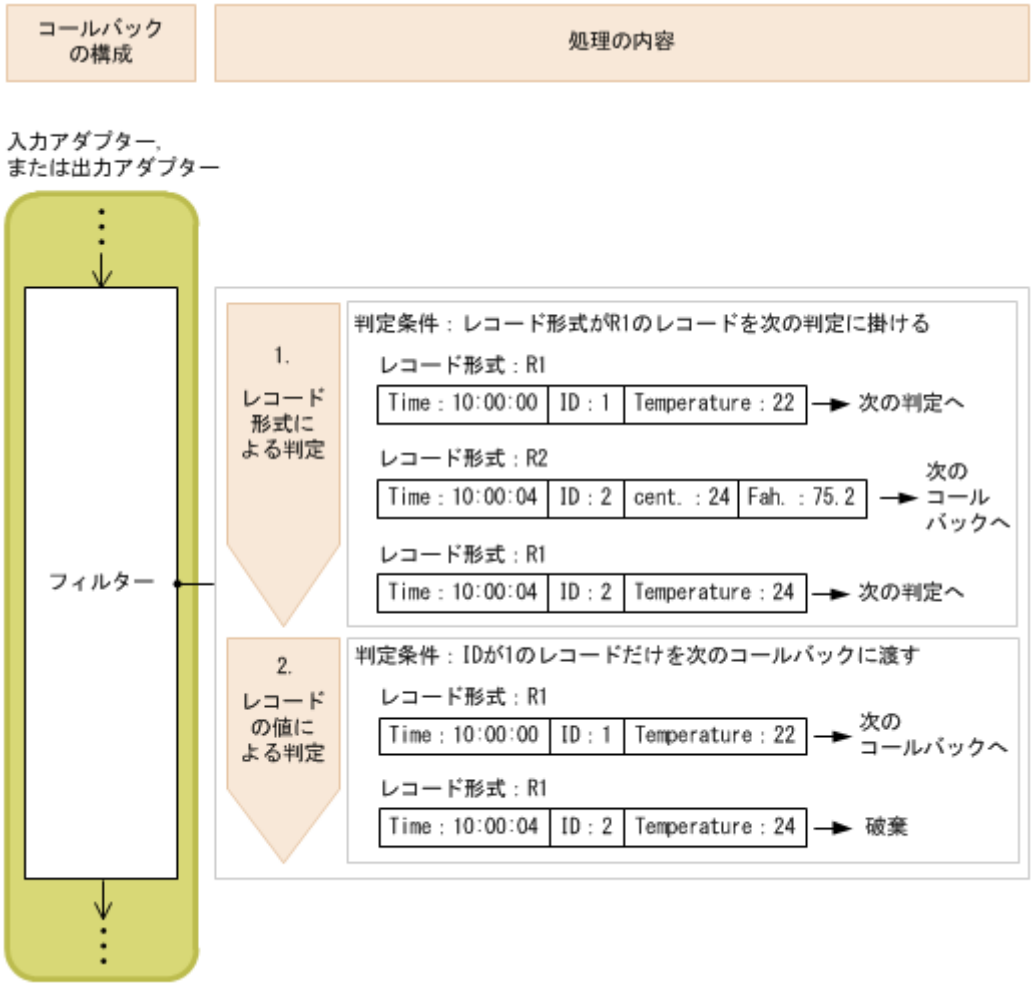
特定のレコードだけをストリームデータ処理の対象にしたい場合，編集用コールバックとして，**フィルター**を使用します。

例えば，複数の観測所で気温を観測している場合に，特定の観測所の気温だけを集計・分析の対象としたいときは，観測所に与えられている ID でフィルタリングを実行してください。

フィルタリングの対象は，共通形式レコードです。入力元がファイルの場合，ファイル入力コネクターで入力形式レコードを抽出したあと，フォーマット変換で共通形式レコードに変換してからフィルターを使用してください。

フィルタリングの判定条件には，レコード形式，およびレコードに設定されている値を指定できます。レコードのフィルタリングでのコールバックの構成と処理の内容を次の図に示します。

図 7-8　レコードのフィルタリングでのコールバックの構成と処理の内容



1. フィルターに入力されたレコードは，最初にレコード形式でフィルタリングされます。
   ここでは，レコード形式が R1 のレコードだけを次の判定に掛けるように条件を指定しているため，この条件を満たすレコードが，次の判定の対象となります。条件を満たさないレコードは，次のコールバックに渡されます。
2. レコード形式でフィルタリングしたあとは，レコードの値でフィルタリングします。
   ここでは，レコードの値に設定されている ID が 1 のレコードだけを次のコールバックに渡すように条件を指定しています。そのため，この条件を満たすレコードが，次のコールバックでの処理の対象となります。条件を満たさないレコードは，破棄されます。

7.4.2　レコードの抽出

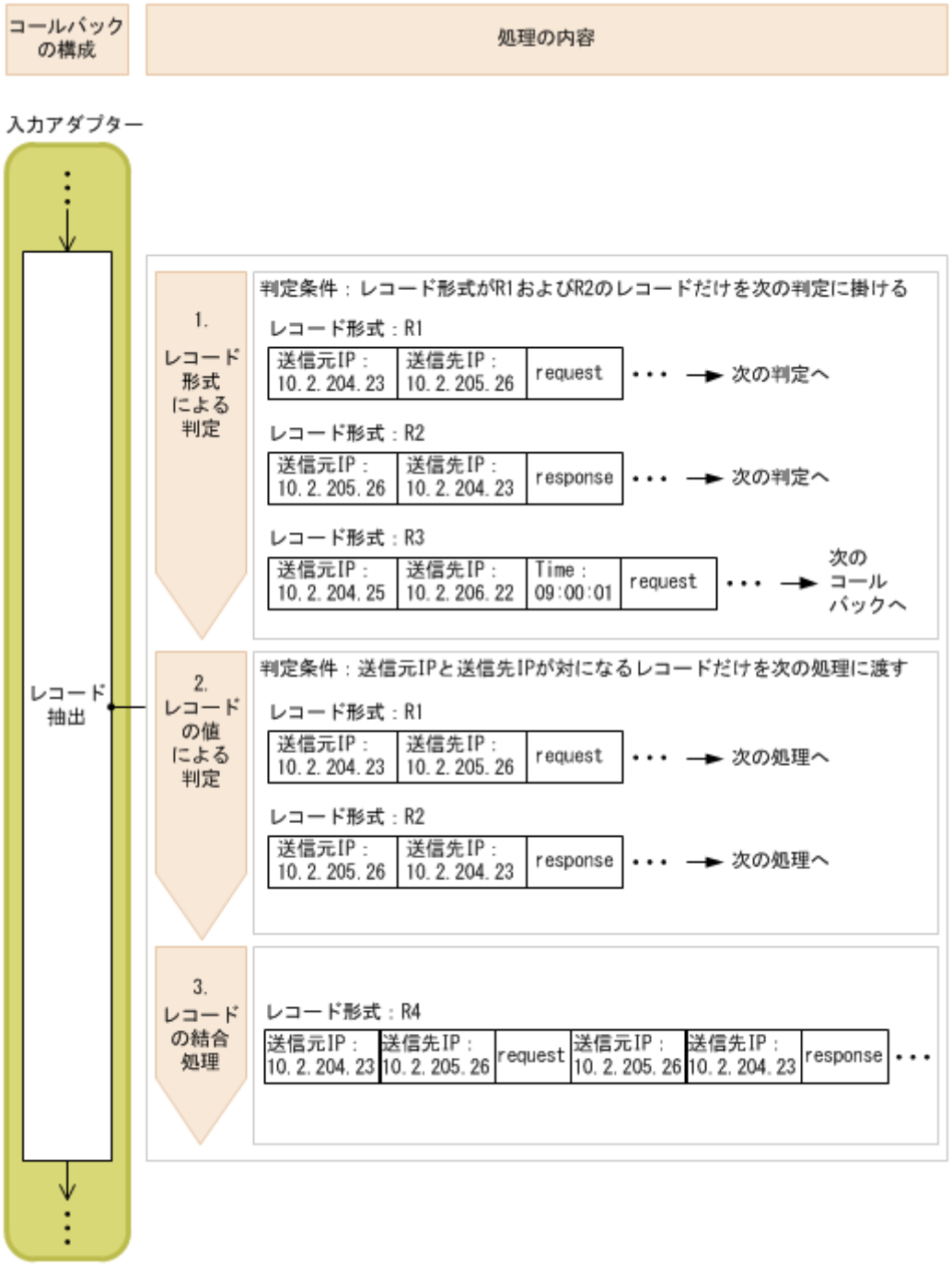
レコードを取捨選択して，特定のレコードを取得したあと，それらのレコードを意味のある 1 つのレコードにする場合，**レコード抽出**を使用します。

例えば，クライアントとサーバ間の応答状況を集計・分析の対象としたい場合，入力用コールバックに HTTP パケット入力コネクターを使用した上で，編集用コールバックとしてレコード抽出を使用します。

ネットワーク上のリクエストとレスポンスのそれぞれの HTTP パケットを，レコード抽出によって送信元 IP と送信先 IP で結び付けた 1 つのレコードにすれば，応答時間などが明確になり，集計・分析がしやすくなります。

レコード抽出では，レコード形式およびレコードに設定されている値で目的のレコードをフィルタリングしたあと，それらのレコードを結合して新たなレコードを生成します。レコードの抽出でのコールバックの構成と処理の内容を次の図に示します。

図 7-9　レコードの抽出でのコールバックの構成と処理の内容



1. レコード抽出に入力されたレコードは，最初にレコード形式でフィルタリングされます。

ここでは，レコード形式が R1 および R2 のレコードだけを次の判定に掛けるように条件を指定しているため，この条件を満たすレコードが，次の判定の対象となります。条件を満たさないレコードは，次のコールバックに渡されます。

2. レコード形式でフィルタリングしたあとは，レコードの値でフィルタリングします。

   ここでは，リクエストの送信元 IP および送信先 IP，ならびにレスポンスの送信元 IP および送信先 IP が，それぞれ対になるレコードだけを次の処理に渡すように条件を指定しています。そのため，この条件を満たすレコードが，次の処理での対象となります。

3. レコード形式，およびレコードに設定されている値でフィルタリングされたレコードを結合し，1 つのレコードとします。

   ここで結合されたレコードが，次のコールバックでの処理の対象となります。

7.5　内部アダプターの自動生成機能

標準提供内部アダプターのうち TCP データ入力アダプターとカスケーディングアダプターは，SDP ブローカーと連携することで，特定の契機に自動で生成・起動・停止します。アダプターを自動生成する場合は，そのアダプターに対するアダプター構成定義ファイル，およびインプロセス連携用プロパティファイルを作成する必要はありません。アダプターの自動生成・起動・停止の契機を次に示します。

アダプターの自動生成

クエリグループが登録されると，次のアダプターが自動的に生成されます。

- TCP データ入力アダプター（外部入力アダプター，およびカスケーディングアダプター接続用）
- カスケーディングアダプター（外部出力アダプター接続用）
- カスケーディングアダプター（TCP データ入力アダプター接続用）

アダプターの起動

アダプターの自動起動時に，ほかのコマンドが実行中のため起動に失敗した場合は，起動処理を 1 秒間隔で 30 秒間リトライします。リトライしても起動できなかった場合，エラーログがログファイルに書き込まれ，起動処理が中断されます。

データパラレル構成の場合，構成の種類によって，アダプターは次のように起動します。

スケールアウト構成

スケールアウト構成では，外部アダプター，およびカスケーディングアダプターから入出力ストリームへの接続が要求されると，スケールアウトされた各運用ディレクトリで TCP データ入力アダプター，およびカスケーディングアダプターが自動的に起動します。また，スケールアウト構成の SDP サーバが，カスケーディングアダプターで別の SDP サーバへデータを送信する構成の場合，スケールアウト構成の SDP サーバそれぞれでカスケーディングアダプターが自動的に起動します。

スケールアップ構成

スケールアップ構成では，外部アダプター，およびカスケーディングアダプターから入出力ストリームへの接続が要求されると，スケールアップされたクエリグループのスレッドと同じ数の TCP データ入力アダプター，およびカスケーディングアダプターが自動的に起動します。また，スケールアップ構成の SDP サーバが，カスケーディングアダプターで別の SDP サーバへデータを送信する構成の場合，`hsdpcqlstart` コマンドでクエリグループを起動すると，スケールアップされたクエリグループのスレッドと同じ数の，カスケーディングアダプターが自動的に起動します。

自動起動する各アダプターの詳細を説明します。

TCP データ入力アダプター（外部入力アダプター，およびカスケーディングアダプター接続用）

外部入力アダプター，またはカスケーディングアダプターからの接続要求を契機に，接続先の入力ストリーム宛のデータをストリームデータ処理エンジンに入力するための TCP データ入力アダプターを起動します。自動起動するためには，SDP ブローカーが起動している必要があります。

TCP データ入力アダプターの詳細は次のとおりです。

- 起動されるアダプターグループの名前

  tcpinput-*クエリグループ名*-*接続する入力ストリーム名*

  *クエリグループ名*の値は，接続されるストリームを定義するクエリグループの名前を示します。

- TCP データ入力アダプターの名前

  tcpinput[-*N*]

  数字-*N* は，スケールアップ構成の場合にだけ追加されます。*N* の値は 1 以上（3 桁の 10 進数，つまり 001，002 などのクエリグループに割り当てるスレッドの並列数）です。

カスケーディングアダプター（外部出力アダプター接続用）

外部出力アダプターからの接続要求を契機に，接続先の出力ストリームのデータをストリームデータ処理エンジンから取り出して送信するためのカスケーディングアダプターを起動します。自動起動するためには，SDP ブローカーが起動している必要があります。

カスケーディングアダプターの詳細は次のとおりです。

- 起動されるアダプターグループの名前

  cascading-out-*クエリグループ名*-*接続する出力ストリーム名*

  *クエリグループ名*の値は，接続されるストリームを定義するクエリグループの名前を示します。

- カスケーディングアダプターの名前

  cascading[-*N*]

  数字-*N* は，スケールアップ構成の場合にだけ追加されます。*N* の値は 1 以上（3 桁の 10 進数，つまり 001，002 などのクエリグループに割り当てるスレッドの並列数）です。

カスケーディングアダプター（TCP データ入力アダプター接続用）

カスケーディングアダプターは，出力ストリームからデータを取得し，それを接続先入力ストリームに送信します。クエリグループが起動すると，クエリグループ用プロパティファイルで指定された，接続先の入力ストリーム※にデータを送信する，カスケーディングアダプターが起動します。

注※

stream.output.*ストリーム名*.link プロパティの値として指定された入力ストリームを指します。

カスケーディングアダプターの詳細は次のとおりです。

- アダプターグループの名前

  cascading-*クエリグループ名*-*出力ストリーム名*-*M*

  *クエリグループ名*の値は，接続されるストリームを定義するクエリグループの名前を示します。

  *出力ストリーム名*の値は，stream.output.*ストリーム名*.link プロパティで*ストリーム名*として指定された，出力ストリームの名前を示します。

M の値は，接続先の入力ストリームの通番を表す 3 桁の 10 進数を示します。例えば，001，002 などです。この通番は，クエリグループ名の単位で自動的に割り振られます。

- カスケーディングアダプターの名前

```
cascading[-N]
```

数字*-N* は，スケールアップ構成の場合にだけ追加されます。*N* の値は 1 以上（3 桁の 10 進数，つまり 001，002 などのクエリグループに割り当てるスレッドの並列数）です。

アダプターの停止

クエリグループが停止すると，自動的に起動されたアダプターも停止します。

アダプターを自動生成するための設定

アダプターを自動生成するための設定は，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

7.6　内部カスタムアダプター

内部カスタムアダプターとは，入出力データと SDP サーバの間でデータをやり取りする機能を持つアプリケーションです。Hitachi Streaming Data Platform が提供する API を使用して作成します。内部カスタムアダプターを作成することで，内部標準アダプターでサポートされていない任意のデータ形式で入出力ができます。

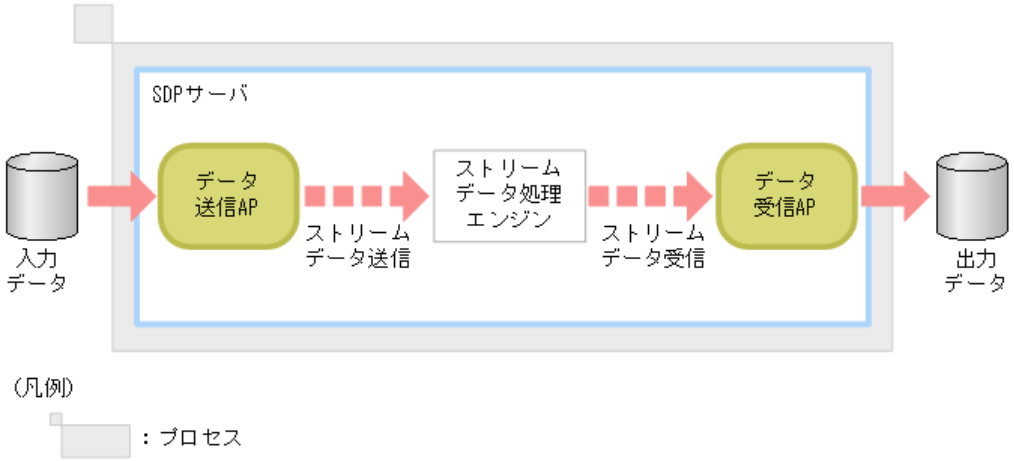
内部カスタムアダプターは，SDP サーバとのデータの送受信を，SDP サーバのプロセス内（インプロセス連携）で実行します。そのため，内部カスタムアダプターは，インプロセス連携カスタムアダプターとも呼ばれます。

