SYBASE®
An SAP® Company

導入ガイド

Sybase Event Stream Processor 5.0

ドキュメント ID： DC01741-01-0500-01
改訂： 2011 年 12 月

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

第 1 章 Sybase Event Stream Processor の概要

Sybase® Event Stream Processor を使用すると、独自の複雑なイベント処理 (CEP) アプリケーションを作成および実行し、イベント・データのストリームから連続する情報をリアルタイムに取り出せます。

イベント・ストリーム処理と CEP

イベント・ストリーム処理は CEP の 1 つの形態で、状況を把握するために、イベントについての情報をリアルタイムに分析する手法です。大量のイベント・メッセージが発生した場合、状況を全体的に把握することは困難です。イベント・ストリーム処理を使用すると、イベントがストリームに流入した時点で分析でき、新しく出現する脅威と機会が発生した時点で特定できます。Event Stream Processor サーバで、データのフィルタ処理、集約、要約が行われるので、より完全で時宜を得た情報に基づいてより良い意思決定ができるようになります。

Event Stream Processor はエンドユーザ・アプリケーションではなく、単純なプロジェクトと複雑なプロジェクトの両方の開発および展開を簡易化するツールを提供する実現技術です。これらのプロジェクトを展開する、高度にスケーラブルなランタイム環境を提供します。

開発プラットフォームとしての Event Stream Processor

Event Stream Processor は、CEP プロジェクトを開発するためのプラットフォームとして、イベントの処理方法と解析方法を定義するための高いレベルのツールを提供します。開発者は、ビジュアル指向またはテキスト指向のオーサリング環境で作業できます。受信イベントに適用されるロジックを定義して、以下を実行できます。

- 複数のソースからのデータを組み合わせて、より豊富でより完全な情報を含む、派生イベント・ストリームを生成する。
- 価値の付加された情報を計算して、迅速な意思決定を可能にする。
- 特定の状態またはパターンを監視して、瞬時対応を可能にする。
- 要約データ、統計量、傾向情報などの高いレベルの情報を生成し、多くの個々のイベントの全体像または最終的な影響を把握する。
- 受信データの複雑な分析に基づいて、主要な運用値を連続的に計算する。
- 生データおよび結果データを収集して履歴データベースに格納し、履歴分析および法令遵守に活用する。

Event Stream Processor のランタイム環境

Event Stream Processor はイベント駆動型アーキテクチャ (EDA) のエンジンとして、イベントの吸収、集約、相互の関連付け、分析ができ、応答をトリガできる高い

レベルの新しいイベントと、ビジネスの現状を示す高いレベルの情報を生成できます。 Event Stream Processor では以下のことが行われます。

- 到着したデータを連続して処理する。
- ディスクに格納される前にデータを処理する。これによって、非常に高速なスループットと少ない遅延時間が達成され、より完全で時宜を得た情報に基づくより良い意思決定が可能になります。
- ビジネス・ロジックをデータ管理から分離する。これによって、ビジネス・ロジックの保守が容易になり、総所有コストが削減されます。
- エンタープライズ・クラスのスケーラビリティ、信頼性、セキュリティを提供する。

イベント・ストリーム

ビジネス・イベントは、発生した実際のビジネス・イベントに関する情報を含むメッセージです。多くのビジネス・システムでは、何かが発生すると、イベントが生成されます。

次は、イベント・メッセージのストリームとして送信されることが多いビジネス・イベントの例です。

- 取引イベントと相場イベントを送信する金融市場データ・フィード
- RFID (Radio Frequency Identification System) タグを近くで検知したことを示すイベントを送信する RFID センサ
- ユーザが Web サイトのリンク、ボタン、またはコントロールをクリックするたびにメッセージ (クリック イベント) を送信するクリック・ストリーム
- レコードがデータベースに追加されたりデータベース内のレコードが更新されたりするたびに発生するトランザクション・イベント

多くのアプリケーションは、リアルタイムでイベントを生成するように設計されており、通常、イベントをメッセージ・バスに発行します。このように設計されていないアプリケーションは、Sybase® Replication Server® などのツールを使用してイベントに対応可能です。これらのツールは、トランザクション・ログをモニタし、アプリケーション・データベースの更新に基づいてイベント・ストリームをリアルタイムで生成できます。

Event Stream Processor とデータベースの比較

Sybase Event Stream Processor は従来のデータベースを補完して、連続した、イベント駆動型のデータ分析が必要な新しい種類の問題を解決するのを支援します。

Event Stream Processor は、データベースの代替となるものではありません。データベースは、静的データの格納とクエリ、信頼性の高い方法でのトランザクション

処理に優れています。ただし、データベースは、高速のデータ・ストリームの連続的な分析には効果的ではありません。

- 従来のデータベースでは、すべてのデータをディスク上に格納した後に、処理を開始する必要があります。
- データベースは、事前に登録された継続クエリを使用しません。データベースのクエリは、「1 回かぎり」のクエリです。1 秒間に質問を 10 回行うには、クエリを 1 秒間に 10 回発行する必要があります。このモデルは、多くのクエリを連続して評価する必要がある場合には機能しません。
- データベースは、段階的な処理を使用しません。Event Stream Processor は、データが到着するごとに、クエリを段階的に評価できます。

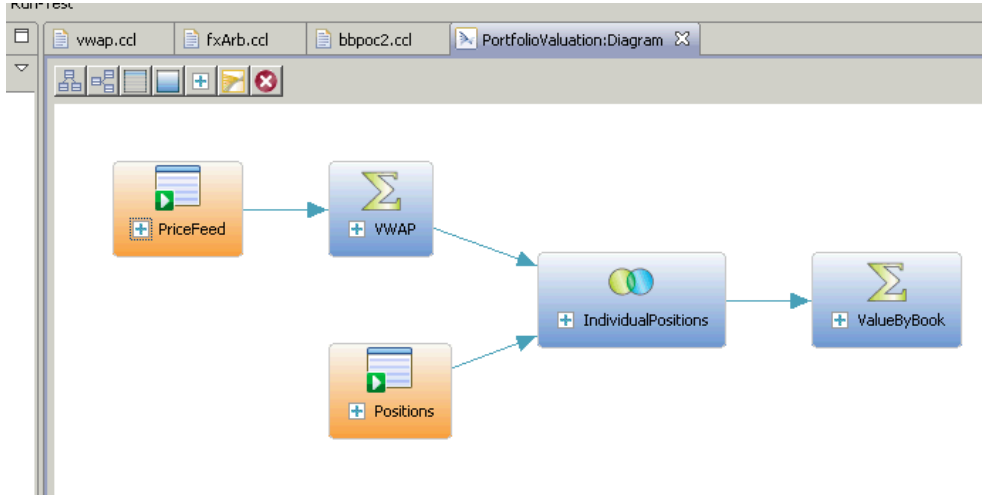
Event Stream Processor は、インメモリ・データベースではありません。インメモリ・データベースは、要求される処理速度を達成するために、メモリ内で動作し、すべてのデータをメモリに保持します。Event Stream Processor は、インメモリ・データベースにいくつかの点で類似していますが、インメモリ・データベースと異なり、オンデマンド・クエリを効果的に処理するように設計されており、連続するイベント駆動型クエリに最適化されたデータフロー・アーキテクチャを使用します。

データフロー・プログラミング

データフロー・プログラミングでは、一連のイベント・ストリームとそれらの間の接続を定義し、データがソースから出力に流れるときの操作をデータに適用します。

データフロー・プログラミングは、複雑になる可能性のある計算を、1 つの操作から次の操作に流れるデータに伴う一連の操作に分割します。 この方法では、各操作がイベント駆動型であり、独立して適用されるので、スケーラビリティと潜在的な並列化がもたらされます。 各操作は異なるスレッドで実行し、他の操作から受信したイベントのみを処理します。操作間で他の調整は必要ありません。

図 1 : データフロー・プログラミング



ESP Projects：アダプタ、ストリーム、ウィンドウ、継続クエリ

ESP プロジェクトは、イベント・ストリーム、他の必要なデータソース、結果を生成するために受信イベント・データに適用されるビジネス・ロジックのセットを定義します。

プロジェクトは、最も基本的なレベルで、以下で構成されます。

- **入力ストリームと入力ウィンドウ** – ここから、入力データがプロジェクトに流れ込みます。入力ストリームは、イベント駆動型を基本として受信イベント・データを受信でき、1 回ロードされるか定期的にリフレッシュされるデータの静的セットまたは疑似静的セットも受信できます。
- **アダプタ** – 入力ストリームまたは入力ウィンドウをデータソースに接続します。 Sybase Event Stream Processor には、多くの組み込みアダプタと、カスタム・アダプタを作成するのに使用できる SDK が同梱されています。アダプタは、出力ストリームまたは出力ウィンドウを送信先に接続することもできます。
- **派生ストリームと派生ウィンドウ** – 1 つ以上のストリームまたはウィンドウからデータを取得し、継続クエリを適用して新しいストリームまたはウィンドウを生成します。