内部カスタムアダプターは，処理内容によって，データ送信 AP とデータ受信 AP の 2 種類に分類できます。

- データ送信 AP
  入力データを受け付け，SDP サーバに対してストリームデータを送信します。
- データ受信 AP
  SDP サーバで分析した結果（クエリ結果）のストリームデータを受信します。

内部カスタムアダプターの位置づけについて，次の図に示します。

図 7-10　内部カスタムアダプターの位置づけ



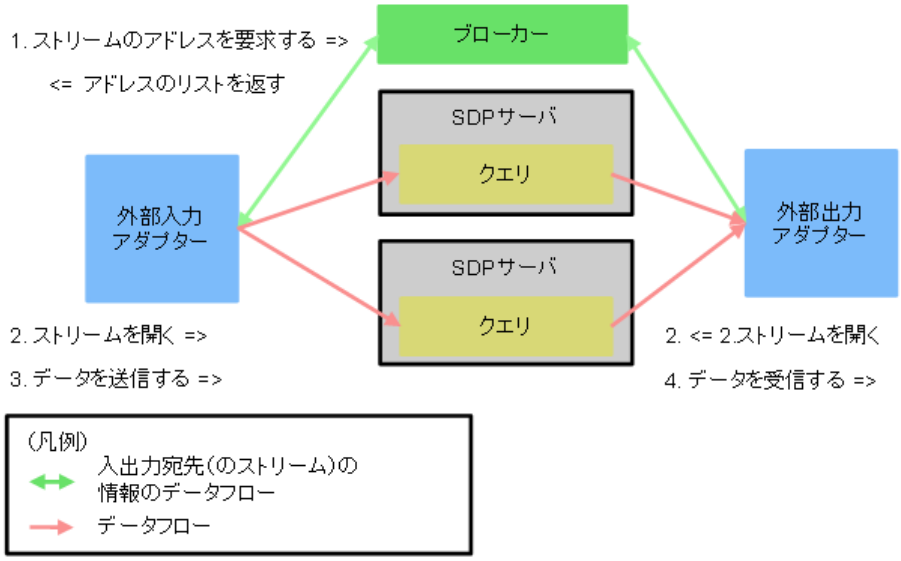
8 外部アダプター

この章では，SDP サーバに分析データを送信したり，SDP サーバから分析データを受信したりするために使用される，外部アダプターの機能について説明します。

8.1 外部アダプターとは

外部アダプターは，SDP サーバに分析データを送信したり，SDP サーバから分析データを受信したりするために使用されます。

外部アダプターの概要



説明

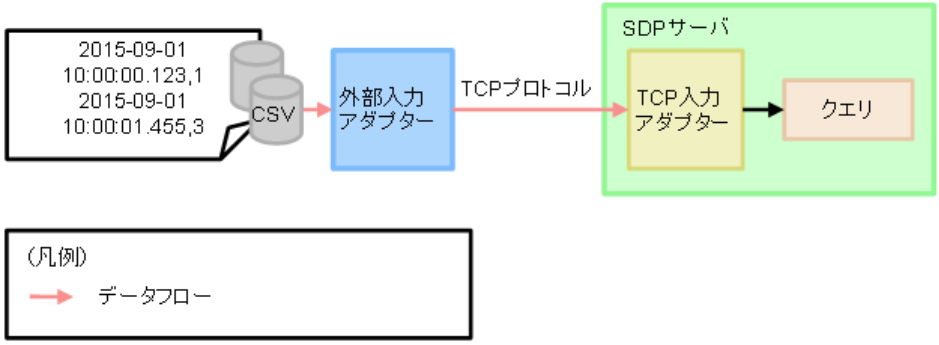
外部アダプターは，SDP サーバに分析対象となるデータを送信し，SDP サーバから分析結果を受信するのに使用されます。外部アダプターは SDP ブローカーからデータ送信ストリームまたは受信宛先ストリームのアドレスを取得し，対応する対象ストリームに接続します。SDP サーバがパラレル構成で実行している場合は，アダプターはすべての SDP サーバと接続します。このため開発者は，外部アダプターの開発時に個々の SDP サーバのアドレスを意識する必要はありません。

> **メモ**
>
> 外部アダプターを SDP サーバ以外のホストにデプロイすることもできます。

8.2　外部入力アダプター

外部入力アダプターは，SDP サーバに分析データを送信します。

外部入力アダプターの概要



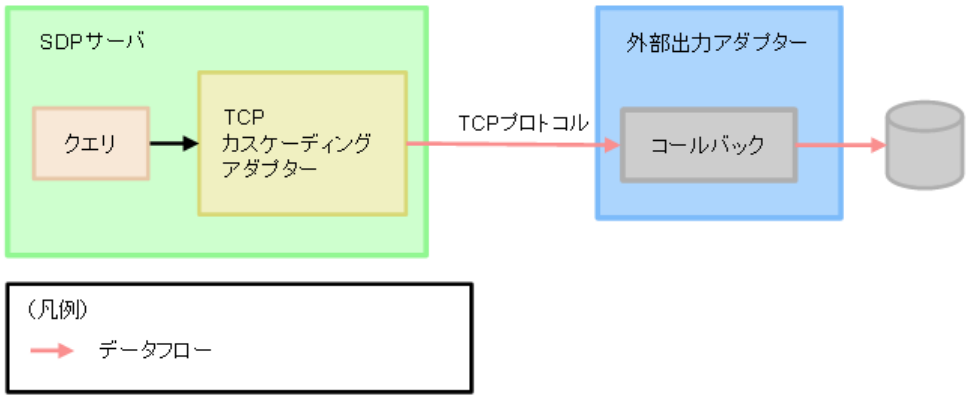
説明

外部入力アダプターは，SDP サーバ上で TCP データ入力アダプターと接続し，TCP IP プロトコルを介してデータを送信できます。

8.3　外部出力アダプター

外部出力アダプターは，SDP サーバから分析結果を受信します。

外部出力アダプターの概要



説明

外部出力アダプターは，SDP サーバから分析結果を受信します。このアダプターは SDP サーバ上で TCP カスケーディングアダプターと接続し，TCP プロトコルを介してデータを受信します。

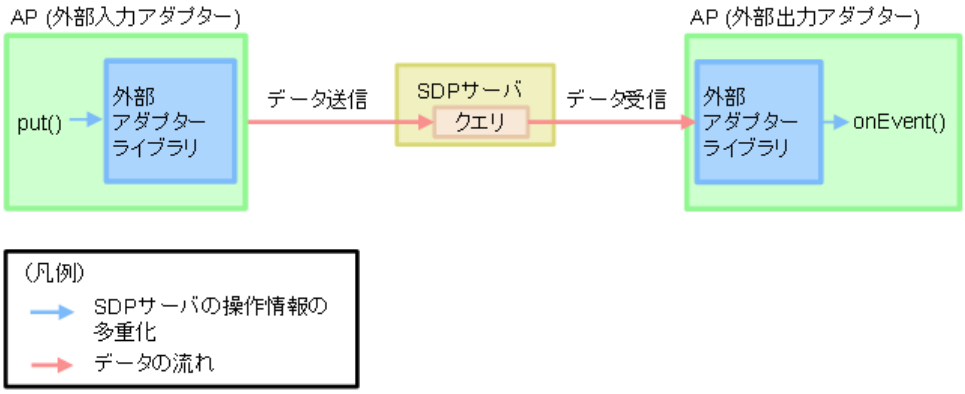
外部出力アダプターがデータを受信すると，外部出力アダプターによって登録されたコールバックが，非同期で呼び出されます。インテグレーションデベロッパーは，コールバックとして処理することで，受信したデータを処理できます。

複数の外部出力アダプターを使用して，同一の出力ストリームに接続できます。また，1 つの外部出力アダプターにコールバックを複数登録して，同一の出力ストリームに接続できます。これらの場合，各外部出力アダプターおよびコールバックに対応するカスケーディングアダプターが同じ数だけ起動し，出力ストリームに出力されたストリームデータがそれぞれのカスケーディングアダプターすべてに複製されます。

8.4　外部アダプターライブラリ

インテグレーションデベロッパーは，外部アダプターライブラリを使用して，外部入力アダプターおよび外部出力アダプターを作成します。

外部アダプターライブラリの概要



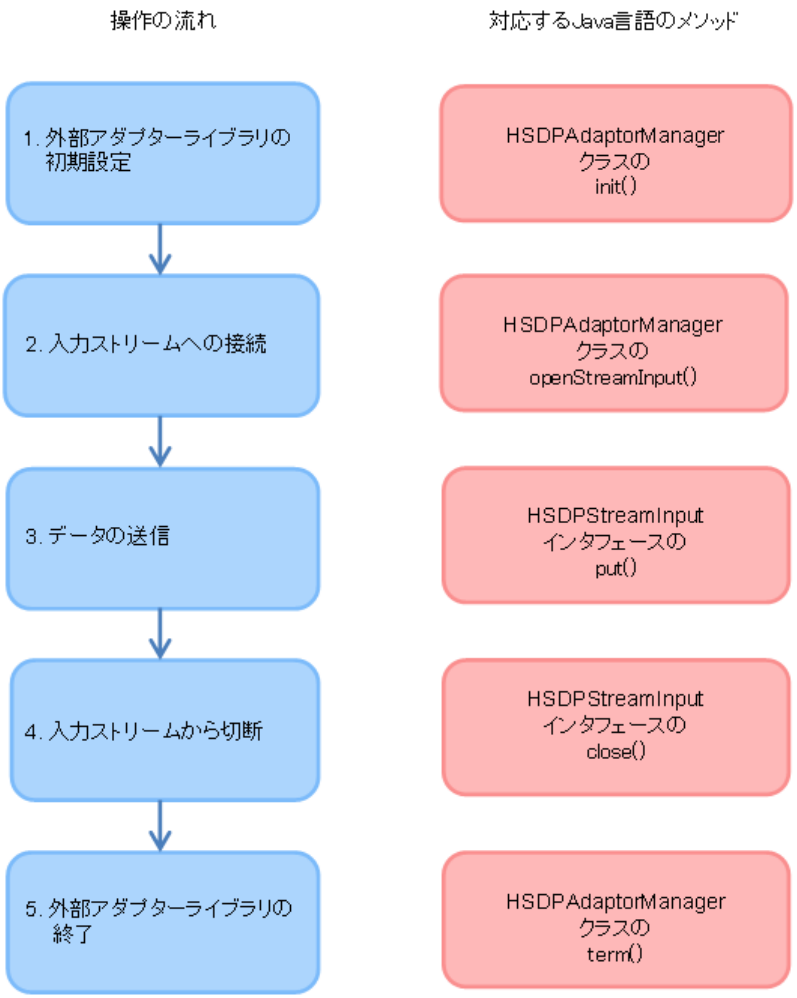
説明

外部アダプターライブラリは，外部入力アダプターまたは外部出力アダプターを Java アプリケーションとして作成するのに使用できます。

8.4.1　外部入力アダプターを作成するためのワークフロー

外部アダプターライブラリを使用して，外部入力アダプターを作成できます。

外部入力アダプターの操作の流れと，外部アダプターライブラリの実装メソッドおよび関数



説明

外部入力アダプターを作成するために実施される操作は次のとおりです。

1. 外部アダプターライブラリを初期設定します。外部アダプター定義ファイルのパスを指定します。

   メモ

   初期設定は，外部アダプターの起動後に 1 回だけ必要です。

2. SDP サーバの入力ストリームに接続します。SDP サーバがパラレル構成で実行している場合は，外部アダプターライブラリはすべての入力ストリームと接続します。

   メモ

   パラレル構成には，クエリグループに割り当てるスレッドの並列数を指定して登録されたクエリグループが含まれます。

3. 入力ストリームにデータを送信することで，データを伝送します。宛先 SDP サーバがパラレル構成で実行している場合は，指定した SDP サーバ設定によってデータディスパッチの方法が決められます。ただし，アプリケーションにカスタムディスパッチャーが指定された場合は，カスタムディスパッチャーのルールによって，データのディスパッチが制御されます。
4. 入力ストリームに送信するデータがなくなった場合は，入力ストリームとの接続を切断します。
5. 外部アダプターの動作を終了するには，外部アダプターライブラリの終了メソッドまたは関数を呼び出します。

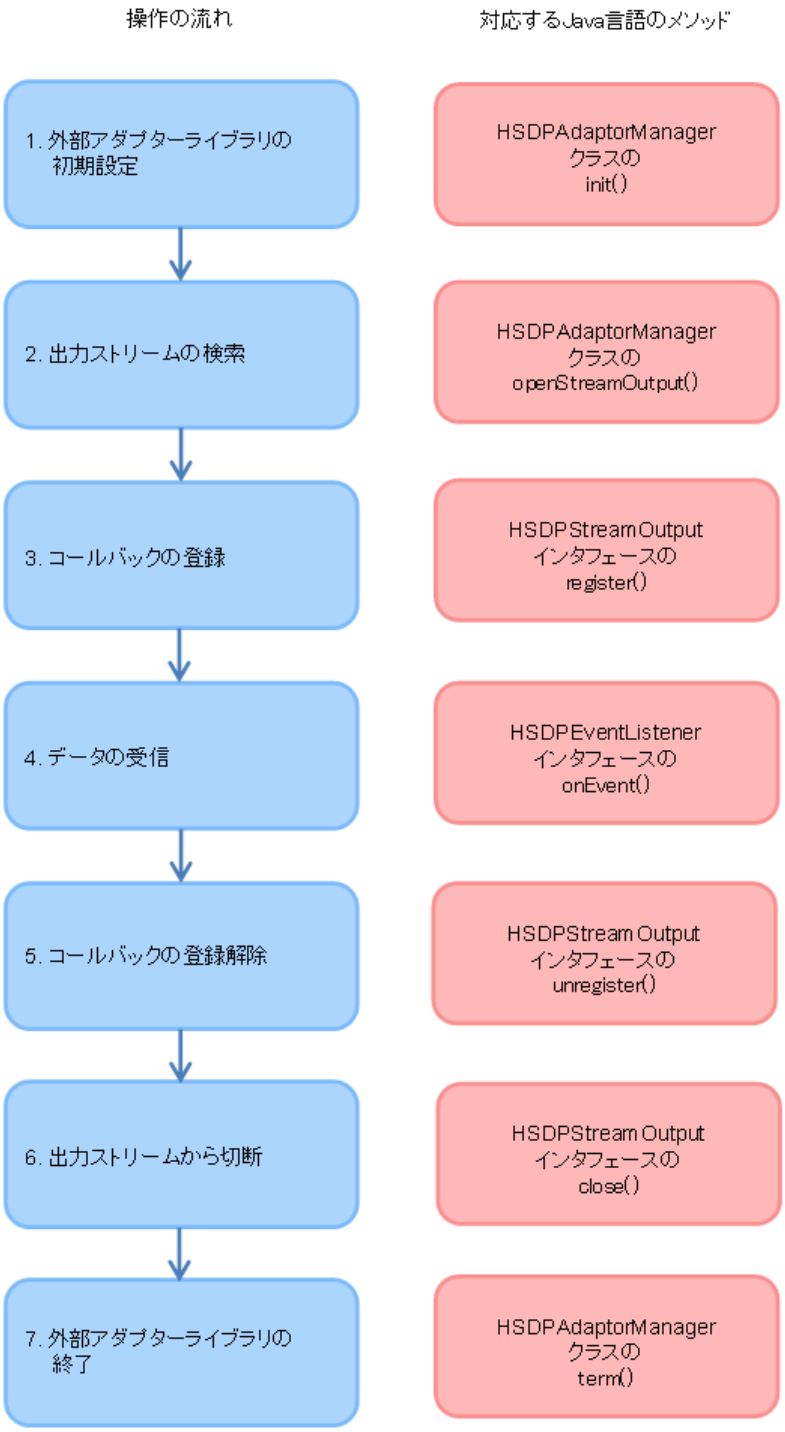
次の場所にある，Streaming Data Platform software development kit のサンプルファイルから，外部入力アダプターのプログラム例を使用できます。

```
/opt/hitachi/hsdp/sdk/samples/exadaptor/inputadaptor/src/ExternalInputAdaptor.java
```

8.4.2 外部出力アダプターを作成するためのワークフロー

外部アダプターライブラリを使用して，外部出力アダプターを作成できます。

外部出力アダプターの操作の流れと，外部アダプターライブラリの実装メソッドおよび関数



説明

外部出力アダプターを作成するために実施される操作は次のとおりです。

1. 外部アダプターライブラリを初期設定します。外部アダプター定義ファイルのパスを指定します。

> **メモ**
>
> 初期設定は，外部アダプターの起動後に 1 回だけ必要です。

2. SDP サーバの出力ストリームを探します。SDP サーバがパラレル構成で実行している場合は，外部アダプターライブラリはすべての出力ストリームを見つけようとします。
3. コールバックを登録して，データ受信処理を実行します。コールバックが登録されると，見つかった出力ストリームに接続します。外部アダプターで待機時間を設定し，接続先による分析が完了するまで，コールバックの登録がキャンセルされないようにしてください。接続先の分析が停止した場合は，コールバックの登録をキャンセルします。
4. コールバックが登録されると，データ受信のたびにコールバックが呼ばれます。コールバックの中で，通知された受信データから必要なデータを取得します。コールバックへはコールバック登録以降に受信したデータが通知されます。
5. コールバックの登録をキャンセルし，データの受信を終了します。
6. 出力ストリームから切断して，出力ストリームからの受信を終了します。
7. 外部アダプターを終了するには，外部アダプターライブラリの終了処理を実施します。

次の場所にある，Streaming Data Platform software development kit のサンプルファイルから，外部出力アダプターのプログラム例を使用できます。

```
/opt/hitachi/hsdp/sdk/samples/exadaptor/outputadaptor/src/ExternalOutputAdaptor.java
```

8.4.3 コールバックを作成する

HSDPEventListener インタフェースを実装するクラスを作成し，onEvent()メソッドのコールバック時に実行される処理を記述することでコールバックを作成できます。

説明

HSDPEventListener インタフェースを実装するクラスのオブジェクトをHSDPStreamOutput インタフェースのregister()メソッドを使用して登録した後で，タプルが SDP サーバで作成されると，登録されたオブジェクトのonEvent()メソッドがコールバックされます。

次の場所にある，Streaming Data Platform software development kit のサンプルファイルから，コールバックのプログラム例を使用できます。

```
/opt/hitachi/hsdp/sdk/samples/exadaptor/outputadaptor/src/ExternalOutputAdaptor.java
```

8.5　並列処理する SDP サーバに接続する

SDP サーバがパラレル構成で実行している場合，外部アダプターは SDP ブローカーからすべてのアドレスを取得します。これは，任意の SDP サーバに接続するために使用されます。このため開発者は，外部アダプターの開発時に個々の SDP サーバのアドレスを考慮する必要はありません。

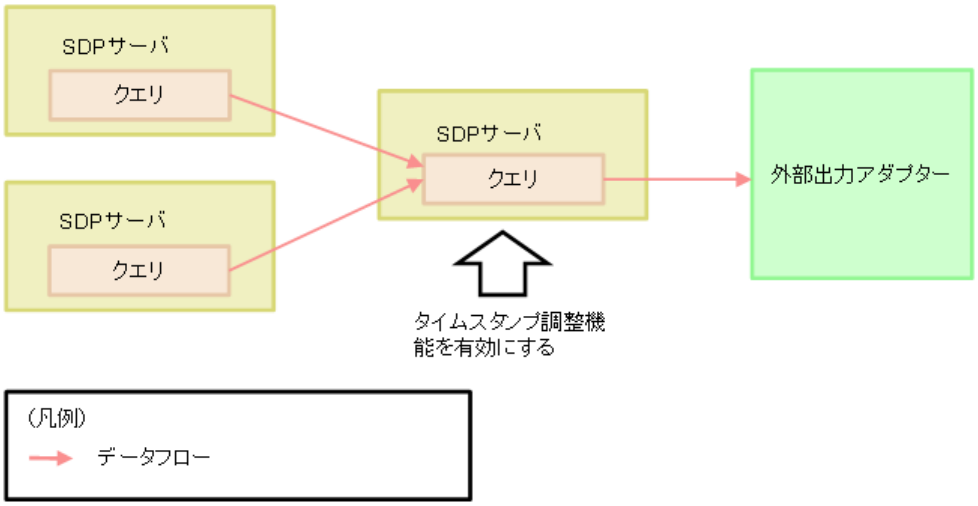
- 外部入力アダプターから並列処理する SDP サーバへ接続する場合

  SDP サーバ設定は，外部入力アダプターが，並列処理する SDP サーバにデータを送信する際に使用する，負荷分散方式を確認します。各負荷分散方式（ハッシュ化，ラウンドロビンなど）については，カスケーディングアダプターが負荷分散する場合と同様です。詳細については，「7.3.1　カスケーディングアダプター」を参照してください。ただし，カスタムディスパッチャーを使用する場合，SDP サーバの設定ではなく，外部アダプターの設定によって負荷分散方式が決まります。カスタムディスパッチャーの詳細については，「8.6　カスタムディスパッチャー」を参照してください。

- 外部出力アダプターから並列処理する SDP サーバへ接続する場合

  外部出力アダプターが，並列処理する複数の SDP サーバからデータを受信すると，コールバックによってデータを受信した順にデータの通知が送信されます。これらのデータ通知は時系列順になっていません。通知を時系列順で受信したい場合は，次の図のように，タイムスタンプ調整機能を有効にした SDP サーバを介することで，データをソートする必要があります。

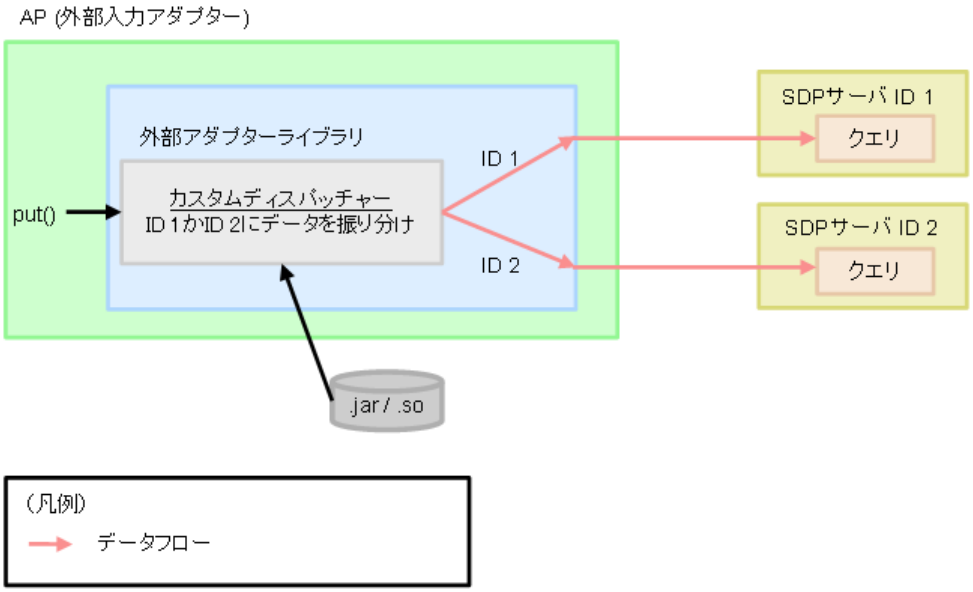
タイムスタンプ調整機能を使用してタプルを時系列にソートする



8.6　カスタムディスパッチャー

カスタムディスパッチャーを使用すると，SDP サーバではなく，外部アダプターで負荷分散方式を決めることができます。

カスタムディスパッチャーの概要



説明

カスタムディスパッチャーは，外部アダプターの任意の構成に基づくディスパッチ先を決めるのに使用できます。

> **メモ**
>
> HSDP 標準の負荷分散方式に基づくディスパッチ先を決めるためにカスタムディスパッチャーを使用しないでください。

カスタムディスパッチャーを使用するには，次の条件を満たしている必要があります。

- 外部アダプターファイル用にカスタムディスパッチャーが作成されている必要があります。これらのファイルには，Java 用の.jar または.class ファイルが含まれます。
- 宛先 SDP サーバのクエリグループ用プロパティファイルで，`stream.input.`*ストリーム名*`.dispatch.type=custom` が設定されている必要があります。

8.6.1　クラスファイルの作成ルール

カスタムディスパッチャーを実装するクラスファイルは，引数のないコンストラクタを使用してインスタンスを生成します。任意のパッケージ名またはクラス名を指定できます。

説明

クラスファイルを作成するには，次の条件を満たしている必要があります。

- 値public をクラス修飾子に指定する必要があります。抽象クラス（abstract）は利用できません。
- デフォルトのコンストラクタを使用するか（コンストラクタを作成しないでください），引数なしのコンストラクタを作成することで，引数ありのコンストラクタだけ利用できるインスタンスを避けてください。値public（引数なし）をコンストラクタの修飾子に指定する必要があります。

  > **メモ**
  >
  > クラスファイルの作成に必要な条件が満たされない場合は，カスタムディスパッチャーを実装するクラスファイルからインスタンスを生成できません。この結果，カスタムディスパッチャーを登録するHSDPStreamInput インタフェースファイルのloadDispatcher メソッドからエラーが発生します。

- HSDPDispatch インタフェースが，カスタムディスパッチャーを実装するクラスファイルに実装される必要があります。次のメソッドがインタフェースによって実装されなければなりません。

  ```
  public int dispatch(HSDPDispatchInfo dispatchInfo, byte[] data);
  ```

- ディスパッチ先の ID を返すメソッドを実装します。

  > **メモ**
  >
  > ID は，宛先の数を基に1 から割り当てられます。これはスケールアップおよびスケールアウト構成の両方の宛先に適用されます。
  >
  > このメソッドは，HSDPStreamInput インタフェースファイルのput メソッドの実行によって実行されます。

（カスタムディスパッチャーを実装する）クラスファイルパッケージの名前が含まれるクラス名が，外部アダプターの実行時に指定されたクラスパスのクラスファイルと同じクラス名でないことを確認します。クラス名が同じ場合は，外部アダプターの実行時にクラスパスで指定されたクラスが優先して読み込まれ，外部アダプターが正常に動作しないおそれがあります。

外部アダプターが実行している場合，カスタムディスパッチャーを実装するクラスファイルのパスを外部アダプターのクラスパスに指定しないでください。指定されると，外部アダプターの実行中にカスタムディスパッチャーを変更できません。

8.6.2 dispatch メソッドの実装例

カスタムディスパッチャーの dispatch メソッドを実装する例として，Java の外部アダプターを使用します。

説明

Java 外部アダプター

ディスパッチ先の ID を返すdispatch メソッドを実装します。整数型の最初のカラムを参照して，ディスパッチ先を決定する例を次に示します。

```
public class Dispatcher implements HSDPDispatch {

    @Override
    public int dispatch(HSDPDispatchInfo dispatchInfo, byte[] data) {
        // Destination ID
    int destID;
    // The number of destinations
    int destNum = dispatchInfo.getDestNum();

        // The First column is of VARCHAR type (two byte header + String -type data).
        ByteBuffer buffer = ByteBuffer.wrap(data);
        byte[] val1 = new byte[buffer.getShort()];
        buffer.get(val1);
        // Determine the destination.
        destID = new String(val1).hashCode() % destNum + 1;

    return destID;
    }
}
```

8.7 ハートビートの送信

投入するデータがなくなっても，ハートビートを一定の間隔で送信すると，分析を停止せずに SDP サーバの時刻を進めることができます。

SDP サーバがデータソースモードで実行している場合，入力されるデータがなくなると，SDP サーバの時刻が進まなくなります。このため，クエリから分析結果の出力が得られなくなります。

この問題は，分析を停止し，クエリから対象ストリームを削除することで解決できます。または，分析を停止せずに，一定の間隔でハートビートを送信することで，SDP サーバの時刻が進むようにできます。Java の`HSDPStreamInput` インタフェースの`heartbeat()`メソッドを使用してハートビートを送信できます。

8.8 トラブルシュート

外部アダプターの実行時にエラーが発生すると，メッセージログファイルとトレースログファイルが，それぞれのファイルの場所に出力されます。ただし，ログファイルは，init()メソッドの実行前または HSDPAdaptorManager クラスのterm()メソッドの実行後は出力されません。

関連トピック

ログファイルの場所および仕様については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

8.9 HSDP がインストールされていないサーバでの外部アダプターの実行

外部アダプターは，HSDP がインストールされていないサーバ上でもストリームデータを送受信できます。

HSDP がインストールされていないサーバ上で外部アダプターを動作させる場合，HSDP がインストールされたホストから次のライブラリを取得する必要があります。

表 8-1 HSDP がインストールされていないサーバ上の外部アダプターの動作に必要なライブラリ

項番	ライブラリ	格納場所
1	hsdp-exadp.jar	/opt/hitachi/hsdp/lib/
2	hntrlib2j64.jar	/opt/hitachi/HNTRLib2/classes/

HSDP がインストールされたホストから取得したライブラリを任意のディレクトリに配置します。これらのライブラリをクラスパスに含めて，外部アダプターを実行します。

実行例

java -classpath *ライブラリ格納ディレクトリ*/hsdp-exadp.jar: *ライブラリ格納ディレクトリ*/hntrlib2j64.jar:./bin *外部アダプタークラス名*

外部アダプターのログの収集は，hsdplogcollect コマンドではなく手動で実施します。外部アダプターのログファイルについては，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

9 HSDP クライアント

この章では，HSDP クライアントについて説明します。

9.1 HSDP クライアントの概要

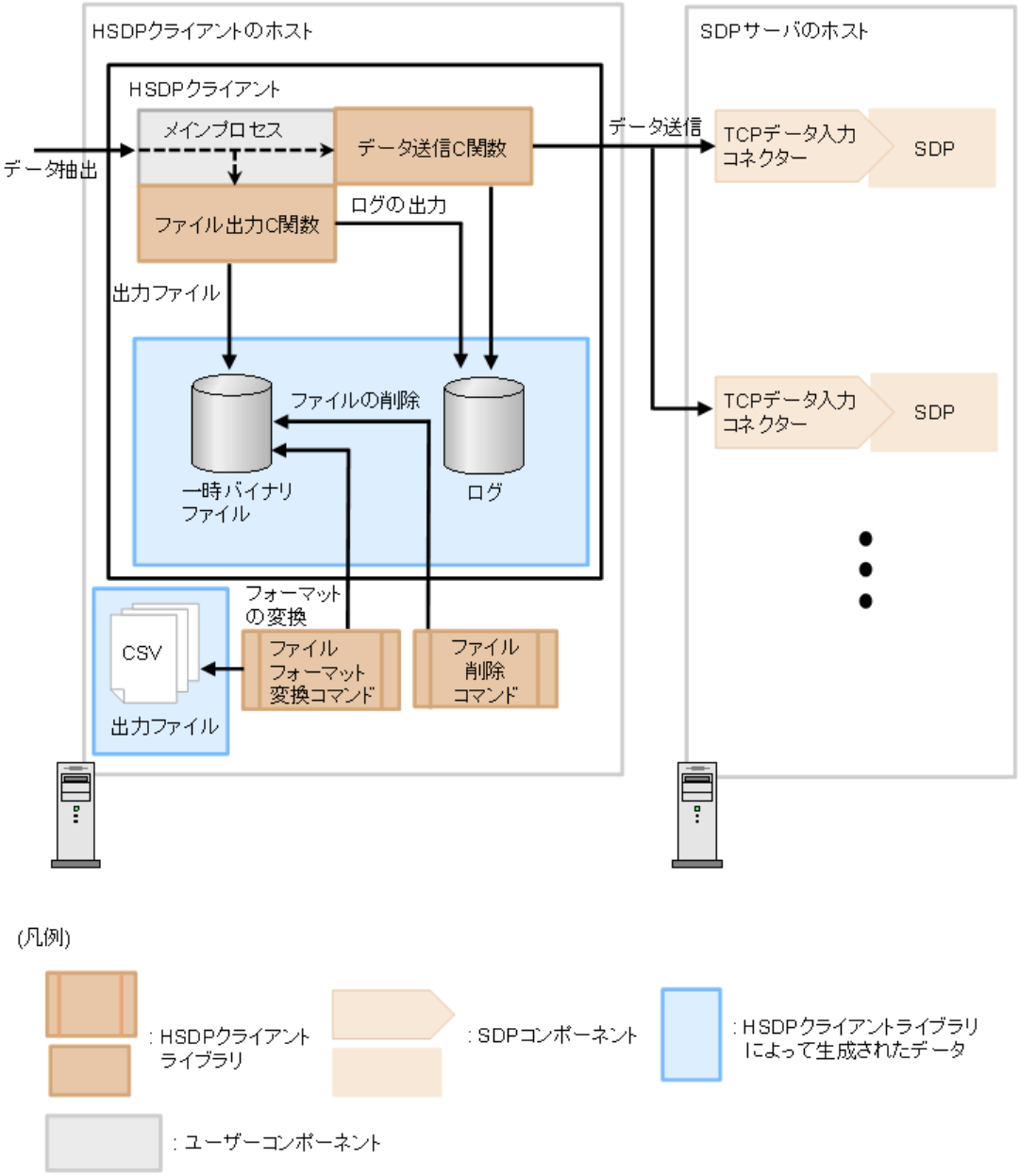
HSDP クライアントとは，HSDP クライアントライブラリを利用してユーザーが作成する，ファイルの出力および HSDP 上の TCP データ入力アダプターへの高性能（高速）なデータ送信を実行するためのユーザープログラムです。外部アダプターと異なり，HSDP クライアントは接続先の TCP データ入力アダプターを SDP ブローカー連携によって自動起動することはできません。HSDP クライアントを接続する TCP データ入力アダプターは，hsdpstartinpro コマンドによって手動起動してください。

HSDP クライアントライブラリは，次の 3 つの機能を提供します。

- データ送信
- ファイル出力
- トラブルシュート

HSDP クライアントライブラリには，C 関数とコマンドが含まれています。C 関数には，初期化，終了，データ送信，およびファイル出力用アドレスを取得するための関数が含まれています。HSDP クライアントの開発者は，これらの関数をインストールする必要があります。コマンドは，ファイルのフォーマット変換とファイルの消去（削除）に使用されます。これらのコマンドは，HSDP クライアントで出力されたファイルをほかのプログラムで読み込めるよう任意のフォーマットに変換したり，これらのファイルを削除したりする場合に使用できます。次の図に，HSDP クライアントライブラリの概要を示します。