ESP プロジェクトからの結果の取得

Event Stream Processor では、実行中のプロジェクトからの出力を以下の 4 つの方法で取得できます。

- アプリケーションは、プロジェクトの作成時にストリームにアタッチされた内部出力アダプタからの情報を自動的に受信します。
- アプリケーションは、外部サブスクライバの手段によってデータ・ストリームにサブスクライブできます。外部サブスクライバは、製品と共に提供されるサブスクリプション API を使用して作成できます。
- ユーザは、実行中のプロジェクトを再設定することなく、そのプロジェクト内のストリームにバインド (接続) する新しいプロジェクトを起動できます。
- ユーザは、esp_query ツールを使用してオンデマンド・クエリを実行し、実行中の ESP プロジェクト内の出力ウィンドウにクエリできます。詳細については、『ユーティリティ・ガイド』を参照してください。

ストリームとウィンドウ

ストリームとウィンドウの両方が、イベントを処理します。これらの違いは、ウィンドウでは状態は維持されてデータの保持と格納が行えますが、ストリームはステートレスでデータの保持と格納が行えないことです。

ストリームは、ストリームにアタッチされている継続クエリに従って、受信イベントを処理し、出力イベントを生成しますが、データは保持しません。

それとは対照的に、ウィンドウには、受信イベントがローの追加、既存のローの更新、またはローの削除を行うことができるテーブルがあります。ウィンドウのサイズを、時間または記録されるイベントの数に基づいて設定できます。たとえば、過去 20 分間のすべてのイベント、または最新の 1,000 イベントを保持するように、ウィンドウを設定できます。すべてのイベントを保持するようにウィンドウを設定することもできます。この場合、受信イベント・ストリームは自己管理を行い、ウィンドウへのローの挿入とウィンドウからのローの削除の両方を行うイベントを含む必要があります。こうすることで、ウィンドウが無制限に大きくなるのを防げます。

入力、出力、ローカルのストリームとウィンドウ

ストリームとウィンドウは、入力、出力、またはローカルとして指定できます。入力ストリームは、データが外部ソースからアダプタを介してプロジェクトに入力されるポイントです。 プロジェクトには、任意の数の入力ストリームを設定できます。入力ストリームには、継続クエリをアタッチできません。フィルタは定義できます。

ローカルと出力のストリートとウィンドウは、アダプタからではなく、他のストリームまたはウィンドウから入力を取得し、継続クエリを適用して出力を生成します。ローカル・ストリームとローカル・ウィンドウは、外部のサブスクライバから隠蔽されていることを除いて、それぞれ出力ストリームと出力ウィンドウに同じです。このため、サブスクライバがサブスクライブ先となるストリームまたはウィンドウを選択する場合に、出力ストリームと出力ウィンドウのみが利用できます。

注意： ビジュアル・オーサリングのパレットには、ローカルと出力のストリームが派生ストリームとして示され、ローカルと出力のウィンドウが派生ウィンドウとして示されます。

参照：
- *手動による入力ウィンドウの追加* (28 ページ)
- *単純な継続クエリの追加* (29 ページ)

スキーマ

ストリームまたはウィンドウには、それぞれにスキーマがあります。スキーマは、ストリームまたはウィンドウによって生成されるイベント内のカラムを定義します。

各カラムには、名前とデータ型の属性があります。単一のストリームまたはウィンドウから出力されるすべてのイベントは、同じカラムのセットを持ちます。例を示します。

- RFID リーダから受信される、RFIDRaw と呼ばれる入力ストリームには、文字列データを含む ReaderID と TagID のカラムが含まれることがあります。
- 証券取引所から受信される、Trades と呼ばれる入力ストリームには、Symbol (文字列)、Volume (整数値)、Price (浮動小数点値)、Time (日時値) のカラムが含まれることがあります。

参照：
- *スキーマ検出* (26 ページ)
- *スキーマの検出* (26 ページ)

挿入、更新、削除

オペレーション・コード (opcode) によって、挿入、更新、削除のイベントがウィンドウと関連付けられます。オペレーション・コードは、これらのイベントを自動的に適用することによって、開発を簡略化し、パフォーマンスを向上させます。

Event Stream Processor の多くのユース・ケースでは、イベントは相互に独立しています。各イベントは、発生した事柄に関する情報を伝達します。これらの場合、イベントのストリームは、一連の独立したイベントで構成されます。このタイプのイベント・ストリームでウィンドウを定義する場合、受信イベントがそれぞれウィンドウに挿入されます。ウィンドウをテーブルと考えるならば、新規イベントがウィンドウに新しいローとして追加されます。

その他のユース・ケースでは、イベントは以前のイベントに関する新規の情報を伝達します。ESP サーバでは、受信イベントがビューを継続的に更新するため、

情報セットのビューを最新の状態に維持する必要があります。2 つの一般的な例を挙げるならば、資本市場における証券の注文帳簿、または充当システムにおける未出荷注文です。両方のアプリケーションで、受信イベントが以下を実行する必要があることを示す場合があります。

- 注文を未出荷注文のセットに追加する。
- 既存の未出荷注文のステータスを更新する。
- キャンセルされた注文または充当された注文を未出荷注文のセットから削除する。

受信イベントによって更新された情報セットを処理するために、Event Stream Processor は受信イベントに関連付けられた挿入、更新、削除の操作を認識します。イベントに opcode (イベントが挿入イベント、更新イベント、または削除イベントのいずれであるかを示す特別なフィールド) を含むタグを追加できます。マッチング・キーを使用して既存のレコードを更新するか、または新規レコードを挿入するアップサート opcode もあります。

入力ウィンドウは、イベントが到着したときに、挿入、更新、削除のイベントをウィンドウのデータに直接適用します。挿入、更新、削除はクエリ・グラフ、つまりすべての下流となるストリーム派生ウィンドウで送信されます。イベントが入力ウィンドウのレコードを更新または削除すると、その操作は自動的にすべての下流となるストリーム派生ウィンドウに適用されます。更新と削除のこのネイティブ処理によって、高いパフォーマンスと簡略化を実現できます。受信イベントを調べ、その受信イベントをウィンドウに適用する方法を決定するために、ユーザが手動でロジックを定義する必要はありません。

製品のコンポーネント

Event Stream Processor には、データのストリームを処理し、相互に関連付けるためのサーバ・コンポーネント、サーバで実行するアプリケーションの開発、テスト、起動ができるスタジオ環境、および管理ツールが同梱されています。

コンポーネントとして以下があります。

- **ESP サーバ –** データのストリームを実行時に処理し、相互に関連付けるソフトウェア。Event Stream Processor は 1 秒間で数十万件のメッセージを処理し、分析できます。クラスタリングによって、ESP サーバのスケールアウト・サポートが提供されます。サーバ・クラスタを使用すると、複数のプロジェクトを同時に実行できます。また、高い可用性とフェールオーバ機能が達成され、クラスタ接続を管理するためのセキュリティとサポートを一元的に適用できます。
- **ESP スタジオ –** ESP プロジェクトの作成、変更、テストができる統合開発環境です。

- **CCL コンパイラ** – ESP サーバで処理されるようにするためにプロジェクトを翻訳し、最適化するコンパイラです。ESP スタジオまたはコマンド・ラインから起動できます。
- **入力アダプタと出力アダプタ** – Event Stream Processor とデータソースとの間の接続と、ESP サーバと Event Stream Processor からの出力を受け取るコンシューマとの間の接続を確立するコンポーネントです。
- **統合 SDK** – カスタム関数ライブラリを統合し、ライブ・プロジェクトの管理と監視を行うために、カスタム・アダプタを C/C++、Java、.NET で作成するための一連の API です。
- **ユーティリティ** – 多くの管理機能、プロジェクト開発、発行とサブスクリプション、他の機能にコマンド・ラインからアクセスできるようにする一連の実行プログラムです。

入力アダプタと出力アダプタ

Event Stream Processor は、入力アダプタを使用して動的外部ソースと静的外部ソースからメッセージを受信し、出力アダプタを使用して動的外部送信先と静的外部送信先にメッセージを送信します。