図 9-1　HSDP クライアントライブラリの概要



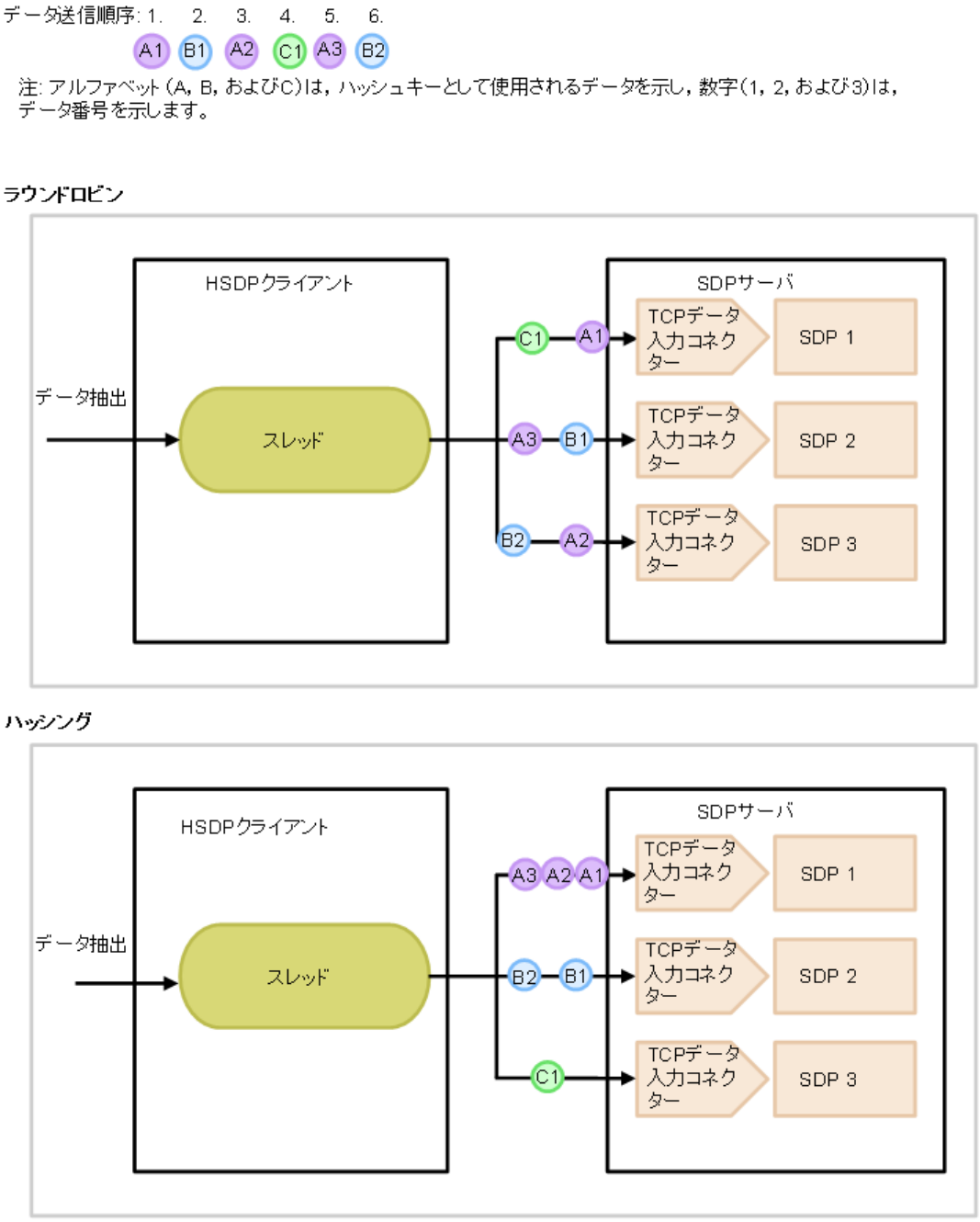
9.2 データ送信機能

HSDP クライアントのデータ送信機能は，TCP データ入力コネクターへの高性能（高速）なデータ送信に使用されます。

次の表でデータ送信機能について説明します。

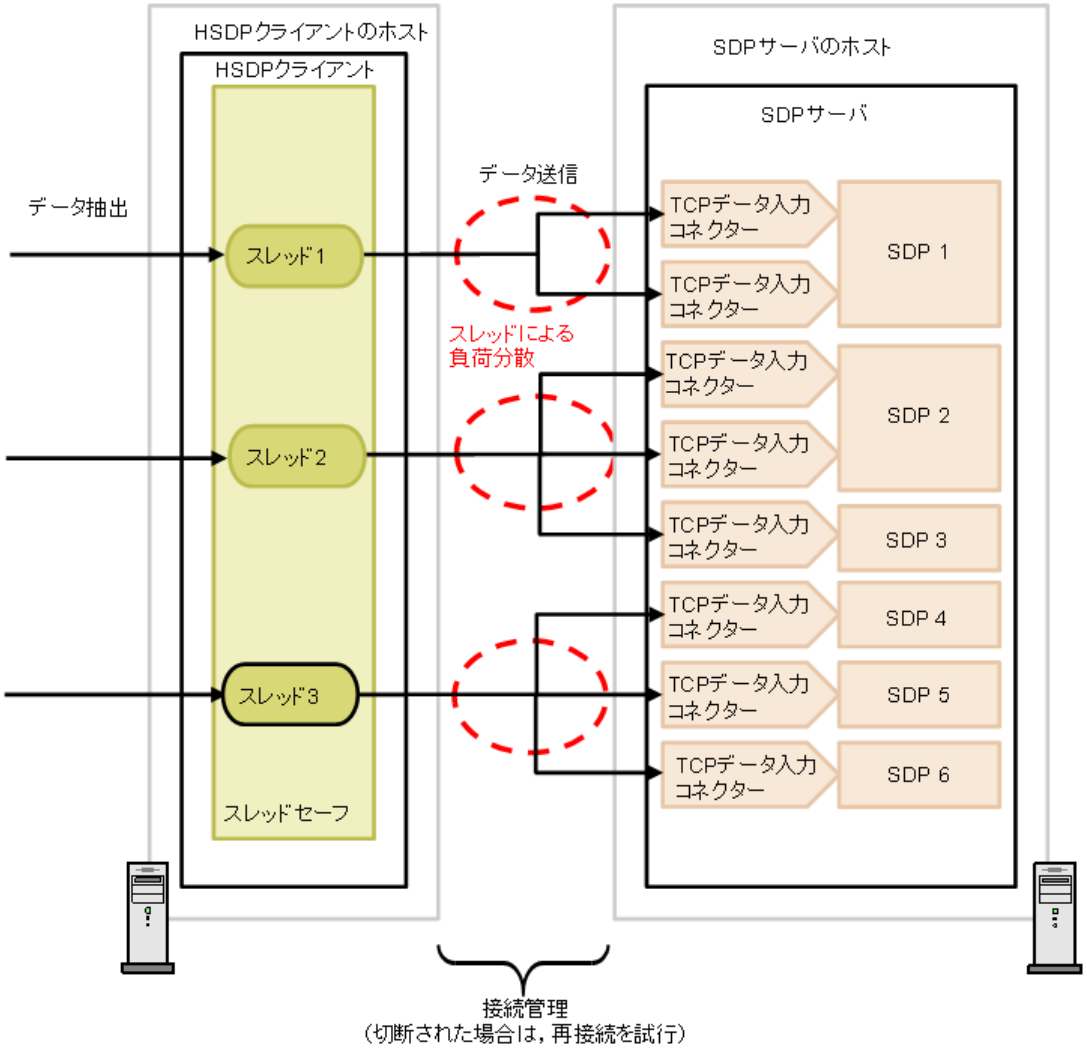
機能	説明
HSDP へのデータ送信	HSDP クライアントは，TCP データ入力コネクターにデータを送信します。HSDP クライアントと HSDP は，別々のホストにインストールできます。送信するデータの形式を指定できます。
接続管理	HSDP クライアントと TCP データ入力コネクターの接続が切断されると，HSDP クライアントは，自動的に再接続を試みます。
スレッドセーフ	HSDP クライアントは，マルチスレッドを使用することで，複数の TCP データ入力コネクターにデータを送信できます。HSDP クライアントライブラリは，マルチスレッドをサポートします。
負荷分散	データ送信スレッドは，次の 2 種類の負荷分散の方法によって，複数の HSDP にデータを送信できます。 ラウンドロビン データを各 TCP データ入力コネクターに順番に送信することで，作業負荷を分散します。これは，同等の作業負荷を SDP サーバに分散させる場合に使用できます。 ハッシング 複数の列から成るハッシュキーに基づき，複数の TCP データ入力コネクターにデータを送信して，作業負荷を分散します。同じキーを持つデータは同じ SDP サーバに送信できます。例えば，モニター ID が A のデータは，SDP サーバ X に送信できます。 負荷分散については，次の図を参照してください。

図 9-2　負荷分散の方法の概要（HSDP クライアントの 1 スレッドから 3 つの TCP データ入力コネクターに分散する場合）



次の図に，データ送信機能の概要を示します。

図 9-3　データ送信機能の概要



9.3 ファイル出力機能

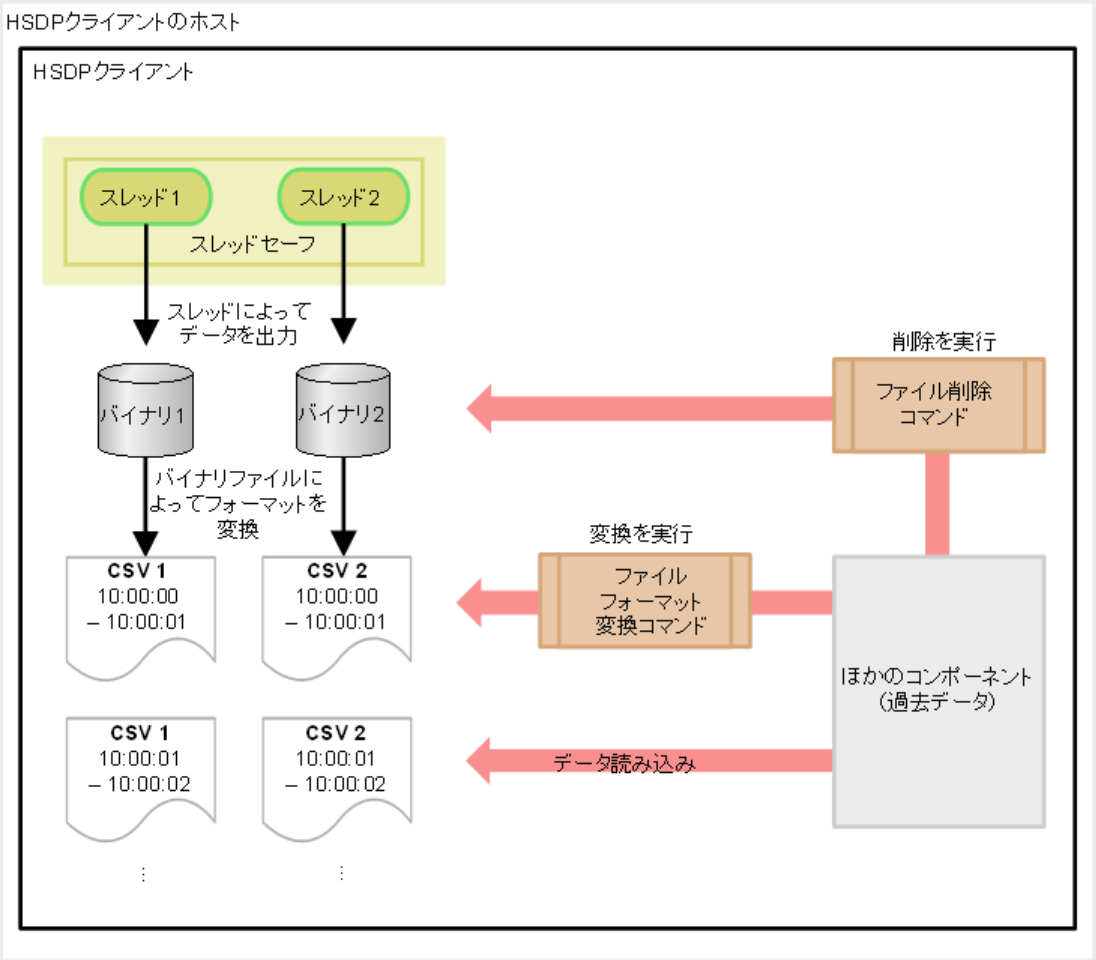
HSDP クライアントのファイル出力機能は，高性能（高速）でバイナリファイルにデータを出力します。ほかのプログラムで使用するために，ファイルをほかのフォーマットに変換するコマンドと，ファイルを削除するコマンドも提供されます。

次の表でファイルの出力機能について示します。

機能	説明
バイナリファイルのデータ出力	HSDP クライアントは，データのコピーだけを使用して，バイナリファイルにデータを出力できます。HSDP クライアントの各スレッドは，ファイル出力機能によって返されたアドレスにデータをコピーします。これによって，指定したディレクトリにバイナリファイルとしてデータが出力されます。
スレッドセーフ	HSDP クライアントは，マルチスレッドを使用することで，データをバイナリファイルとして出力できます。HSDP クライアントライブラリは，マルチスレッドをサポートします。
ファイルフォーマット変換コマンド（hsdpcliconv コマンド）	ファイルフォーマット変換コマンド（hsdpcliconv コマンド）は，HSDP クライアントのファイル出力機能によって出力されたバイナリファイルを別のファイルフォーマットに変換する場合に使用できます。ファイルフォーマット変換では，次のフォーマットがサポートされています。 • CSV
ファイル削除コマンド（hsdpclirm コマンド）	ファイル削除コマンド（hsdpclirm コマンド）は，HSDP クライアントのファイル出力機能によって出力されたバイナリファイルを削除する場合に使用できます。このコマンドには，次の 2 種類の条件を設定できます。 時間に基づく条件 指定した時間の前にバイナリファイルが削除されます。 容量に基づく条件 バイナリファイルは，バイナリファイルの合計サイズが指定したサイズより小さくなるよう削除されます。

次の図に，ファイル出力機能の概要を示します。

図 9-4　ファイル出力機能の概要



9.4　トラブルシュート機能

HSDP クライアントライブラリは，C 関数が呼び出された場合や，データ送信時やファイルの出力時にエラーが発生した場合などに，独自のメッセージおよびトレースログを出力します。

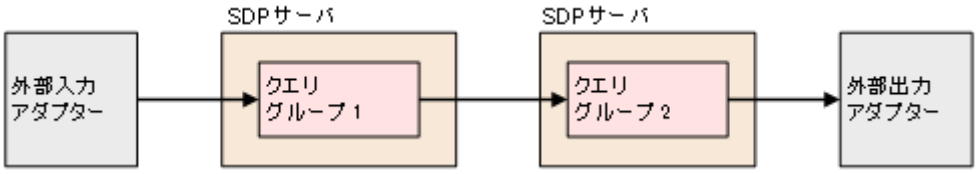
10 データパラレル構成，スケールアップ構成，およびスケールアウト構成

この章では，データパラレル構成，スケールアップ構成，およびスケールアウト構成について説明します。

10.1　データパラレル構成

分析シナリオを複数のプロセスまたはスレッドに分散して処理することを「データパラレル」と呼びます。データパラレルによる処理では，クエリの処理を負荷分散できるため，分析シナリオを 1 つのプロセスまたはスレッドで動かしている場合に比べ，高速にデータを分析できます。データパラレルを実現するシステム構成として，スケールアップ構成とスケールアウト構成があります。

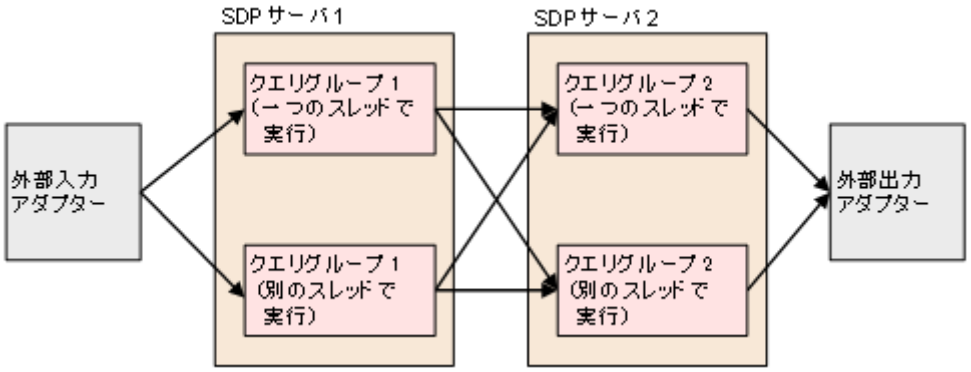
次のクエリグループ 1 とクエリグループ 2 から成る分析シナリオを，スケールアップ構成，およびスケールアウト構成で分析する場合を例にとり，それぞれの構成の詳細を説明します。



10.1.1　スケールアップ構成

スケールアップ構成では，同じクエリグループが SDP サーバ内の複数のスレッドで並列に実行されます。処理の多重化に十分な CPU リソースがありながら，メモリリソースの利用が限られる場合は，ハイパフォーマンスシステムを構築するのに，スケールアウト構成ではなくスケールアップ構成を選択してください。

クエリグループを処理するためのスケールアップ構成



図は，クエリグループ 1 および 2 を処理するスケールアップ構成を示しています。クエリグループへのデータは分散して送信されます。

- 使用方法

  クエリグループは，`-thread` オプションにクエリグループに割り当てるスレッドの並列数を指定した `hsdpcql` コマンドを実行することで登録できます。なお，並列のスレッドで稼働するクエリグループへ送信されるデータの負荷分散方式は，その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で指定します。

- クエリグループ名

  クエリグループの登録時に，hsdpcql コマンドの-thread オプションを使用してクエリグループに割り当てるスレッドの並列数に 2 以上を指定すると，SDP サーバに以下の名前のクエリグループが登録されます。これらのクエリグループは，SDP サーバで並列に実行されているスレッドに対応します。

  定義されたクエリグループ名-N

  定義されたクエリグループ名：hsdpcql コマンドで指定されるクエリグループ名

  N：1 からクエリグループに割り当てるスレッドの並列数（3 桁の 10 進数）

- 内部標準アダプター

  スケールアップ構成 SDP サーバの内部標準アダプターを起動するには，hsdpstartinpro コマンドをスケールアップしたクエリグループの数だけ実行します。これらのアダプターのアダプター構成定義ファイルには，接続先クエリグループ名として，クエリグループ名のセクションに示す番号付きのクエリグループ名を指定してください。

  なお，SDP ブローカー連携によって自動起動するスケールアップ構成 SDP サーバの内部 TCP データ入力アダプターについては，スケールアップしたクエリグループ数分のアダプターが自動で起動します。このとき，TCP データ入力アダプターを自動起動するための設定（データ送信元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のstream.output.*ストリーム名*.link プロパティや外部入力アダプターの外部アダプター定義ファイルのhsdp.target.name.*n* プロパティなど）における，TCP データ入力アダプターに接続するクエリグループ名として，クエリグループ名のセクションに示す番号付きのクエリグループ名ではなく，元々の定義されたクエリグループ名を指定してください。

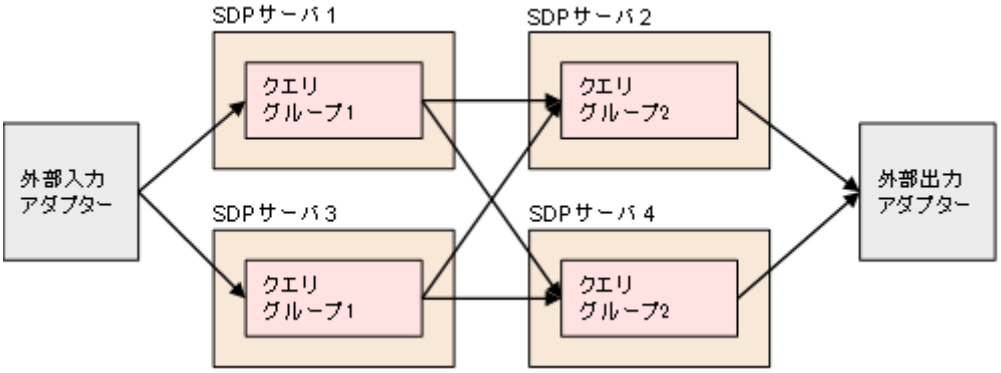
- 内部カスタムアダプター

  hsdpstartinpro コマンドを使用して内部カスタムアダプターを起動する場合，スケールアップ構成であるかどうかに関わらず，内部カスタムアダプターを 1 つだけ起動できます。内部カスタムアダプターが接続するクエリグループの名前には，クエリグループ名のセクションで指定したクエリグループ名を指定してください。

10.1.2 スケールアウト構成

スケールアウト構成では，複数の SDP サーバプロセスのクエリグループを（並列に）分析します。多重化に十分な CPU と，メモリリソースがある場合は，ハイパフォーマンスシステムを構築するのに，スケールアップ構成ではなく，スケールアウト構成を選択してください。

クエリグループを処理するためのスケールアウト構成



説明

図は，クエリグループ 1 および 2 を処理するスケールアウト構成を示しています。クエリグループへのデータは分散して送信されます。

- 使用方法

  同じコーディネーターグループのホスト上で，同じサーバクラスター名を持つ複数の運用ディレクトリを作成します。運用ディレクトリの作成後に，同じクエリグループ名を持つクエリグループを登録して起動します。複数のホスト間にスケールアウト構成を構築することもできます。なお，同じクエリグループ名を持ち並列に稼働するクエリグループへ送信されるデータの負荷分散方式は，それぞれの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に指定します。

- 内部標準アダプター

  スケールアウト構成のクエリグループに接続する内部標準アダプターを起動するには，スケールアウトを構成する各運用ディレクトリで`hsdpstartinpro` コマンドを実行します。

  なお，SDP ブローカー連携によって自動起動するスケールアウト構成 SDP サーバの内部 TCP 入力アダプターについては，スケールアップ構成のすべての運用ディレクトリでアダプターが自動で起動します。

- 内部カスタムアダプター

  スケールアウト構成の場合，内部カスタムアダプターは，スケールアウト構成をとる各運用ディレクトリで`hsdpstartinpro` コマンドを実行し，起動する必要があります。

11 データレプリケーション

この章では，アダプターが同一データを複数の宛先ストリームに送信できるようにする，データレプリケーション機能について説明します。

11.1　データレプリケーションとは

データレプリケーション機能は，外部入力アダプターとカスケーディングアダプターで利用できます。データレプリケーションの方法は，カスケーディングアダプターの負荷分散方式「すべて」とは異なります。データレプリケーションは，同一データを複数の異なる入力ストリームに送信するために使用されます。負荷分散方式「すべて」を設定することで，データパラレル構成で，同一データを並列で実行する入力ストリームに送信できます。

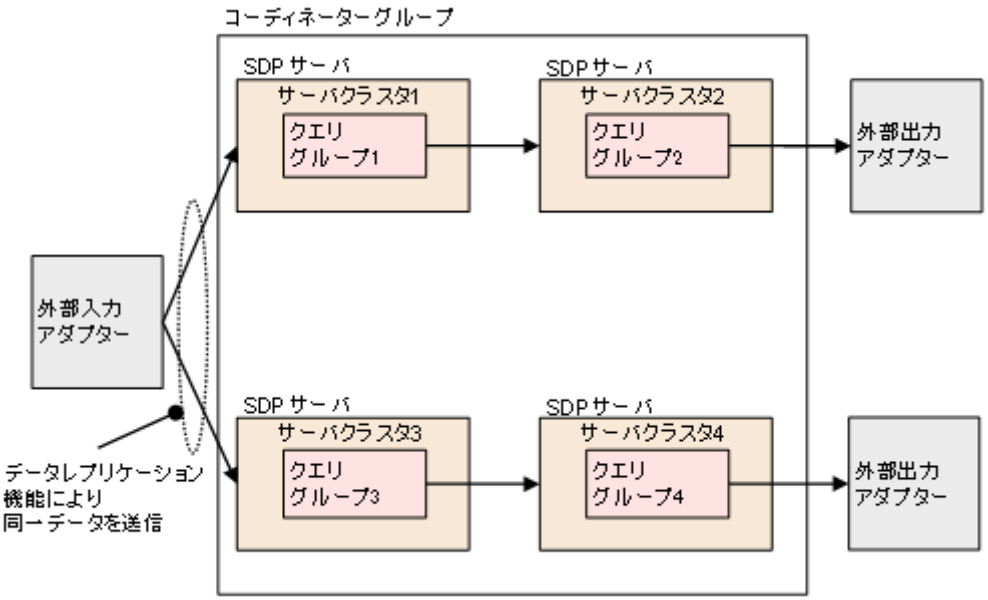
データレプリケーションの使用例については「11.2　データレプリケーションの使用例」，設定については「11.3　データレプリケーションの設定」を参照してください。

負荷分散方式「すべて」については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

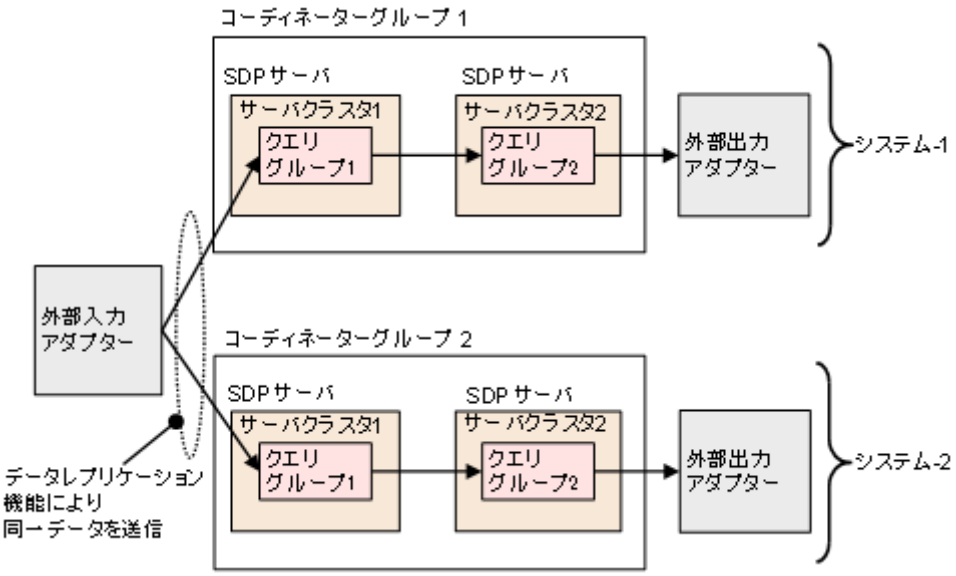
11.2　データレプリケーションの使用例

データレプリケーション機能は，タスクパラレル構成での分析および２つのアクティブシステムを備えた冗長構成（システムの冗長性）の場合に使用されます。

タスクパラレル構成の例



システムの冗長構成の例



データレプリケーション機能を使用した上記の構成例の説明は次のとおりです。

表 11-1　データレプリケーションを使用した構成例

項番	例	説明
1	タスクパラレル構成	タスクパラレル構成では，複数の分析方法によってデータセットを分析します。この分析は，データレプリケーションの機能を使用して実現できます。データレプリケーション機能を使用してタスクパラレル構成を構築できます。
2	システムの冗長構成	2 つのアクティブなシステムがある場合，システムのより高い可用性を実現するために，冗長構成が用いられます。同一のデータがそれぞれのシステムに送信され，それぞれのコーディネーターグループで同一の分析が実行されます。一方のシステムが停止しても，もう一方のシステムで分析が実行されます。

11.3　データレプリケーションの設定

データレプリケーション機能は，各アダプターの定義ファイル内で，複数の宛先ストリームを設定することで使用できます。ただし，2つのアクティブフローがある冗長システムを設定するには，異なるコーディネーターグループを管理するため，入力ストリームを別々の Broker アドレスに割り当てなければなりません。

次の表で，特定のアダプターを使用してデータを複製する場合の定義ファイルの指定方法を示します。

アダプターの種類	定義ファイルの指定方法	例
外部入力アダプター	外部アダプター定義ファイルの hsdp.target.name.*n* プロパティに，同一データの送信先である複数の入力ストリームを（コンマで区切って）指定してください。	target.name.1= /192.168.12.40:20425/qg1/s1,/192.168.12.41:20425/qg1/s1 HSDPAdaptorManager クラスのopenStreamInput()メソッドにtarget.name.1 を指定することで，同一のデータを複数の宛先に送信できます。
カスケーディングアダプター	クエリグループ用プロパティファイルの stream.output.*ストリーム名*.link プロパティに，同一データの送信先である複数の入力ストリームを（コンマで区切って）指定してください。	stream.output.q1.link=qg1/s1,qg2/s1

12 制御タプル

この章では，制御タプル，および制御タプルを使用したストリームデータの制御について説明します。

12.1　制御タプルを使用したストリームデータ制御の仕組み

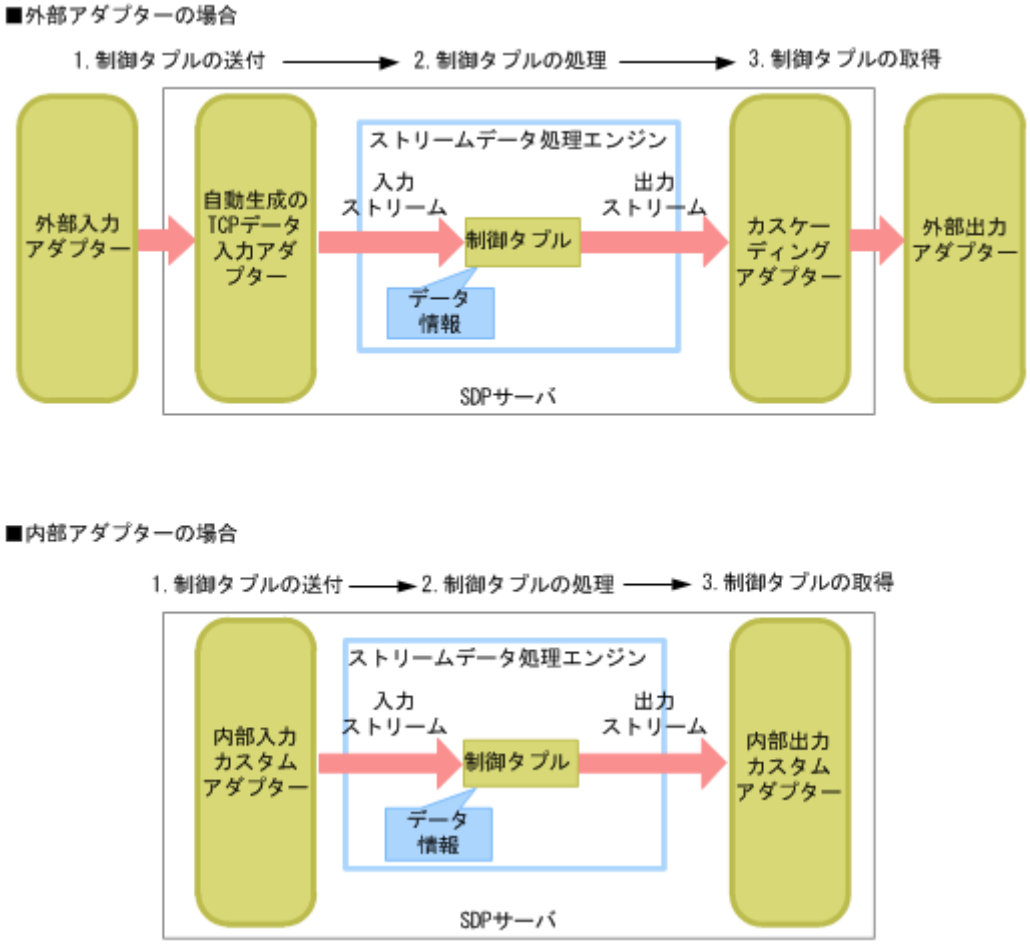
制御タプルとは，ストリームデータ内にチェックポイントとして挿入するタプルです。制御タプルには，識別子や日付などのデータ情報を持たせられます。外部入力アダプターまたは内部入力カスタムアダプターで，制御タプルを使用した次のような処理を実装することで，SDP サーバに入出力するデータを制御できます。

- 入力アダプターから，入力データにチェックポイントを挿入する。
- 出力アダプターで，チェックポイントに到達したことを記録する。
- ストリームデータ処理エンジンや入出力アダプターのプロセスダウン時に，データ再送信およびリカバリ処理をする。

制御タプルのデータの流れ

制御タプルのデータの流れを次の図に示します。

図 12-1　制御タプルのデータの流れ



1. 制御タプルの送付

   API を使用して，外部アダプターまたは内部入力カスタムアダプターから制御タプルを送付します。送付する制御タプルには，0～n 個の識別子（文字列），日付，時刻，シーケンス番号などの任意のデータ

情報を付与できます。API の詳細については，マニュアル『Hitachi Streaming Data Platform アプリケーション開発ガイド』を参照してください。

2. 制御タプルの処理

   入力ストリームからストリームデータ処理エンジンに到達した制御タプルは，何も処理されないで出力ストリームに出力されます。ただし，外部出力アダプターまたは内部出力カスタムアダプターが開始されていない場合は，制御タプルは破棄されます。出力ストリームにデータを出力する際，入力ストリームに入力されたタプル（制御タプルも含む）の順序性は維持されます。

3. 制御タプルの取得

   API を使用して，外部出力アダプターまたは内部出力カスタムアダプターから制御タプルを取得します。なお，複数の出力アダプターがコールバックを使用して出力ストリームに接続する場合は，出力ストリームが複製されるため，それぞれの出力ストリームで制御タプルを取得できます。

制御タプルの時刻制御の方法は，タイムスタンプモードによって異なります。

- サーバモード

  制御タプルが SDP サーバに到着した時刻のタイムスタンプを使用して時刻制御をします。

- データソースモード

  制御タプル内に含まれる時刻データを使用して時刻制御をします。このため，制御タプルを SDP サーバに送付する際に，制御タプルに時刻データを付与する必要があります。

> **メモ**
>
> 制御タプルは，`hsdpstatusshow` コマンドで取得できる稼働情報のうち，アダプターが送受信するレコード数に含まれます。

スケールアップまたはスケールアウト構成時の処理

スケールアップ構成，またはスケールアウト構成の場合，外部入力アダプターから送付された制御タプルは，すべてのスレッドまたはすべてのサーバに送付され，外部出力アダプターまで到達します。