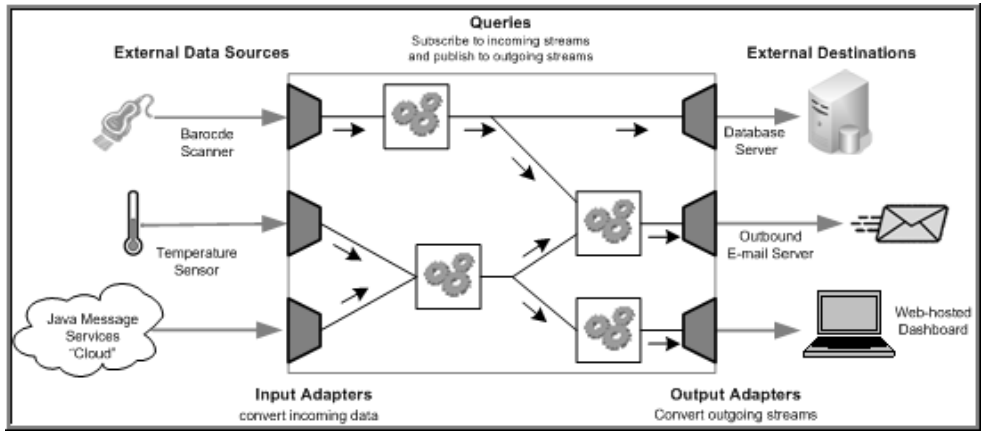
外部ソースまたは外部送信先には、以下があります。

- データ・フィード
- センサ・デバイス
- メッセージング・システム
- RFID (Radio frequency identification) リーダ
- 電子メール・サーバ
- リレーショナル・データベース

入力アダプタは外部データソースに接続し、外部ソースからの受信メッセージを、ESP サーバによって受け付けられるフォーマットに変換します。出力アダプタは、Event Stream Processor によって処理されたローを外部送信先と互換性のあるメッセージ・フォーマットに変換し、それらのメッセージを下流のストリームに送信します。

以下の図は、温度センサ、バー・コード・スキャナ、JMS (Java Message Service) クラウドからのメッセージを、Event Stream Processor と互換性のあるフォーマットに変換する、一連の入力アダプタを示しています。データが Event Stream Processor 内のさまざまなクエリを使用して処理されると、出力アダプタが生成されたローを外部のデータベース・サーバ、電子メール・サーバ、Web サービス・ダッシュボードに送信されるアップデートに変換します。

図 2 : Event Stream Processor 内のアダプタ



Event Stream Processor に同梱されているアダプタの完全なリストについては、『アダプタ・ガイド』を参照してください。

参照：

- *入力アダプタの追加* (24 ページ)

カスタム・アダプタ

Event Stream Processor で提供されるアダプタ以外に、独自のアダプタを作成してサーバに統合できます。標準のアダプタが管理できないさまざまな外部要件を処理するように、アダプタを設計できます。

Event Stream Processor ではさまざまな SDK が用意されており、以下を含む多くのプログラム言語でアダプタを作成できます。

- C
- C++
- Java
- .NET (C#、Visual Basic など)

カスタム・アダプタを作成する方法の詳細については、『アダプタ・ガイド』を参照してください。これらの SDK でサポートされているバージョンについては、『インストール・ガイド』を参照してください。

オーサリング手法

Event Stream Processor スタジオは、ビジュアル・オーサリング環境とテキスト・オーサリング環境を提供します。

ビジュアル・オーサリング環境では、グラフィック・ツールを使用してプロジェクトを開発し、ストリームとウィンドウの定義、それらの接続、入力アダプタと出力アダプタの統合、さまざまな単純なクエリの作成が行えます。

テキスト・オーサリング環境では、任意のテキスト・エディタ内と同様に、CCL (Continuous Computation Language) でプロジェクトを開発できます。データのストリームとウィンドウの作成、クエリの開発、階層モジュールとプロジェクト内でのそれらの編成が行えます。

ビジュアル・エディタと CCL エディタは、いつでも簡単に切り替えられます。一方のエディタで加えた変更は、もう一方のエディタに反映されます。プロジェクトをスタジオ内でコンパイルすることもできます。

ビジュアル・オーサリングとテキスト・オーサリングのコンポーネントに加えて、スタジオには、サンプル・プロジェクトで作業するための環境や、さまざまなデバッグ・ツールを使用してアプリケーションを実行したり、テストしたりするための環境が用意されています。スタジオではまた、プロジェクトのアクティビティの記録と再生、ファイルからのデータのアップロード、手動による入力レコードの作成、サーバへのコマンドの発行、サーバに対するアドホック・クエリの実行が行えます。

コマンド・ラインから作業する場合は、**esp_server**、**esp_client**、**esp_compiler** のコマンドを使用してプロジェクトの開発と実行が行えます。Event Stream Processor のユーティリティの全一覧は、『ユーティリティ・ガイド』を参照してください。

Continuous Computation Language

CCL (Continuous Computation Language) は、Event Stream Processor の主要なイベント処理言語です。ESP プロジェクトは、CCL で定義されます。

CCL は SQL (Structured Query Language) を基本としており、イベント・ストリーム処理用に変更されています。

CCL は高度なデータ選択能力と計算能力をサポートし、以下の機能を提供します。データのグループ化、集約、ジョイン。さらに、CCL には、データ・ストリーム上のウィンドウや、パターンとイベントの一致処理などのリアルタイム連続処理時のデータ操作に必要な機能も用意されています。

CCL を特徴付ける重要な機能は、動的データを連続的に処理する能力です。SQL クエリは通常、データベース・サーバに発行されるごとに 1 回のみ実行され、ユーザまたはアプリケーションがクエリの再実行を必要とするごとに再発行される必要があります。それとは対照的に、CCL クエリは連続しています。プロジェクト内で CCL クエリを定義すると、連続実行として登録され、いつまでもアクティブな状態に維持されます。プロジェクトを ESP サーバで実行すると、登録されているクエリが、そのデータソースの 1 つからデータが到着するごとに実行されます。

CCL では SQL 構文を使用して継続クエリを定義しますが、ESP サーバは SQL クエリ・エンジンを使用しません。その代わりに、CCL を効率の高いバイト・コードにコンパイルします。このコードは、ESP サーバによってデータフロー・アーキテクチャ内の継続クエリを構築するために使用されます。

CCL クエリは、CCL コンパイラによって実行可能な形式に変換されます。一般的に、コンパイルは Event Stream Processor スタジオ内で実行されますが、コマンド・ラインから CCL コンパイラを呼び出しても実行できます。

SPLASH

Stream Processing LAnguage SHell (SPLASH) は、CCL に拡張性をもたらすスクリプト言語で、標準の SQL では提供されないカスタム演算子やカスタム関数の作成を可能にします。

SPLASH スクリプトを CCL に埋め込めるので柔軟性が非常に高まり、それを CCL エディタ内で行うことができるので、ユーザの生産性が大きく向上します。SPLASH を使用すると、手続き型のロジックを使用して複雑な計算を定義できるため、リレーションのパラダイムを使用するより容易に定義を行うことができます。

SPLASH は簡易なスクリプト言語で、他の値から値を計算するために使用される式、変数、ループ構成体で構成されます。また、関数内で命令を編成する機能もあります。SPLASH の構文は、C と Java に似ており、比較的小さなプログラミング問題を解決する言語 (AWK や Perl など) との類似性も持ち合わせています。

第 2 章 ESP スタジオでのプロジェクトの探索

ESP スタジオでサンプル・プロジェクトを使用して、プロジェクトの構造を理解します。また、パースペクティブとビューの移動方法について習得します。

スタジオで操作してみて、各パースペクティブで何ができるかを習得することから始めます。 次に、サンプル・プロジェクトを使用して、いくつかのプロジェクト構造や図の例をビジュアル・エディタで観察します。

注意： 『導入ガイド』に記載されているタスクや概念の詳細については、『スタジオ・ユーザーズ・ガイド』を参照してください。

1. デスクトップにある [Sybase ESP Studio] ショートカットをダブルクリックして、スタジオを起動します。
2. (オプション) [Welcome] 画面で、移動するためのボタンを使用するか、[Welcome] 画面タブを閉じます。
   - [Product Overview] または [Getting Started] をクリックすると、ヘルプが開きます。
   - [Learning] をクリックすると、スタジオが開き、[Learning] パースペクティブがアクティブになります。
   - [Studio] をクリックすると、スタジオが開き、[Authoring] パースペクティブがアクティブになります。

スタジオのワークスペースの基本

スタジオのワークスペースでは、さまざまなパースペクティブとビューを使用して、用例の実行、プロジェクトの作成と編集、実行中の Event Stream Processor サーバでのプロジェクトの実行とテストが行えます。

デフォルトでは、すべてのパースペクティブが開きます。別のパースペクティブに切り替えるには、メイン・メニュー・バーの直下にある、そのパースペクティブのタブをクリックします。