スケールアップ構成，またはスケールアウト構成での制御タプルのデータの流れを次の図に示します。

図 12-2　スケールアップ構成での制御タプルのデータの流れ

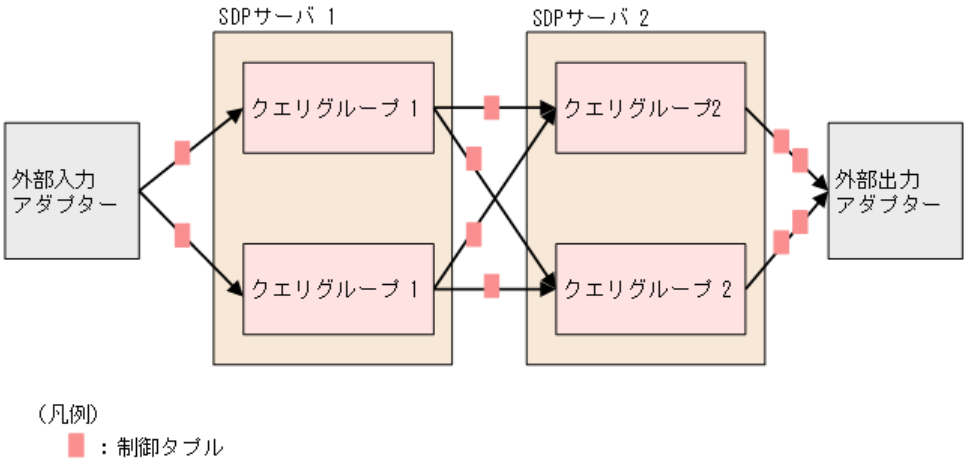
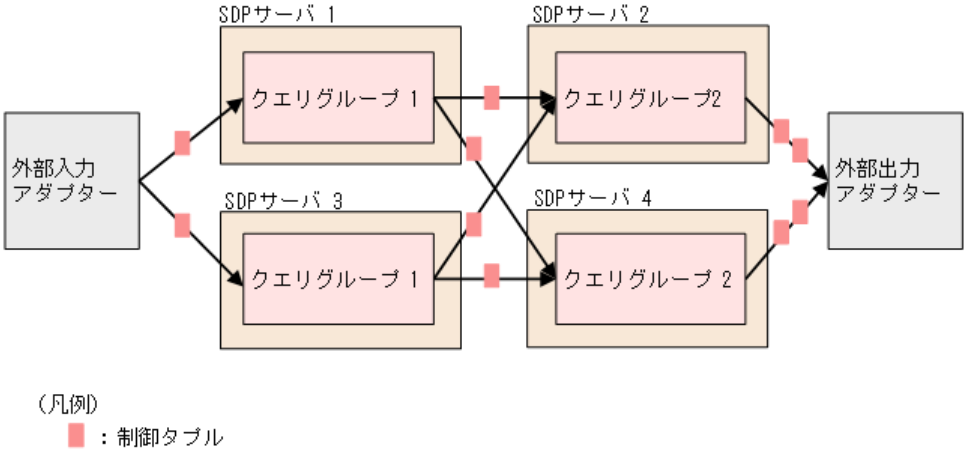


図 12-3　スケールアウト構成での制御タプルのデータの流れ



12.2　制御タプルの使用例

制御タプルを使用したストリームデータ制御の例を示します。

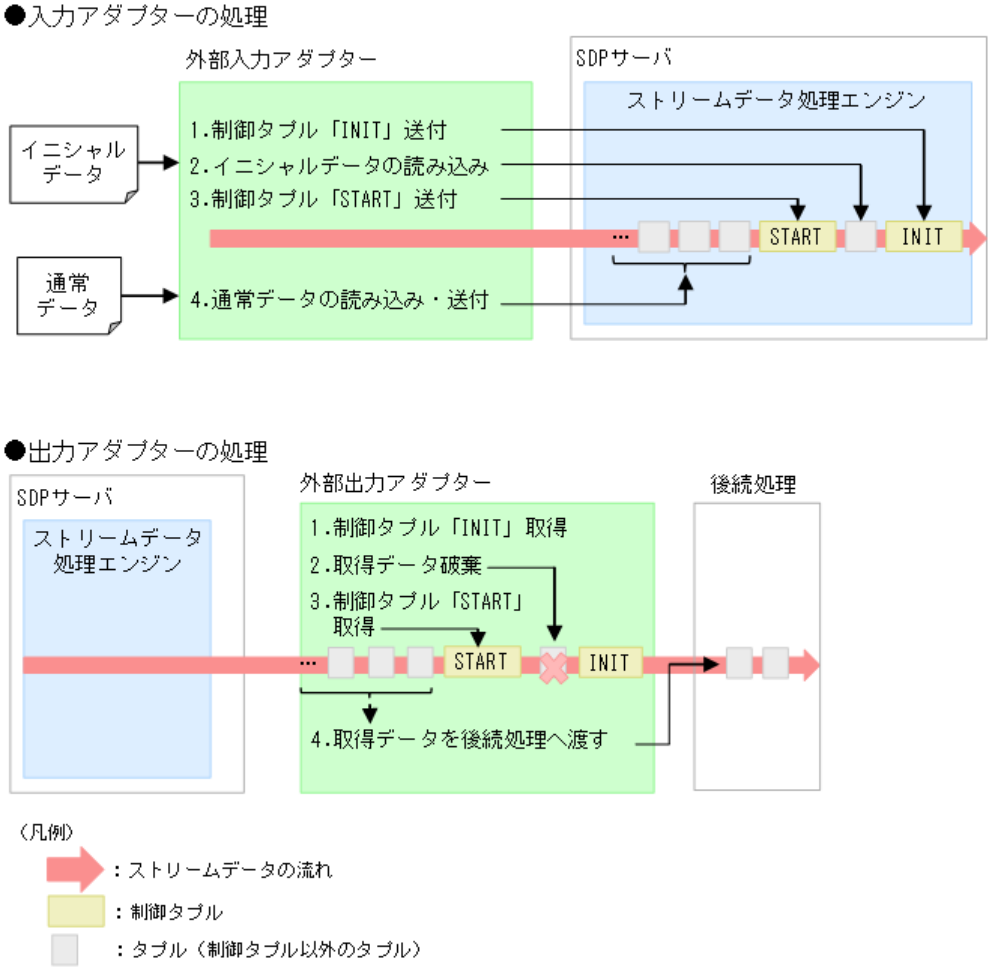
イニシャルデータの切り分け

次の要件の場合に，入力アダプターと出力アダプターで実装する処理について説明します。

要件

初期処理としてストリームデータではないイニシャルデータを読み込む必要がある。ただし，出力アダプターではイニシャルデータを処理しない。

図 12-4　イニシャルデータの切り分けの例



入力アダプターで実装する処理内容

1. 初期処理としてイニシャルデータを読み込む直前に，識別子「INIT」を付与した制御タプルを送付します。
2. イニシャルデータを読み込みます。
3. イニシャルデータの読み込みが完了したら，識別子「START」を付与した制御タプルを送付します。
4. 通常のストリームデータの読み込みを開始し，処理を実施します。

出力アダプターで実装する処理内容

1. 識別子「INIT」の制御タプルを取得します。
2. 次の制御タプルが来るまでの出力ストリームは破棄して，後続の処理を実施しません。
3. 識別子「START」の制御タプルを取得します。
4. これ以降の出力ストリームデータは通常の処理を実施し，後続の処理に引き継ぎます。

13 タプルログ

この章では，タプルログの出力，表示，およびタプルログを使用したクエリの再実行について説明します。

13.1　タプルログの出力と表示

タプルログとは，入力タプルと出力タプルの情報，および API の実行ログ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。

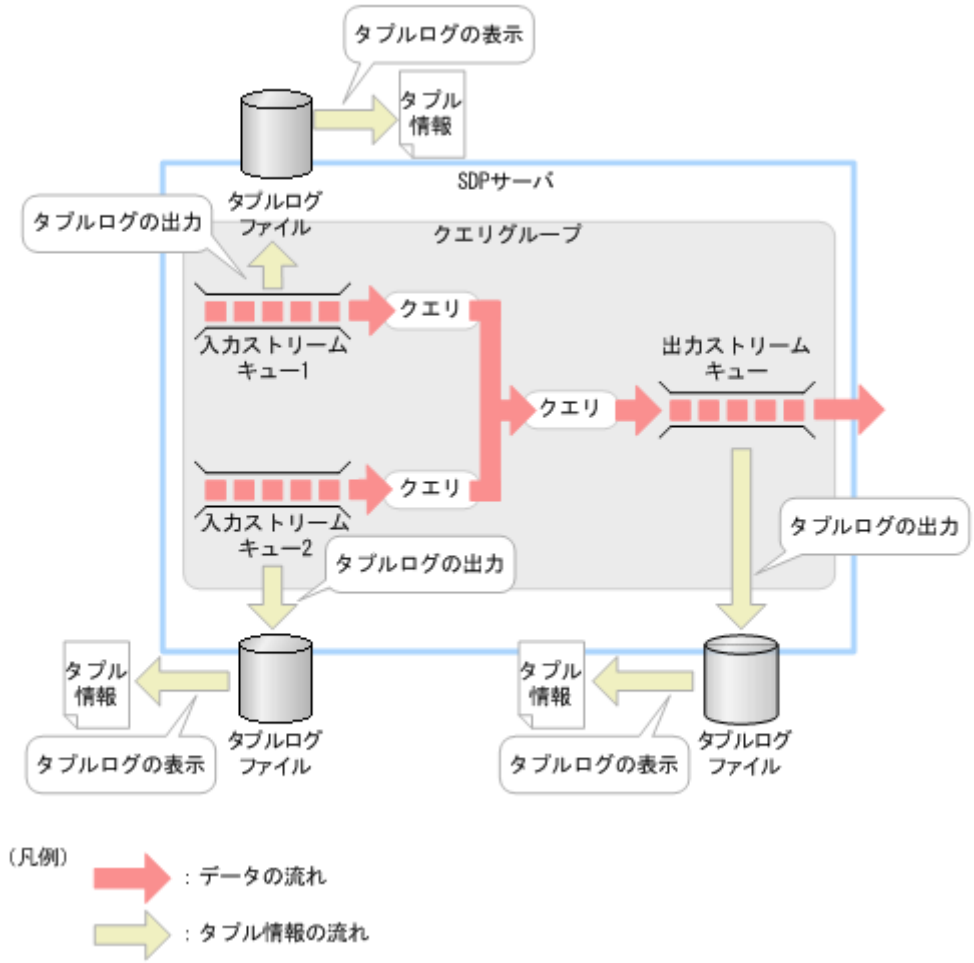
- SDP サーバに到着したタプルの確認
- クエリの再実行，および結果の確認

クエリの再実行については，「13.2　タプルログを使用したクエリの再実行」を参照してください。

なお，タプルログは SDP サーバが Java エンジンで動作しているときだけ出力できます。SDP サーバが acceleration CQL エンジンで動作している場合，タプルログを使用する設定にした場合でも，タプルログは出力されません。

タプルログの出力と表示の流れを次の図に示します。

図 13-1　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

  hsdptplls 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。

  hsdptplls コマンドについては，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

13.1.1 タプルログの出力の設定

タプルログを出力するには，システムコンフィグプロパティファイル（system_config.properties）の engine.useTupleLog パラメーターにtrue を指定します。また，タプルログの出力に関する詳細な設定は，システムコンフィグプロパティファイルの「tpl.」で始まるパラメーターで指定します。

engine.useTupleLog パラメーター，および「tpl.」で始まるパラメーターについては，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』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 13-1 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 13-2 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hsdptplls コマンドで確認してください。

  hsdptplls コマンドについては，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』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

  投入中のタプルを取得できない場合があります。

13.1.2 タプルログファイルの名称

タプルログファイルは，入力タプルと出力タプルの情報，および API の実行ログを出力したバイナリデータのファイルです。タプルログの出力先ディレクトリに，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，hsdpstart コマンドを実行したユーザーの権限に従います。

- タプルログファイル
- 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*

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

  タプルログの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入力，または出力ストリーム名です。
- ファイル通番

  ファイルの通番です。001 からtpl.bufferCount パラメーターの指定値までの 3 桁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までの 2 桁の数字が設定されます。

タプルログの出力先ディレクトリは，*運用ディレクトリ*/trc/tuplelog/です。

13.1.3　タプルログの表示

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

hsdptplls コマンドについては，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

13.2　タプルログを使用したクエリの再実行

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，hsdptplput 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。

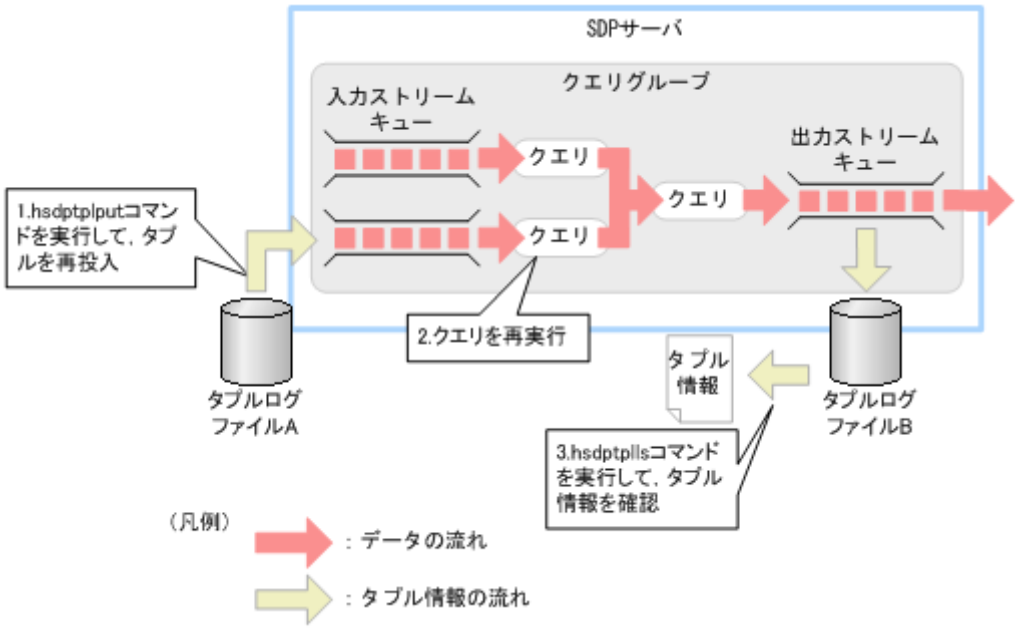
クエリの再実行の結果は，hsdptplput コマンドを使用してタプルログファイルに取得した出力ストリームキューのタプル情報を表示して確認します。

メモ

- 入力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ことがあります。
- hsdptplput 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が満杯になった場合，処理待ちが発生することがあります。

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 13-2　クエリの再実行のデータの流れ



1. hsdptplput コマンドを実行して（タプルログファイル 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. hsdptplplls コマンドを実行して（タプルログファイル B を指定），タプル情報を確認します。

13.2.1　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

> **メモ**
>
> タイムスタンプ調整機能は，最初に投入したタプルのタイムスタンプが基準時刻になります。そのため，データソースモードでは，入力アダプターが最初に投入したタプル以外から再投入した場合，タイムスタンプ調整機能の基準時刻が変わり，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る場合があります。同じ結果を再現したい場合は，入力アダプターが最初に投入したタプルから再投入してください。

タイムスタンプ調整機能については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

13.2.2　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，クエリを再実行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，システムコンフィグプロパティファイル（system_config.properties）を，タプルログ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。

   システムコンフィグプロパティファイルのengine.useTupleLog パラメーターにtrue を指定します。また，tpl.outputTrigger パラメーターにbuffer を指定して，出力ストリームキューでタプルログを取得するようにします。

   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。

   hsdpcql 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。

   hsdpcqlstart 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。

   hsdptplput 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。

   重要

   hsdptplput 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。

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hsdptplls コマンドで表示し，結果を確認します。

hsdpcql コマンド，hsdpcqlstart コマンド，hsdptplput コマンド，またはhsdptplls コマンドについては，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

14 ロガー

この章では，SDP コンポーネントからログファイルにメッセージを出力するために使用される，ロガー機能について説明します。

14.1　ログファイルの生成

ロガー機能は，SDP コンポーネントからログファイルにメッセージ，またはトレースを出力するために使用されます。

説明

ロガーによって出力されるログファイルを次の 2 つの表に示します。

ログファイルの詳細については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

表 14-1　ロガーによって生成されるログファイル（メッセージ）

項番	コンポーネント		生成されるログファイル
1	共通	コマンド	hsdpcommandmessage*N*※1.log
2		ロガー	hsdpservermessage*N*※1.log
3	SDP サーバ		SDPServerMessage*N*※1.log SDPServerCMessage*N*※1.log
4			
5	内部アダプター		ADP_*XXX*※2-AdaptorMessage*N*※1.log ADP_*XXX*※2-AdaptorCMessage*N*※1.log
6	内部カスタムアダプター		CADP_*YYY*※3-CustomAdaptorMessage*N*※1.log
7	外部アダプター		ExAdaptorMessage*N*※1.log
8	SDP ブローカー		BrokerMessage*N*※1.log
9	SDP コーディネーター		CoordinatorMessage*N*※1.log
10	SDP マネージャー		ManagerMessage*N*※1.log
11	hsdpsetup コマンド		hsdpsetup.log
			hsdpsetupmanager.log
12	hsdpexport コマンド		hsdpexport.log
13	hsdpmanager コマンド		hsdpmanagercommandmessage*N*※1.log

注※1

N：メッセージログファイルの面数

注※2

XXX：アダプターグループの名前

注※3

YYY：SDPClientLogger インタフェースのinitialize()メソッドで指定した任意の名前

表 14-2　ロガーによって生成されるログファイル（トレース）

項番	コンポーネント		生成されるログファイル
1	共通	コマンド	hsdpcommandtrace*N*※1.log
2		ロガー	hsdpservertrace*N*※1.log
3	SDP サーバ		SDPServerTrace*N*※1.log SDPServerCTrace*N*※1.log
4			
5	内部アダプター		ADP_*XXX*※2-AdaptorTrace*N*※1.log ADP_*XXX*※2-AdaptorCTrace*N*※1.log
6	内部カスタムアダプター		CADP_*YYY*※3-CustomAdaptorTrace*N*※1.log
7	外部アダプター		ExAdaptorTrace*N*※1.log
8	SDP ブローカー		BrokerTrace*N*※1.log
9	SDP コーディネーター		CoordinatorTrace*N*※1.log
10	SDP マネージャー		ManagerTrace*N*※1.log
11	hsdpmanager コマンド		hsdpmanagercommandtrace*N*※1.log