表 1 : [Studio] パースペクティブでのユーザ・アクティビティ

パースペクティブ	アクティビティ
Authoring	• プロジェクトの作成と編集 • グラフィカル編集環境のビジュアル・エディタでのプロジェクトと図の開発 • CCL コードを編集するテキスト指向の編集環境である CCL エディタでのプロジェクトの開発 • プロジェクトのコンパイル • Aleri モデルのインポート
Learning	• 用例プロジェクトのロード • 用例プロジェクトを手順に従って実行することによって、ストリームへのサブスクライブ、デモ・データの発行、結果の表示を行ったときの動作の理解 **注意：** [Learning] パースペクティブで開始したアクティビティは、[Authoring] パースペクティブと [Run-Test] パースペクティブで開きます。これによって、これらのパースペクティブの機能が利用でき、用例プロジェクトの学習が促進されます。
Run-Test	• 起動とサーバへの接続 • プロジェクトの実行 • サーバへのデータ・ファイルのアップロードまたはストリームへのデータの手動入力でのテスト・データの入力 • データの発行 • 実行中のプロジェクトに対するクエリの実行 • Event Tracer とデバッガを使用してのブレークポイントとウォッチポイントの設定、プロジェクトを通してのデータ・フローのトレース • 受信イベント・データのプレイバック・ファイルへの記録と、キャプチャされたデータの実行中のプロジェクトへのプレイバック • パフォーマンスのモニタ

例

Event Stream Processor には、複数の用例プロジェクトが同梱されています。

用例を ESP スタジオで表示し、製品と共にインストールされているサンプル・データに対して実行できます。 [Studio Learning] パースペクティブで用例を手順に従って実行して、単純化されたイベント・データのセットのシステム内での流れ方を的確に理解できます。

また、CCL と SPLASH のサンプル・コードについては、『用例ガイド』、『CCL プログラマーズ・ガイド』、『SPLASH プログラマーズ・ガイド』を参照してください。

参照：
- *サンプル・プロジェクト図* (22 ページ)
- *単純なプロジェクトの作成* (22 ページ)

サンプル・プロジェクトのロード

製品と共にインストールされている用例プロジェクトの 1 つをロードして、ワークスペースで表示できるようにします。

1. [Learning] パースペクティブに移動します。
2. [Examples] ビューで、[LOAD] ボタンの 1 つをクリックして、用例をワークスペースにロードします。
3. [Start Example Project] ダイアログ・ボックスで、[Cancel] をクリックします。

   ここでは、プロジェクトをワークスペースにロードすることが目的なので、実行は行いません。これで、プロジェクトをビジュアル・エディタで表示できます。 以降で、プロジェクトを実行します。

   用例プロジェクトが [Authoring] パースペクティブで開き、図 (`.cclnotation` ファイル) がビジュアル・エディタで開きます。
4. 図形をクリックして図を探索し、プラス記号をクリックしてコンパートメントを開き、カラムまたはプロパティの詳細を表示します。

   注意： 用例を編集しないでください。特に、用例に対して行った変更は保存しないでください。
5. (オプション) 対応する CCL コードを表示するには、右クリックして [Switch to Text] を選択します (または [F4] キーを押します)。
   図が閉じて、プロジェクトが CCL エディタで開きます。
6. 変更を保存せずにプロジェクトを閉じます。

第 3 章 ビジュアル・エディタのオーサリング

ビジュアル・エディタでは、CCL 構文の知識がなくてもプロジェクトの作成と編集を実行できます。

ビジュアル・エディタは、特に複雑なプロジェクトの作業を行う場合に、データ・フローを容易に可視化しプロジェクト内を簡単に移動する手段として、熟練した CCL プログラマにとっても有用なツールです。ビジュアル・エディタでは、プロジェクトは、ストリーム、ウィンドウ、アダプタと、それらの間のデータ・フローを表示する 1 つまたは複数の図によって表されます。

まず、簡単なプロジェクトを開発してみましょう。グラフィカル・ツールを使用して、ストリームとウィンドウを追加し、それらを接続して、アダプタと関連付けます。ビジュアル編集ツールを使用して、単純なクエリを図に直接追加します。

基本的な図を完成させたら、プロジェクトをコンパイルして実行します。

作成した簡単なプロジェクトが機能していると確信できたら、複雑なクエリ、カスタム演算のための Flex 演算子、モジュール性、およびカスタム・アダプタなどの高度な機能に進みます。ビジュアル・オーサリング環境では、これらの機能の多くにアクセスできます。

複雑なクエリやその他の高度な機能では、CCL エディタに切り替えることができます。1 つの CCL ファイルは、一度に 1 つのエディタでしか開くことができません。ビジュアル・エディタと CCL エディタは完全に統合されています。ファイルに保存して、もう一方のエディタに切り替えた場合、新たな作業内容もそのファイルに保存されます。

図

ビジュアル・オーサリングでは、図を使用してプロジェクトのストリーム、ウィンドウ、接続、他のコンポーネントの作成と操作が行えます。また、単純なクエリも作成できます。

プロジェクトをビジュアル・エディタで開くと、ストリームとウィンドウを示す図形のコレクションが表示されます。また、図形は、データの流れを示す矢印を使用して接続されています。プロジェクトを開発するには、パレットから新しい入力と出力のストリーム、ウィンドウ、他の要素を選択して図中にドロップし、それらを接続し、動作を設定します。

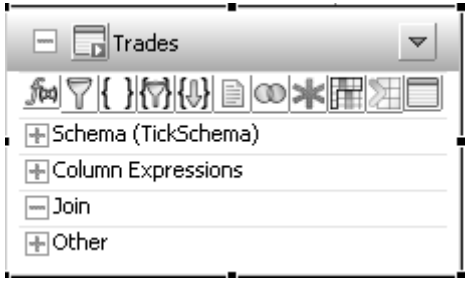
各プロジェクトには、少なくとも 1 つの図があります。1 つのプロジェクトに複数の図を設定できます。プロジェクト間で図を共有することはできません。

図形または他の要素を図に追加して保存すると、それは自動的にプロジェクトに追加されます。要素は、図からのみ、またはプロジェクトから削除できます。

図は、冗長モードまたはアイコン・モードで表示できます。

- **アイコン・モード** – コンパートメントは、表示領域を節約するために折りたたまれます。

- **冗長モード** – 要素内のすべてのコンパートメントが表示されます。



冗長モードとアイコン・モードは、図内のすべての要素に対して一括して適用でき、選択した図形にのみ適用することもできます。

ビジュアル・オーサリング環境

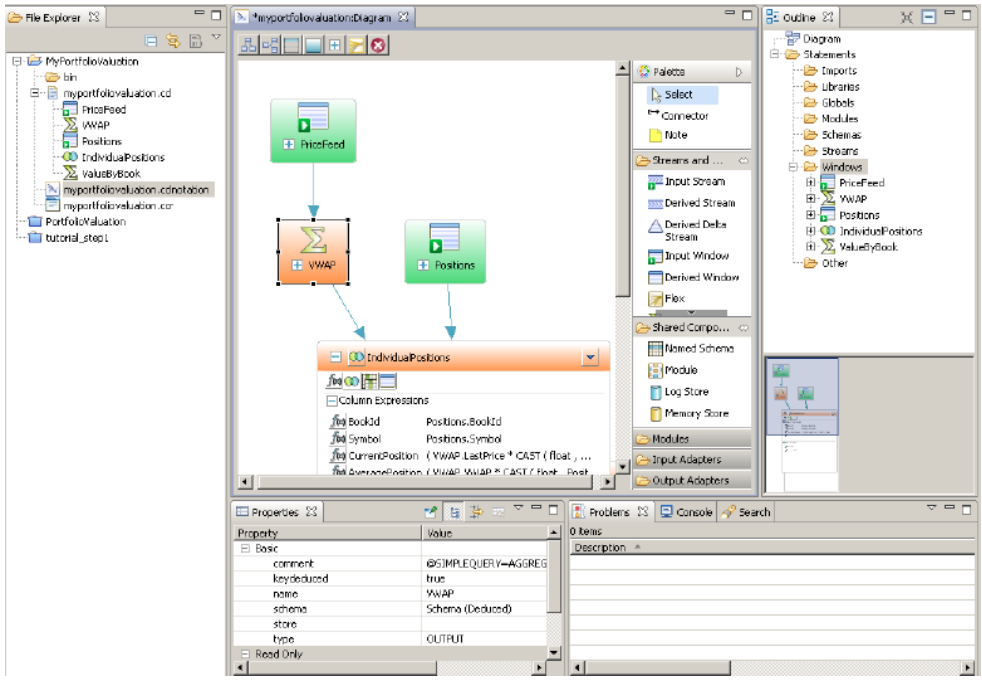
ビジュアル・エディタと、[Authoring] パースペクティブ内の他のツールやビューを使用することによって、図の作成、表示、編集ができます。

- **[Editor] –** [Authoring] パースペクティブの中央にある、図を編集するキャンバスです。
- **[Palette] –** 図中に新しい CCL 要素を作成するために使用されるツールのグループを含みます。パレット上のほとんどの図形は、CCL 文に対応しています。
- **[File Explorer] –** フォルダとファイルの階層的な構造を提供します。
- **[Properties] ビュー –** 図で選択したオブジェクトのプロパティが表示されます。このビューでプロパティを設定したり、式を編集したりできます。
- **[Outline] ビュー –** 図内のすべての要素に対するインデックスを、階層ツリー構造として提供します。アダプタが起動される順序も表示されます。このビュー

の要素を右クリックすると、その要素の図中での表示、削除、変更、子要素の追加が行えます。

- **[Overview]** – 全体像を理解するのに役立ちます。また、巨大で複雑な図で他の領域に簡単に移動するのに役立ちます。
- **[Search]** – ワークスペースでテキスト文字列を見つけるための全文検索機能を提供します。File Explorer での移動、CCL エディタでのプロジェクト・コンテンツの検索に役立ちます。検索結果を絞り込めます。また、見つかった結果のコピー、削除、置換を行えます。
- **[Problems]** – プロジェクトの検証時またはファイルのアップロード時に発見されたエラーが表示されます。
- **[Console]** – スタジオ・スクリプトによって生成されたメッセージが表示されます。

図 3 : [Authoring] パースペクティブのビュー



ビジュアル・エディタでのプロジェクトの編集

グラフィカル・ユーザ・インタフェースで図を編集します。

1. [Authoring] パースペクティブで、[File Explorer] に移動します。

2. 保存したプロジェクトをビジュアル・エディタで開くには、`.cclnotation` ファイル名をダブルクリックします。
3. 図内をクリックし、パレットを使用しながら、編集を開始します。

ヒント： ビジュアル・エディタ・ウィンドウを全画面表示にするには、最上部の [*name*:Diagram] タブをダブルクリックします。元に戻すには、もう一度ダブルクリックします。

4. 編集した後、保存します ([Ctrl] キーを押しながら [S] キーを押します)。
この操作によって、`.cclnotation` ファイル (図) と `.ccl` ファイル (CCL) の両方に変更が保存されます。
5. ビジュアル・エディタと CCL エディタを切り替えるには、[Switch to Text] または [Switch to Visual] を選択するか、[F4] キーを押します。
6. 図を閉じるには、[Ctrl] キーを押しながら [W] キーまたは [Ctrl] キーを押しながら [F4] キーを押すか、エディタの最上部にあるタブの [X] をクリックします。

注意： このマニュアルではビジュアル・エディタで実行できるすべてのタスクについて説明していませんが、その他のグラフィカル・ユーザ・インタフェースと同様、ほとんどのタスクを実行するために、いくつかの方法が利用できます。たとえば、多くのコンテキストで、以下のような操作によってアクションを実行できます。

- 図形やメイン・ツールバーにあるボタンまたはその他のアイコンをクリックする。
- ショートカット・キーを使用する。
- 要素をダブルクリックしてその要素を開く。
- 右クリックしてコンテキスト・メニューから選択する。
- メイン・メニュー・バーから選択する。
- [Properties] ビューで要素値を編集する。

ESP スタジオには、Eclipse ベースのアプリケーションに共通する機能も含まれます。

参照：

- *サンプル・プロジェクトの作成* (23 ページ)

図形の図への追加

ストリーム、ウィンドウ、共有コンポーネントを作成し、継続クエリを使用して関連付け、それらをアダプタにアタッチします。

1. 図をビジュアル・エディタで開きます。

2. パレットの図形ツール ([Input Window]、[Flex] など) をクリックし、図の空白領域をクリックします。
   この操作によって、新しい図形が図に作成されます。赤色の枠線は、図形定義が不完全または正しくないことを示しています。図形定義が完了すると、枠線は灰色に変わります。

   注意： パレットから図へのドラッグ・アンド・ドロップは実行しないでください。

3. 図形定義を完了するために必要なアクションを表示するには、図の図形の上にカーソルを置きます。

次のステップ
実行する必要のあるステップについて、特定の図形のタスクを参照してください。

要素の接続

図内の 2 つの図形を接続し、それらの間のデータ・フローを作成します。

コネクタ・ツールは、ストリームやウィンドウの間のフローの作成、ストリームや共有コンポーネントの間の集約の有効化、または図形間のノートのアタッチを行います。

1. [Palette] ビューで[Connector] ツールを選択します。
2. 出力を生成する図形をクリックします。
   この操作によって、コネクタ・ラインが最初の図形にアタッチされます。
3. データを受信する図形をクリックして、データ・フローの方向を示します。
   図形間の接続が許可されている場合、カーソルの横に接続アイコンが表示されます。接続が許可されていない場合、「接続不可」アイコン  が表示されます。

   ヒント： 複数の接続を追加するには、[Shift] キーを押しながら、[Connector] ツールをクリックして選択を保持し、各接続を追加します。通常の選択状態に戻るには、[Esc] キーを押すか、またはパレットの [Select] ツールをクリックして、[Connector] ツールの選択保持を解除します。

図の表示変更

図を冗長モードまたはアイコン・モードで表示します。図では、要素を左から右へ、または上から下へ配列します。

前提条件
図をビジュアル・エディタで開きます。

- アイコン・モードと冗長モードの間で図形を切り替えるには、以下を実行します。
  - 冗長モードで、左上隅にある「マイナス」記号をクリックし、折りたたみます。
  - アイコン・モードでは、「プラス」記号をクリックして、展開します。
- すべての図形をアイコン・モードまたは冗長モードで表示するには、ビジュアル・エディタのツールバーで、[All Verbose] または [All Iconic] をクリックします。
- 方向を切り替えるには、ビジュアル・エディタのツールバーで、[Layout left to right] または [Layout top down] をクリックします。

注意： その他の表示オプションについては、オブジェクトまたは図の表面を右クリックし、コンテキスト・メニューから選択します。

単純なプロジェクトの作成

単純なプロジェクトをビジュアル・エディタで作成します。

ここで説明されているポートフォリオ評価の例は、ポジション (株式の持ち高) のセットと市場価格フィードを入力とし、ポジションを評価します。

サンプル・プロジェクト図

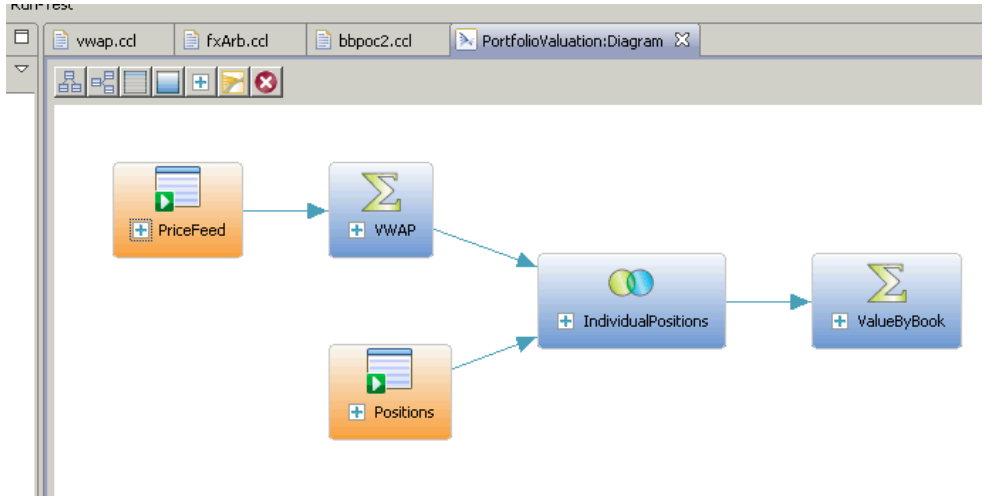
PortfolioValuation 図は、単純なクエリを使用して、2 つの入力ウィンドウからのデータを集約し、ジョインします。

例：

1. 5 つのカラムを持つ入力ウィンドウ PriceFeed からデータを読み込みます。カラムは、Id、Symbol (証券コード)、Price (株価)、Shares (株数)、TradeTime (取引時刻) です。