注※1

N：トレースログファイルの面数

注※2

XXX：アダプターグループの名前

注※3

YYY：SDPClientLogger インタフェースのinitialize()メソッドで指定した任意の名前

15 タイムスタンプ調整

この章では，タプルの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。

15.1　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。

15.2　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。

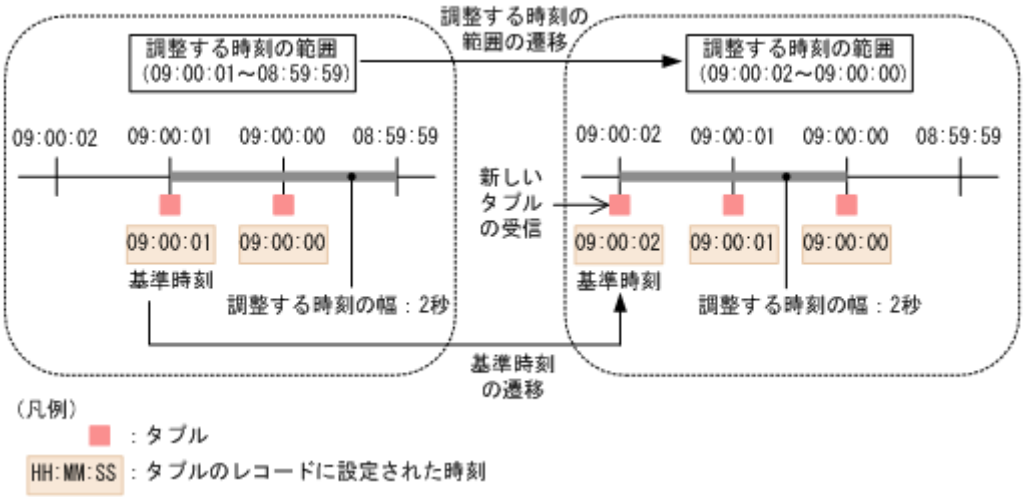
調整する時刻の幅

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，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』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 15-1　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。

15.3　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。

15.3.1　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。

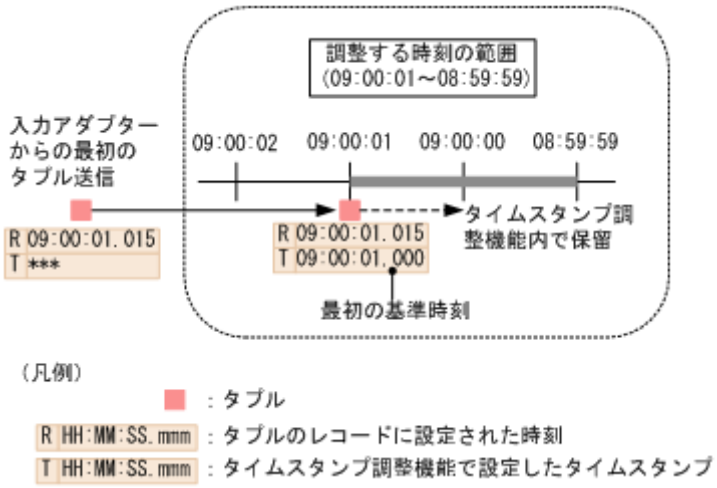
- クエリグループ開始後の最初のタプルの場合
- タプルのレコードに設定された時刻が基準時刻よりも未来の場合
- タプルのレコードに設定された時刻が調整する時刻の範囲内の場合
- タプルのレコードに設定された時刻が調整する時刻範囲より過去の場合

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 15-2　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

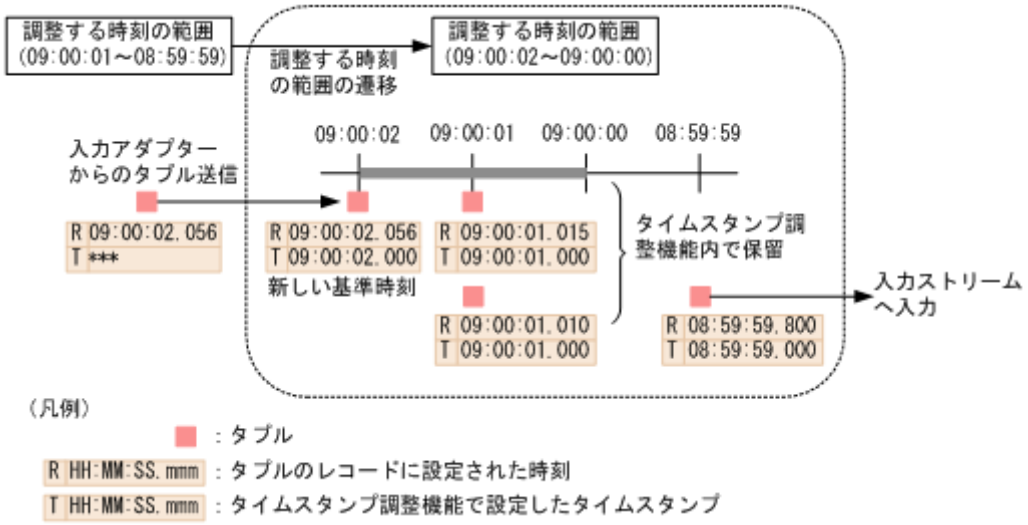
入力アダプターから送信されたタプルのレコードに設定された時刻が基準時刻よりも未来の時刻の場合，次の順序でタプルの時刻を調整します。

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に入力します。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に入力します。

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 15-3　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。

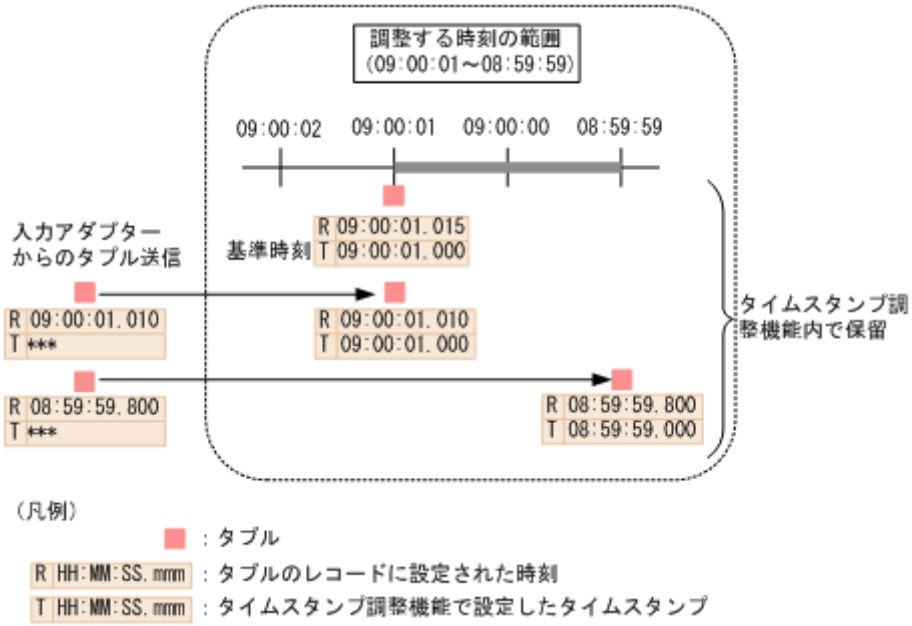
時刻の調整範囲の遷移によって調整範囲から外れる「08:59:59」のタイムスタンプを持つタプルは入力ストリームに入力されます。

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 15-4　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

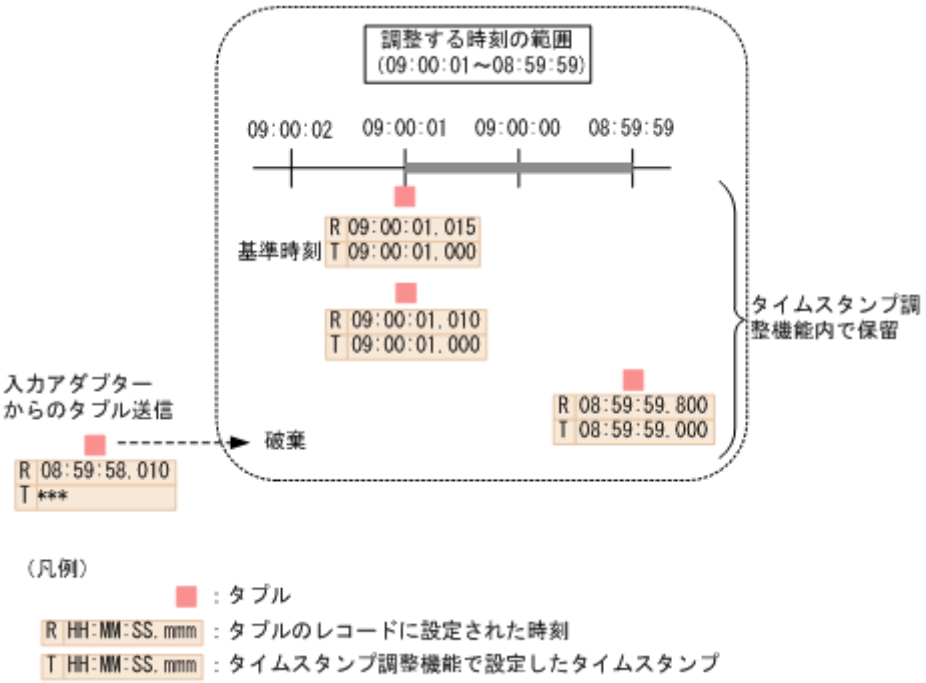


基準時刻が「09:00:01」，調整する時刻範囲が「09:00:01～08:59:59」の場合に，入力アダプターから新しいタプル（レコードに設定された時刻は09:00:01:010）が送信されると，タプルのレコードに設定された時刻が調整する時刻の範囲内のため，タイムスタンプ調整機能内で「09:00:01」のタイムスタンプを持つタプルとして保留されます。さらに，入力アダプターからタプルのレコードに設定された時刻が「08:59:59.800」のタプルが送信された場合も同様に，タイムスタンプ調整機能内で「08:59:59」のタイムスタンプを持つタプルとして保留されます。

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

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 15-5　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。

15.3.2　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。
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 12:15:23:000000000
D	破棄	設定されません。
E	4	2009/03/01 12:15:24:000000000

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。

15.4　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 API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。

15.5　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 15-1　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。

15.6　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，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

15.7　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，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

15.8　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。

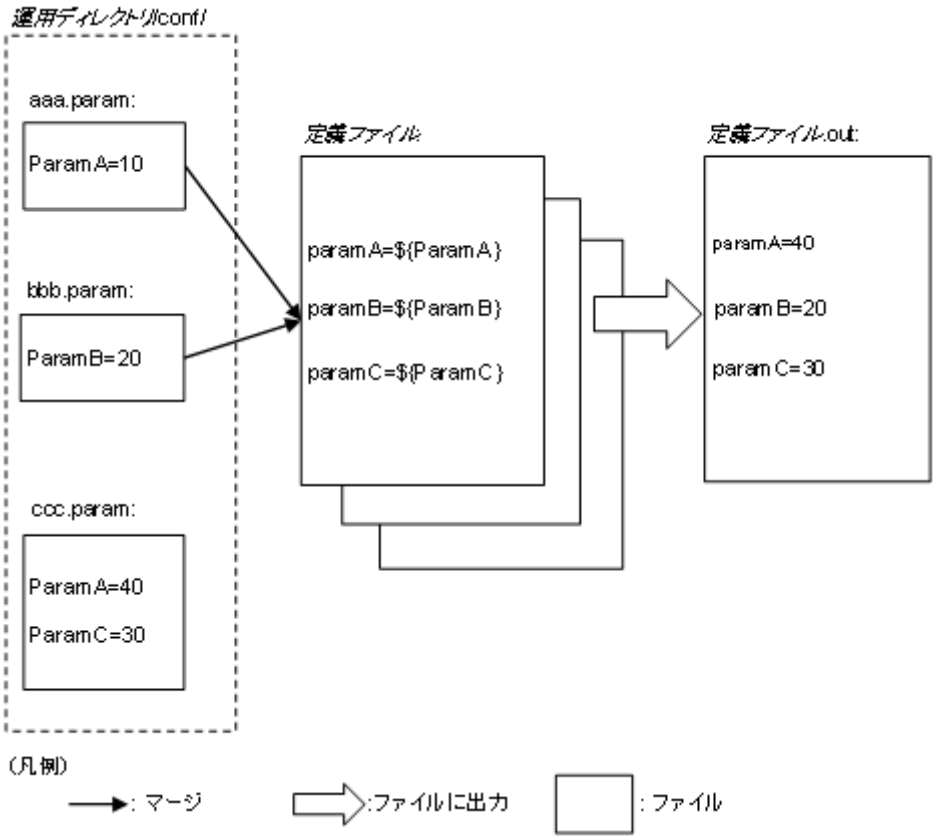
16 定義ファイルへのパラメーター値の設定

この章では，パラメーター値と定義ファイルの関係について説明します。また，クエリ定義ファイルにパラメーター値を設定する例を示します。

16.1　パラメーターファイルと定義ファイルの関係

SDP 定義ファイルの項目のパラメーター値は，必要に応じて設定できます。パラメーターファイルで設定されるパラメーター値（変更する定義や変更しない定義を区別できるようにする）が収集され，これによって定義策定と構成がよりシンプルになります。

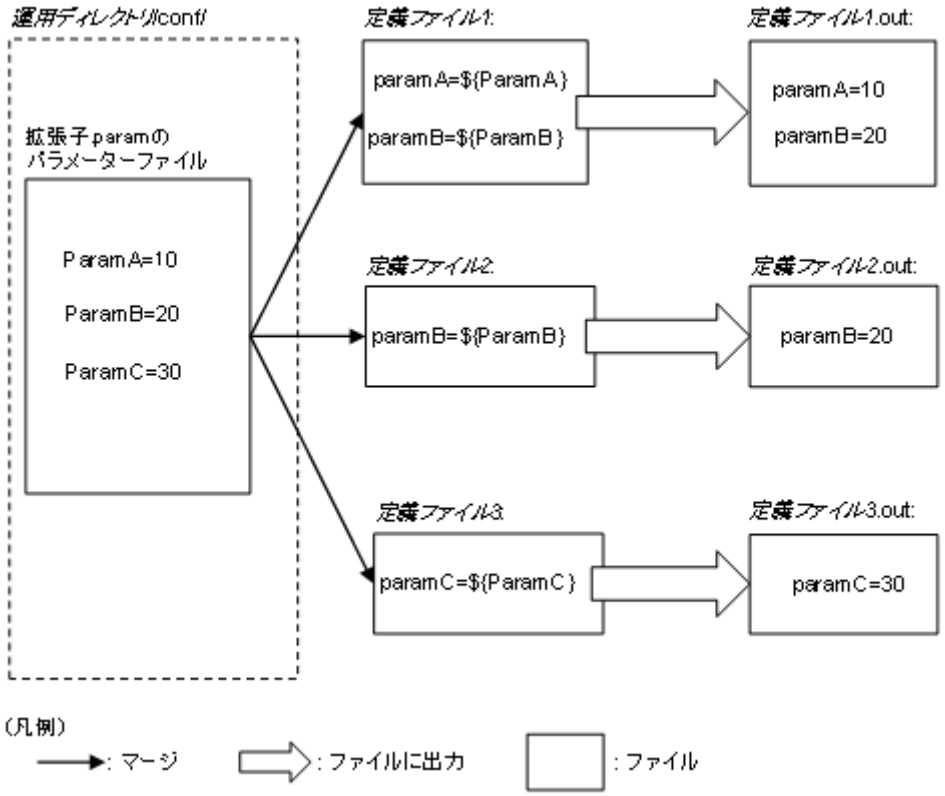
複数のパラメーターファイルの値を定義ファイルにマージする



例は，複数のパラメーターファイルの同じキー名の使用方法を示しています。複数のパラメーターファイルがある場合，SDP は，ASCII コードのファイル名の昇順で，パラメーターファイルを読み込みます。

複数のパラメーターファイルで，同一のキー名が使用されている場合，あとで読み込まれるファイルの値が使用されます。

パラメーターファイルの値を複数の定義ファイルに（分析によって）マージする



説明

SDP 定義ファイルの内容にパラメーター値を設定できます。

パラメーター化が必要な定義ファイルには，パラメーターに代入する値を指定する必要があります。これらの代入値は，キー値形式で，「パラメーターファイル」と呼ばれる別ファイルで指定します。

パラメーター値を設定することで，定義ファイルの変更してはならない部分と，調整しなければならない部分の区別が明確になります。複数の定義ファイルに同一の値を設定するために，パラメーター値を設定して，変更する部分を集約できるようにします。

定義ファイルのファイル名と保存場所は，定義ファイルの指定に従う必要があります。パラメーター値が設定されたら，ファイル名と保存場所は変更できません。拡張子.param のパラメーターファイルは，*運用ディレクトリ*/conf/ディレクトリで作成し保存される必要があります。