2. 単純な集約クエリを適用して、移動平均の出来高加重平均価格 (VWAP) を作成します。VWAP を使用して、すべての少量価格変動のポジション変化の価値ではなく、平均価格に基づくポジションの価値を確認できます。
3. 3 つのカラムを持つ別の入力ウィンドウ Positions からデータを読み込みます。カラムは、BookId (帳簿 Id)、Symbol (証券コード)、SharesHeld (持ち高) です。
4. 単純なジョイン・クエリを適用し、VWAP 集約によって得られる市場価格を持ち高 (Positions) にジョインして、各株式のポジションの価値を確認します。
5. さらに 1 つの集約を適用して、各 "book (帳簿)" の総価値を表示します。この集約 ValueByBook は、ポジションをポートフォリオまたはファンドの異なる "book (帳簿)" に編成する戦略を具現化します。

図 4 : ポートフォリオ評価サンプルの図 (アイコン・モード)



サンプル・プロジェクトの作成

スタジオを使用して、イベント・データの処理手順の新しいセットを定義します。

1. [File] > [New] > [Project...] を選択します。
2. [Name] フィールドに、`MyPortfolioValuation` と入力します。

   プロジェクト名は、以下の要件を満たす必要があります。
   - 文字、アンダースコア、またはドル記号で始まる。
   - その他のすべての文字は、英数字、アンダースコア、またはドル記号である。
   - スペースを含まない。

独自のプロジェクトでは、任意の名前を使用できます。 作成するサンプル・プロジェクトを実行できるようにするために、ここにリストされている値を使用してください。

3. [Directory] フィールドで、デフォルト・ロケーションをそのまま使用するか、または新規プロジェクト・フォルダを保存するディレクトリを選択します。
   指定したディレクトリに、次の 3 つのファイルが作成されます。
   - ***project_name*.ccl** – CCL コードを含みます。
   - ***project_name*.cclnotation** – `.ccl` ファイルに対応する図を含みます。
   - ***project_name*.ccr** – プロジェクト構成を含みます。

   たとえば、trades というプロジェクトでは、`trades.ccl`、`trades.cclnotation`、`trades.ccr` というファイルが `trades` ディレクトリに作成されます。
4. [Finish] をクリックすると、プロジェクトのファイルが作成されます。
   1 つの入力ストリーム NEWSTREAM を持つ新規プロジェクトがビジュアル・エディタで開き、インライン・スキーマが編集可能な状態になります。

入力アダプタの追加

アダプタをアタッチします。これを行うには、アダプタを図に挿入し、ストリームまたはウィンドウに接続し、プロパティを設定します。

入力アダプタは入力ストリームまたは入力ウィンドウの外部ソースを識別し、Event Stream Processor サーバが受け付けるフォーマットに変換します。 アダプタの図への追加は、入力または出力のストリームまたはウィンドウの追加の前または後のいずれでも行えます。

この例は、アダプタを挿入し、スキーマ検出用に有効にし、生成して、入力ウィンドウとそのスキーマに自動的にアタッチする方法を示します。 これは、スキーマ検出をサポートするアダプタに対するベスト・プラクティスです。

代替手法として、ESP スタジオを使用してストリームまたはウィンドウを作成し、その後にアダプタをアタッチすることもできます。 この手法は、アダプタがスキーマ検出をサポートしていない場合か、入力ストリームまたは入力ウィンドウに対してインライン・スキーマを明示的に作成する必要がある場合に使用してください。

スキーマ検出をサポートするアダプタのリストと設定するプロパティの説明については、『スタジオ・ユーザーズ・ガイド』を参照してください。

1. パレットで [Input Adapters] コンパートメントを開き、該当するアダプタを見つけます。
   この例では、[File XML Input] アダプタを選択します。このアダプタは、XML ファイルからデータを読み込みます。

2. パレット内でアダプタをクリックし、次に図中をクリックします。
   アダプタの図形が挿入されます。枠線は、プロパティを定義してアダプタをストリームまたはウィンドウにアタッチするまで、完了していないことを示す赤色で表示されます。
3. アダプタの図形のツールバーで、[Edit Properties] () をクリックします。
4. (オプション) [Adapter Properties] ダイアログ・ボックスで、アダプタを識別するように [Name] を変更します。
5. [Adapter Properties] ダイアログ・ボックスで、スキーマ検出用にアダプタを設定します。
   a) [Set properties locally] を選択します。

      これによって、ダイアログ・ボックスの右側の表で、プロパティを設定できます。 必須プロパティは赤色で表示されます。

      注意： [Use named property set] は選択しないでください。これを選択すると、このアダプタのスキーマを検出できなくなります。

      XML インプット・アダプタ用にスキーマ検出を使用するには、アダプタが読み込むデータ・ファイルの絶対パスを [Directory] プロパティと [File] プロパティに設定します。
   b) [Directory] プロパティの [Value] カラムをクリックし、データ・ファイルを含むディレクトリを参照するためのボタンをクリックします。 [File] プロパティに値を入力します。
      この例では、製品と共にインストールされたサンプル・データへのパスを指定します。

プロパティ	値
Directory	*workspace_install_path*¥exampledata Windows の場合のデフォルト：C:¥Documents and Settings¥*username*¥My Documents¥SybaseESP¥5.0¥workspace¥exampledata Linux と Solaris の場合のデフォルト： *your_home_directory*/SybaseESP/5.0/workspace/exampledata
ファイル	positions.xml

6. [OK] をクリックして保存します。

次のステップ

スキーマをインポートし、データ・ファイルと同じスキーマを持つ、接続された入力ストリームまたは入力ウィンドウを作成します。

参照：
- *入力アダプタと出力アダプタ* (8 ページ)

スキーマ検出
スキーマ検出機能を使用すると、外部スキーマを検出し、アダプタに接続されているデータソースからのデータのフォーマットに基づいて CCL スキーマを作成できます。

ストリームまたはウィンドウの各ローは同じ構造またはスキーマを持つ必要があります。これには、カラムの名前、カラムのデータ型、カラムの配列順序が含まれます。複数のストリームまたはウィンドウが同じスキーマを使用できますが、1 つのストリームまたはウィンドウは 1 つのスキーマしか持つことができません。

スキーマ検出を使用して、スキーマを検出し、アダプタに接続されているデータソースのデータのフォーマットに基づいてスキーマを自動的に作成できます。新しいスキーマを手動で作成する必要はありません。たとえば、データベース・インプット・アダプタの場合、アダプタの接続先のデータベースから特定のテーブルに対応するスキーマを検出できます。

スキーマを検出するには、最初にアダプタ・プロパティを設定する必要があります。スキーマ検出をサポートする各アダプタには、スキーマ検出を有効にするために設定する必要のあるユニークなプロパティがあります。

参照：
- *スキーマの検出* (26 ページ)

スキーマの検出
アダプタ図形の [Schema Discovery] ボタンを使用して、アダプタからのデータ・フォーマットに基づいてスキーマの検出、自動作成を行います。

前提条件
アダプタを図に追加して、そのプロパティを設定します。

手順

1. アダプタ・ツールバーの [Schema Discovery] をクリックします。
   - スキーマが適切に構成されていて、1 つ以上のデータ・セットが検出された場合、スキーマの表示と選択を可能にするダイアログ・ボックスが表示されます。
   - スキーマを正常に検出できなかった場合、そのアダプタのスキーマが検出されなかったことを示すエラー・メッセージが表示されます。この場合、以下を実行できます。

- アダプタ・プロパティがスキーマ検出に対して構成されていることを確認します。
- アダプタがスキーマ検出をサポートしているかどうかを確認します。

2. 必要なスキーマを選択します。
   データ・セットを展開し、スキーマを表示します。
   この例では、[positions.xml] を選択し、[Next] をクリックします。
3. [Create Element] ダイアログで、[Create new input window] を選択します。
   このオプションは、新規ウィンドウの作成とアダプタへのアタッチ、ウィンドウのインライン・スキーマの作成、アダプタから検出されたスキーマを使用したウィンドウの設定を行います。
   アダプタがストリームまたはウィンドウにアタッチされていない場合、その他に、以下のオプションがあります。
   - **Create a new input stream –** 新規ストリームの作成とアダプタへのアタッチ、ストリームのインライン・スキーマの作成、アダプタから検出されたスキーマを使用してストリームの設定を行います。
   - **Create new named schema –** 新しい名前付きスキーマの作成、アダプタから検出されたスキーマを使用してそのスキーマの設定を行います。
4. [Finish] をクリックします。
   新規入力ウィンドウが表示され、自動的に XML インプット・アダプタに接続されます。
5. 入力ウィンドウの [Schema] コンパートメントで、[BookId] カラムと [Symbol] カラムの [Toggle Key]  ボタンをクリックして、プライマリ・キーを指定します。
   プライマリ・キーを指定すると、図形が有効になります。
6. (オプション) 入力ウィンドウの [Edit] ボタンをクリックし、`Positions` という名前を付けます。

次のステップ

別の入力ウィンドウ (PriceFeed) を作成します。次のいずれかを実行します。

- 同じ `exampledata` ディレクトリにある `pricefeed.xml` ファイルから入力を取得する、別の XML インプット ・アダプタを挿入します。スキーマ検出を使用して、自動的にウィンドウを生成、接続し、[Id] カラムをプライマリ・キーとして設定します。
- PriceFeed 入力ウィンドウを手動で作成します。

参照：

- *スキーマ検出* (26 ページ)

手動による入力ウィンドウの追加

サンプルの PortfolioValuation プロジェクトの図に入力ウィンドウを追加します。

これらの手順では、スキーマをインポートせずに、入力ウィンドウを直接作成し、スキーマを定義します。

入力アダプタを使用してスキーマを検出し、両方の入力ウィンドウを自動的に生成している場合は、これらの手順を省略して、次のタスクに直接進みます。

1. ビジュアル・エディタの図の右側のパレットで、[Streams and Windows] コンパートメントを開きます。
2. [Input Window] をクリックします。
3. 入力ウィンドウを挿入する、図中の空白領域をクリックします。
   入力ウィンドウ・オブジェクトがプロジェクトに追加されます。枠線が赤色で表示され、有効にするには、さらに定義を行う必要があることを示します。
4. 次のいずれかの方法を使用して、入力ウィンドウの名前を設定します。
   - アイコン・モードの場合、図形を 1 回クリックして選択し、さらにクリックして名前を編集します。
   - 冗長モードの場合、名前の隣の編集アイコンをクリックします。

   この例では、名前 `PriceFeed` を入力します。
5. 図形を冗長モードに展開し、入力ウィンドウの図形のツールバーの [Add Column] ( ) をクリックして、新しいカラムのそれぞれを追加します。

   ヒント： 任意のアイコンをカーソルでポイントすると、その名前が表示されます。

   新しいカラムが、デフォルトの名前と、デフォルトの整数のデータ型を使用して作成されます。
6. カラム名を入力し、次にデータ型をクリックします。再度クリックして、ドロップダウン・リストからデータ型を選択します。
   この例では、以下のカラム名とデータ型を入力します。
   - **Id –** 整数
   - **Symbol (証券コード) –** 文字列
   - **Price (株価) –** 浮動小数点値
   - **Shares (株数) –** 整数
   - **TradeTime (取引時刻) –** 日
7. Id カラムの ボタンをクリックして、[Key] 記号に切り替えます。
   入力ウィンドウ用にプライマリ・キーが必要です。

これで、Id カラムが PriceFeed 入力ウィンドウのプライマリ・キーとなりました。枠線が、赤色から灰色に変化します。

8. 保存します ([Ctrl] キーを押しながら、[S] キーを押します)。

入力ウィンドウとそのスキーマ (または抽出されたスキーマ) が図に表示されます。保存ポリシーを設定していないので、デフォルト・ポリシーの「すべてのレコードを保存」が適用されます。

単純な継続クエリの追加

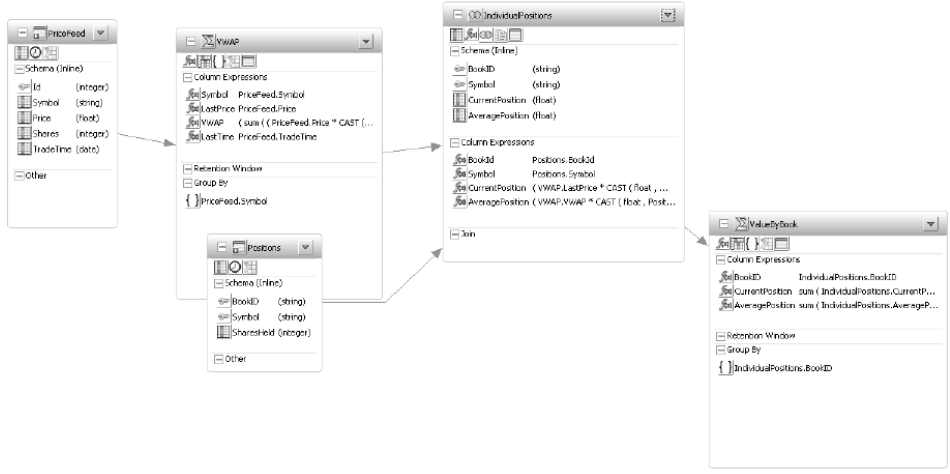
ビジュアル・エディタで利用可能なクエリのタイプを確認します。まず、1 つの単純なクエリをプロジェクトに追加することから始めます。

PortfolioValuation 図を作成するために、次の 3 つの単純なクエリを追加します。

1. VWAP と呼ばれる単純な集約クエリ。出来高加重平均価格 (VWAP) を作成します。この集約クエリは、PriceFeed 入力ウィンドウから入力を取得します。
2. VWAP 出力を Positions 入力ウィンドウにジョインする単純なジョイン・クエリ。ジョインの結果に IndividualPositions という名前を付けます。
3. ValueByBook という名前の 2 つ目の単純な集約クエリ。各 "book (帳簿)" の総価値を表示します。

例の詳細については、図と、単純なクエリの作成と変更の手順を参照してください。

図 5 : ポートフォリオ評価サンプルの図 (冗長モード)



単純なクエリ

ビジュアル・エディタで利用可能なクエリのセットを使用して、最も一般的なクエリ作成タスクを完了します。クエリのセットには、フィルタ処理、集約、ジョイン、計算、ユニオン、パターン一致が用意されています。

パレットの [Streams and Windows] では、次の 6 つのクエリ・ツールが利用できます。

- **[Filter] –** フィルタ式に基づいてストリームをフィルタリングし、対象となるイベントのみを抽出できます。
- **[Aggregate] –** 共通の値を持つイベントをグループ化し、そのグループの要約統計量 (たとえば、平均値) を計算できます。また、イベントの発生時刻または数に基づいて、ウィンドウ・サイズを定義できます。
- **[Join] –** 複数のストリームまたはウィンドウのレコードを結合し、各ソースの情報を持つ新しいレコードを形成できます。
- **[Compute] –** 入力内容と異なるスキーマを持つ新しいイベントを作成し、イベントの各カラム (フィールド) に含める値を計算できます。
- **[Union] –** 共通スキーマを共有する複数のストリームまたはウィンドウすべてを、1 つのストリームまたはウィンドウに結合できます。
- **[Pattern] –** イベントのパターンを確認できます。確認は、1 つのストリームまたはウィンドウでも、複数のストリームまたはウィンドウにまたがっても可能です。ESP サーバは、実行中のプロジェクトのイベント・パターンを検出すると、出力イベントを生成します。

単純なクエリとしての集約の作成

単純な集約クエリをサンプル図に追加して、出来高加重平均価格 (VWAP) を作成します。

集約クエリは、共通の値を持つイベントをグループ化し、そのグループの要約統計量を計算します。

1. ビジュアル・エディタのパレットの [Streams and Windows] で、[Aggregate] (Σ) をクリックします。
2. 図をクリックして、オブジェクトを作成します。
3. デフォルト名 Aggregate1 を VWAP に変更します。
4. PriceFeed を VWAP 集約に接続します。
   a) パレットの [Connector] 図形をクリックします。
   b) [PriceFeed] 入力ウィンドウをクリックし、次に [VWAP] 集約をクリックします。
      集約の枠線が、赤色から黒色に変化し、このオブジェクトが有効になって入力を取得していることを示します。

5. カラム式を入力します。
   a) 図形ツールバーで [Copy Columns from Input] () をクリックして、集約ウィンドウのスキーマにコピーするカラムを選択します。
   b) カラムを追加するには、図形ツールバーの [Add Column Expressions]  をクリックします。

   この例では、次のカラムをコピーします。
   - PriceFeed.Symbol
   - PriceFeed.Price
   - PriceFeed.TradeTime

   次に別のカラムを追加し、名前を VWAP に変更します。
6. カラム式を編集します。これを行うには、ダブルクリックしてインライン・エディタを開くか、または式を選択し、[Ctrl] キーを押しながら、[F2] キーを押してポップアップ・エディタで式を開きます。
   この例では、次の変更を行います。
   - Price を LastPrice に編集
   - TradeTime を LastTime に編集
   - VWAP カラムを次に編集：

```
( sum ( ( PriceFeed.Price *
  CAST ( FLOAT , PriceFeed.Shares ) ) ) /
  CAST (FLOAT , sum ( PriceFeed.Shares ) ) )
```