パラメーター値を設定できる定義ファイルの読み込み時に，SDP はパラメーターファイルの内容を定義ファイルにマージします。同様に，ファイルのマージ時に，SDP は，設定されたパラメーター値の部分をパラメーターファイルで指定された値で置き換えます。ファイルがマージされると，マージされた定義ファイルの内容に従って，SDP が動作します。マージされた定義ファイルの内容は，別ファイル（定義ファイルのファイル名とファイル拡張子.out を使用）に保存され，定義ファイルと同じディレクトリに出力されます。同じ名前のファイルがある場合は，このファイル（マージされた定義ファイルの内容を含む）が上書きされます。出力ファイルを参照すると，置き換わった結果を確認できます。

SDP が停止されたあとも，拡張子.out のファイルは残ります。このファイルを削除する場合は，SDP が停止したあとで削除してください。また，このファイルの内容は変更しないでください。置き換わった結果を確認するのに，参考として使用できます。

パラメーター値を設定できる定義ファイルの一覧

パラメーター値を設定できる定義ファイルは，以下のとおりです。

> **メモ**
>
> 一覧に示されていない定義ファイルには，パラメーター値を設定できません。このようなファイルがパラメーター値を受け入れると，コマンドの実行時にエラーが発生します。

- 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
- 外部定義関数定義ファイル
- アダプター構成定義ファイル

16.2 クエリ定義ファイルとクエリグループ用プロパティファイルのパラメーター値の設定例

クエリ定義ファイルにパラメーター値を設定する場合は，定義ファイルの名前と保存場所を変更しないでください。名前と保存場所は，各定義ファイルでの指定と一致しなくてはなりません。拡張子.param のパラメーターファイルは，*運用ディレクトリ*/conf/ディレクトリに保存される必要があります。SDP コマンドを使用することで，関連する定義ファイルとすべてのパラメーターファイルが，*運用ディレクトリ*/conf/ディレクトリに読み込まれ，マージされます。マージ後，定義ファイルはマージ前と同じディレクトリに出力され，そのディレクトリで保存されます。出力ファイルは，拡張子.out で保存されます。.out ファイルは，参考用です。コマンドでは，変更された.out ファイルの読み込みや分析を実行しません。定義内容を変更する場合は，定義ファイル（パラメーターが設定されている）とパラメーターファイルを編集してください。

クエリグループ用プロパティファイル（パラメーターあり，なし）およびパラメーターファイル（拡張子.param）の例

定義ファイルとパラメーターファイルの詳細については，マニュアル『Hitachi Streaming Data Platform システム構築ガイド』を参照してください。

クエリグループ用プロパティファイル（パラメーターなし）

```
querygroup.cqlFilePath=/home/user1/wk1/query/q001
stream.input.S1.dispatch.type= hashing
stream.input.S1.dispatch.rule= column11,column12,column13
stream.input.S2.dispatch.type= hashing
stream.input.S2.dispatch.rule= column21,column22
    :
(snip)
```

クエリグループ用プロパティファイル（パラメーターあり）

```
querygroup.cqlFilePath=${cqlFilePath}
stream.input.S1.dispatch.type=${S1_type}
stream.input.S1.dispatch.rule=${S1_rule}
stream.input.S2.dispatch.type=${S2_type}
stream.input.S2.dispatch.rule=${S2_rule}
    :
(snip)
```

パラメーターファイル（拡張子.param）

```
cqlFilePath=/home/user1/wk1/query/q001
S1_type=hashing
S1_rule=column11,column12,column13
S2_type=hashing
S2_rule=column21,column22
```

```
    :
(snip)
```

16.3　アダプターのスキーマ自動解決

アダプターのスキーマ自動解決機能を利用すると，アダプターは接続先のクエリグループとストリームからデータのスキーマ情報を取得し，アダプターで処理されるデータについてスキーマを自動的に解決します。そのため，この機能を利用すると，アダプター構成定義ファイルに明示的にスキーマ情報を指定する必要がありません。

説明

SDP の標準提供アダプターを使用する場合，クエリグループとタプルを入出力するため，アダプター構成定義ファイルのクエリグループの入力または出力データのスキーマを事前に定義しておかなければなりません。ただし，アダプターによって送受信されるデータが，クエリグループの入力および出力データの構造と同じ構造である場合は，この定義を無視できます。

SDP で提供されるテンプレートを変更すると，標準提供アダプターのアダプター構成定義ファイルは，ユーザーによる作成が必要になります。また，アダプターが起動するときに，ユーザーは接続先クエリグループ名およびストリーム名をコマンドで指定しなければなりません。ユーザーは，クエリグループに入力またはクエリグループから出力するデータのスキーマに関する情報を保持しておく必要はありません。

アダプターのスキーマ自動解決機能を使用する場合，1 つのアダプター構成定義ファイルに複数の SDP 標準提供アダプターの定義を含めないでください。複数のアダプターを作成するには，アダプターごとに別々のアダプター構成定義ファイルを作成してください。1 つのアダプター構成定義ファイルに複数のアダプターが含まれる場合，SDP は，アダプター構成定義ファイルに含まれるすべてのアダプターが同じ 1 つのストリームに接続するよう，スキーマ情報を解決します。

TCP データ入力アダプターで，ユーザーがスキーマを記述する場合のアダプター構成定義ファイルと，スキーマが自動的に解決されるアダプター構成定義ファイルの例を，それぞれ次に示します。

TCP データ入力アダプターから入力クエリグループにデータを入力する，クエリ定義ファイルの内容：

```
register stream DATA0(name VARCHAR(10), num BIGINT);
register query FILTER1 ISTREAM(SELECT name FROM DATA0[ROWS 1]);
```

ユーザーによってスキーマが記述された，アダプター構成定義ファイルの内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2016, 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/FileInput
ConnectorDefinition"
    xmlns:docon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/Dashboard
OutputConnectorDefinition"
    xmlns:focon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/FileOutpu
tConnectorDefinition"
```

```
    xmlns:form="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/FormatDefi
nition"
    xmlns:scon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/SendConnec
torDefinition"
    xmlns:rcon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/ReceiveCon
nectorDefinition"
    xmlns:tcpicon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/TcpData
InputConnectorDefinition"
    xmlns:caclcon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/Cascadi
ngClientConnectorDefinition"
>

    <cmn:CommonDefinition>
        <cmn:AdaptorTraceDefinition/>
    </cmn:CommonDefinition>

    <adp:InprocessGroupDefinition name="tcp">

        <adp:InputAdaptorDefinition name="tcp" charCode="UTF-8" lineFeed="LF" language="Java
">

            <cb:InputCBDefinition class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.tcpinput
.TcpDataInputCBImpl" name="Inputer">
                <tcpicon:TCPDataInputConnectorDefinition>
                    <tcpicon:input port="25452" charCode="ASCII">
                        <tcpicon:binary>
                            <tcpicon:data name="NAME" type="STRING" size="10" />
                            <tcpicon:data name="NUM" type="LONG" size="8" />
                        </tcpicon:binary>
                    </tcpicon:input>
                    <tcpicon:output>
                        <tcpicon:record name="RECORD">
                            <tcpicon:fields>
                                <tcpicon:field name="NAME" />
                                <tcpicon:field name="NUM" />
                            </tcpicon:fields>
                        </tcpicon:record>
                    </tcpicon:output>
                </tcpicon:TCPDataInputConnectorDefinition>
            </cb:InputCBDefinition>

            <cb:SendCBDefinition class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.
SendConnectorCBImpl" name="Sender">
                <scon:SendConnectorDefinition>
                    <scon:streamInputs>
                        <scon:streamInput>
                            <scon:record name="RECORD" />
                            <scon:stream name="DATA0" querygroup="Inprocess_QueryGroupTest"
/>
                        </scon:streamInput>
                    </scon:streamInputs>
                </scon:SendConnectorDefinition>
            </cb:SendCBDefinition>
        </adp:InputAdaptorDefinition>

    </adp:InprocessGroupDefinition>
```

```
</root:AdaptorCompositionDefinition>
```

自動スキーマ解決形式のアダプター構成定義ファイルの内容は次のとおりです。

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2016, 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/FileInput
ConnectorDefinition"
    xmlns:docon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/Dashboard
OutputConnectorDefinition"
    xmlns:focon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/FileOutpu
tConnectorDefinition"
    xmlns:form="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/FormatDefi
nition"
    xmlns:scon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/SendConnec
torDefinition"
    xmlns:rcon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/ReceiveCon
nectorDefinition"
    xmlns:tcpicon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/TcpData
InputConnectorDefinition"
    xmlns:dscon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/Distribut
edSendConnectorDefinition"
    xmlns:caclcon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/Cascadi
ngClientConnectorDefinition"
    xmlns:lwicon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/Forwardi
ngInputConnectorDefinition"
    xmlns:lwscon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/Forwardi
ngSendConnectorDefinition"
>

    <cmn:CommonDefinition>
        <cmn:AdaptorTraceDefinition/>
    </cmn:CommonDefinition>

    <adp:InprocessGroupDefinition name="tcp">

        <adp:InputAdaptorDefinition name="tcp" charCode="UTF-8" lineFeed="LF" language="Java
">

            <cb:InputCBDefinition class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.tcpinput
.TcpDataInputCBImpl" name="Inputer">
                <tcpicon:TCPDataInputConnectorDefinition>
                    <tcpicon:input port="25452" charCode="${hsdp_adp_charCode}※">
                        <tcpicon:binary>
                            ${hsdp_adp_tcpBinary}
                        </tcpicon:binary>
                    </tcpicon:input>
                    <tcpicon:output>
                        <tcpicon:record name="RECORD">
                            <tcpicon:fields>
                                ${hsdp_adp_tcpFields}
```

```
                        </tcpicon:fields>
                    </tcpicon:record>
                </tcpicon:output>
            </tcpicon:TCPDataInputConnectorDefinition>
        </cb:InputCBDefinition>

        <cb:SendCBDefinition class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.
SendConnectorCBImpl" name="Sender">
            <scon:SendConnectorDefinition>
                <scon:streamInputs>
                    <scon:streamInput>
                        <scon:record name="RECORD" />
                        <scon:stream name="${hsdp_adp_inputStreamName}" querygroup="${hs
dp_adp_inputQueryGroupName}" />
                    </scon:streamInput>
                </scon:streamInputs>
            </scon:SendConnectorDefinition>
        </cb:SendCBDefinition>
    </adp:InputAdaptorDefinition>

</adp:InprocessGroupDefinition>

</root:AdaptorCompositionDefinition>
```

注※

アダプターが接続するクエリグループ名に対応する，クエリグループ用プロパティファイル内の querygroup.encoding パラメーターに指定した値が埋め込まれます。

これらの例で示したとおり，自動スキーマ解決に使用されるアダプター構成定義ファイルでは，スキーマ情報を記述するセクションのパラメーター値を特定の変数名を使用して設定することで，スキーマ情報が自動的に解決されます。

スキーマ情報に加えて，アダプターによって接続されるクエリグループ名，およびストリーム名のパラメーター値を設定することもできます。アダプター起動時に実際に指定するクエリグループ名とストリーム名が，パラメーター値を設定したクエリグループ名とストリーム名に自動的に置換されます。

用語解説

英字

CQL（Continuous Query Language）

クエリを記述するためのクエリ言語です。

SDP サーバ

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です。

SDP サーバ用定義ファイル

SDP サーバの動作を設定するファイルです。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を設定します。

ア行

アダプター

入出力データとストリームデータ処理エンジン間でのデータの送受信に必要なプログラムです。

アダプターには，製品が提供する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カスタムアダプターがあります。

それぞれのアダプターで，入力データとストリームデータ処理エンジン間で使用するアダプターを入力アダプターといい，出力データとストリームデータ処理エンジン間で使用するアダプターを出力アダプターといいます。

アダプターグループ

入出力アダプターの集合です。標準提供アダプターでは，アダプターグループ単位でアダプターを運用します。

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**といいます。

アダプター構成定義ファイル

標準提供アダプターの動作を設定するファイルです。アダプターグループの構成や，アダプターが使用する入出力コネクターの詳細などを設定します。

インプロセス連携

内部アダプターと SDP サーバは，同一プロセスで動作してデータを送受信します。これをインプロセス連携と呼びます。インプロセス連携のアダプターは，内部アダプターと同義です。

ウィンドウ

ストリームデータに対し，集計・分析をするために設定する範囲です。クエリ中に定義します。

ウィンドウ演算

ウィンドウを指定するための演算です。CQL でクエリ中に記述します。

運用ディレクトリ

HSDP を用いて分析するためのファイルを配置するユーザーディレクトリです。1 つの運用ディレクトリにつき，1 台の SDP サーバを起動させることができます。

オペレーター

ストリームデータ処理の最小処理単位です。1 つのクエリは，1 つ以上のオペレーターで構成されます。

カ行

外部定義関数

ユーザーが Java または C 言語の API などを使用して作成する関数です。

- Java の外部定義関数

  Java の外部定義関数の場合，外部定義関数の処理ロジックを，ユーザーが Java で記述して作成するクラスファイルにメソッドとして実装することで，任意の処理を実行できます。

- C 言語の外部定義関数

  C 言語の外部定義関数の場合，外部定義関数の処理ロジックを，ユーザーが C 言語で記述して作成するソースファイルに関数として実装することで，任意の処理を実行できます。

外部定義関数定義ファイル

外部定義関数の処理を実装したクラス，およびメソッドについて定義するファイルです。

カスタムアダプター

Streaming Data Platform が提供する Java の API を使用してユーザーが作成するアダプターです。

関係演算

ウィンドウ演算によって抽出されたデータの処理を指定する演算です。計算，集計，結合などを指定できます。

共通形式レコード

ストリームデータ処理システムで処理をするためのレコードの形式です。

クエリ

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定義したものです。CQL で記述します。

クエリグループ

あらかじめユーザーが作成するストリームデータの集計・分析シナリオです。入力ストリームキュー（入力ストリーム），出力ストリームキュー（出力ストリーム），およびクエリで構成されます。

組み込み関数

HSDP が提供する関数です。統計関数を提供する組み込み集合関数や，数学関数や文字列関数を提供する組み込みスカラ関数があります。

コールバック

標準提供アダプターの機能を使用した処理の単位です。

コネクター

標準提供アダプター内に定義する Streaming Data Platform と外部を接続するインタフェースです。

Streaming Data Platform への入力に使用するコネクターには，ファイル入力コネクター，TCP データ入力コネクター，および HTTP パケット入力コネクターがあります。Streaming Data Platform からの出力に使用するコネクターには，ファイル出力コネクター，ダッシュボード出力コネクター，カスケーディングクライアントコネクターがあります。

サ行

サーバモード

タプルのタイムスタンプの設定モードの一つです。ストリームデータ処理エンジンにタプルが到着した時点で，Streaming Data Platform が動作するサーバのシステム時刻をタプルに設定します。

時刻解像度機能

RANGE ウィンドウを任意の単位時間に分割（メッシング）し，分割した時間ごとに演算処理をする機能です。

出力形式レコード

ストリームデータの処理結果をファイルに出力する場合のレコード形式です。

出力リレーション

関係演算によって抽出されたタプルの集合です。このタプルの集合が，ストリーム化演算の対象となります。

ストリーム

ストリームデータの型です。入力ストリームキューを通過するストリームデータの型を**入力ストリーム**といい，出力ストリームキューを通過するストリームデータの型を**出力ストリーム**といいます。

ストリーム化演算

出力リレーションのデータの出力方法を指定する演算です。

ストリーム間演算

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を実行し，別のストリームデータに変換する演算です。

ストリーム間演算を使用する場合は，CQL でストリーム間演算関数を定義するほかに，外部定義関数の作成が必要です。

ストリームキュー

ストリームデータの入出力で使用される通路です。ストリームデータ処理エンジンへの入力で使用されるストリームキューを**入力ストリームキュー**といい，ストリームデータ処理エンジンからの出力で使用されるストリームキューを**出力ストリームキュー**といいます。

ストリームデータ

連続して発生する多大な時系列順のデータです。

ストリームデータ処理エンジン

クエリに従い，実際にストリームデータ処理をする部分です。

制御タプル

ストリームデータに挿入して，SDP サーバに入出力するデータを制御するためのタプルです。識別子，日付，時刻，シーケンス番号などのデータ情報を付与できます。

タ行

タイムスタンプ

タプルに設定されたデータの時刻です。

タプル

ストリームデータを構成する，値と時刻（タイムスタンプ）を併せ持ったデータです。

中間リレーション

関係演算の処理中に，WHERE 句で抽出されたタプルの集合です。

データ受信 AP

クライアント AP であり，SDP サーバから出力されるストリームデータに対してイベント処理するアプリケーションです。

データ送信 AP

クライアント AP であり，ストリームデータを SDP サーバへ送信するデータ送信アプリケーションです。

データソースモード

タプルのタイムスタンプの設定モードの一つです。ログファイルなど，入力とするデータソース上に時刻情報がある場合に，その時刻情報をタプルに設定します。

ナ行

入力形式レコード

入力元がファイルの場合に取得したレコードです。

入力リレーション

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，関係演算の対象となります。

ハ行

標準提供アダプター

Streaming Data Platform が提供するアダプターです。入力データの形式としては，ファイル，HTTP パケット，および TCP データがあります。出力データの形式としては，ファイル，TCP データ（カスケーディングアダプター），ダッシュボードがあります。

フィールド

レコードを構成するデータ項目の単位です。

ラ行

リレーション

生存期間を持つレコード群です。CQL でのウィンドウ指定によって，ストリームデータからウィンドウで指定された生存期間を持つリレーションに変換されます。

レコード

ストリームデータ処理で扱われるデータの 1 行の単位です。

レコード構成

複数のフィールド（フィールド名とそれに対応するフィールド値）の組み合わせを表した構成です。

株式会社 日立製作所
〒 100-8280　東京都千代田区丸の内一丁目 6 番 6 号