7. [Add GroupBy Clause] ([{ }]) をクリックして、集約オブジェクト内のカラムのグループ化を編集します。

   注意： [Aggregate] の図形は、GROUP BY 式を 1 つだけ持つ必要があります。

   この例では、グループ化カラムとして、[PriceFeed.Symbol] を選択します。
8. (オプション) [Set Keep Policy]  をクリックして、保存ウィンドウを作成します。
   デフォルトのポリシーでは、受信データのすべてのローが維持されます。最後のローだけを維持、特定の数のローを維持、または特定の時間のローを維持、のいずれも選択できます。これは、CCL **KEEP** 句を定義します。
9. (オプション) [Toggle]  オプションを使用して、集約オブジェクトを [LOCAL] から [OUTPUT] に変更します。

単純なクエリとしてのジョインの作成

CCL の **FROM** 句のジョイン式オプションに類似した、複数のデータソースを結合する単純なジョイン・クエリをサンプル図に追加します。

1. ビジュアル・エディタ・パレットの [Streams and Windows] で、[Join] を選択します。

必要に応じて、[Streams and Windows] の下のコンパートメントを閉じるか、またはコンパートメントの下の矢印を使用して、[Join] が見えるようにします。

2. 図をクリックして、オブジェクトを作成します。
   この例では、ジョイン・オブジェクト名を編集して `IndividualPositions` にします。
3. ジョイン・オブジェクトを適切なストリームまたはウィンドウに接続します。
   ジョイン・オブジェクトを任意のストリーム、ウィンドウ、または Flex 演算子にアタッチします。ジョイン・オブジェクトには複数の入力がありますが、出力は 1 つだけです。

   注意： ストリーム、ウィンドウ、デルタ・ストリームは、ジョインに含めることができます。ただし、デルタ・ストリームをジョインに含めることができるのは、KEEP 句を指定している場合のみです。1 つのジョインに含めることができるストリームは 1 つだけです。サポートされるジョインについては、『スタジオ・ユーザーズ・ガイド』または『CCL プログラマーズ・ガイド』を参照してください。

   この例では、VWAP 集約オブジェクトと Positions 入力ウィンドウを IndividualPositions ジョイン・オブジェクトに接続します。
4. ジョイン図形ツールバーの [Copy Columns] () をクリックして、コピーするカラムを選択します。
   この例では、[Select All] を選択し、2 つ目の [Symbol] フィールドのチェック・ボックスをオフにします。
5. [Add Column Expressions] () をクリックします。
   この例では、Current Position と Average Position という 2 つのカラムを追加します。
6. カラム式を変更するには、次の手順のいずれかに従います。
   - [F2] キーを押すと、インライン・エディタが開きます。入力カラムと組み込み関数を示すドロップダウン・メニューが表示されます。使用する入力カラムまたは組み込み関数を選択して、式を定義します。
   - [Ctrl] キーを押しながら [F2] キーを押すと、式エディタが開きます。[Ctrl] キーを押しながら [Space] キーを押すと、利用可能な入力カラムと組み込み関数が表示されます。または、希望する式を手動で入力します。
   - [Properties] ビューで式を修正します。

   この例では、以下のカラム式を作成します。
   - `BookId Positions.BookId`
   - `Symbol Positions.Symbol`
   - `CurrentPosition ( VWAP.LastPrice * CAST ( float , Positions.SharesHeld ) )`

- `AveragePosition ( VWAP.VWAP * CAST ( float , Positions.SharesHeld ) )`

7. ジョイン図形の [Join Conditions] コンパートメントで、ジョイン条件を設定します。
   この例では、ジョインを VWAP 入力と Positions 入力にこの順番で接続した場合、[Join Conditions] コンパートメントには 2 つの要素があります。最初の要素は、ジョインの一番左の要素を定義します。最初に VWAP に接続した場合、最初の要素 (ジョインの左側) は VWAP です。これは、ダブルクリックして変更できます。この例では、2 番目のジョイン要素を構成する必要があります。
   a) 2 番目のジョイン要素をダブルクリックすると、[Edit Join Expression] ダイアログ・ボックスが表示されます。
   b) ジョインするカラムを [Source 1] と [Source 2] から選択します。
      この例では、Positions と VWAP の両方に対して [Symbol] を選択します。
   c) [Add] をクリックします。
      選択したカラムが [Join Constraints] に表示され、そこに、`Positions.Symbol=VWAP.Symbol` が表示されます。ビジュアル・エディタでは、ジョイン制約を手動で編集できません。
   d) (オプション) ジョイン・タイプを選択します。
      この例では、デフォルトの [LEFT] を使用します。
8. [OK] をクリックします。

サンプル・プロジェクトの完了

未使用の要素を削除して図をクリーンアップします。

1. ValueByBook と呼ばれる、さらに別の単純な集約クエリを作成します。
   a) カラム式を設定します。
      - `BookID IndividualPositions.BookID`
      - `CurrentPosition sum ( IndividualPositions.CurrentPosition )`
      - `AveragePosition sum ( IndividualPositions.AveragePosition )`
   b) `IndividualPositions.BookID` に Group By を設定します。
   c) [OUTPUT] に切り替えます。
   d) それを IndividualPositions ジョイン・オブジェクトに接続します。
2. プロジェクトの作成時に自動的に追加された、未使用の入力ストリーム要素 NEWSTREAM を削除します。
   その他の未使用の要素をすべて削除します。これで、プロジェクトを実行できます。

3. (オプション) アイコン・モードに切り替えて、作成した図と「ポートフォリオ評価サンプルの図」の図を比較します。
   - 図形の左上隅にある [Toggle Image] ボタンをクリックします。または、
   - ツールバーの [All Iconic] ボタンまたは [All Verbose] ボタンをクリックします。

要素の削除

要素をプロジェクトから削除すると、完全に削除されます。図からのみ削除することもできます。

1. 図中で 1 つ以上の要素を選択します。
2. 右クリックして次のいずれかを選択します。
   - [Delete Element] — 要素をプロジェクトから削除します。
   - [Delete from Diagram] — 要素を図からのみ削除します。プロジェクト内には保持されます。プロジェクトを実行すると、図中にないものも含めて、プロジェクト内のすべてを実行します。
3. [Delete Element] を選択した場合には、表示される確認メッセージで削除を承認します。

第 4 章 CCL オーサリング

CCL エディタは、ESP スタジオ内のテキスト・オーサリング環境で、CCL コードを編集するために用意されています。

CCL エディタのみで作業したり、CCL エディタをビジュアル・エディタの補助ツールとして使用したりできます。CCL エディタには、構文完了オプション、構文チェック、エラー検証が用意されています。

1 つの CCL ファイルは、一度に 1 つのエディタでしか開くことができません。ビジュアル・エディタと CCL エディタは完全に統合されています。ファイルに保存して、もう一方のエディタに切り替えた場合、新たな作業内容もそのファイルに保存されます。

Event Stream Processor を初めて使用する場合は、ビジュアル・エディタから始めた方が簡単です。製品に習熟し、簡単なプロジェクトのコンパイルと実行ができるようになれば、CCL エディタを使用して、プロジェクトに高度な機能を追加できます。

ESP スタジオ内で CCL エディタを使用する方法については、『スタジオ・ユーザーズ・ガイド』を参照してください。

CCL 言語の使用方法と参照情報の詳細については、『CCL プログラマーズ・ガイド』を参照してください。

CCL エディタでの編集

スタジオの CCL エディタで、CCL コードをテキストとして更新、編集します。

1. [Authoring] タブをクリックします。
2. File Explorer で、プロジェクト・コンテナを展開し、`.ccl` ファイル名をダブルクリックして、CCL エディタでそのファイルを開きます。

   デフォルトで、`.ccl` ファイルは CCL エディタで開きます。また、`.cclnotation` ファイルはビジュアル・エディタで開きます。

   注意： CCL に詳しいユーザは、IMPORT 文を使用して別ファイルから共有スキーマとモジュール定義をインポートすることによって、複数の CCL ファイルを同じプロジェクトで使用できます。
3. CCL エディタ・ウィンドウでテキストの編集を開始します。

注意： 文字列リテラル内の円記号は、エスケープ文字として使用されます。そのため、すべてのウィンドウ・ディレクトリ・パスでは、円記号を 2 つ続けて指定する必要があります。

4. (オプション) [Ctrl] キーを押しながら [Space] キーを押すと、構文完了プロポーザルが表示されます。
5. (オプション) CREATE 文のテンプレート・コードを挿入するには、右クリックして、[Create] を選択し、作成する要素を選択します。
6. [File] > [Save] を選択するか、[Ctrl] キーを押しながら [S] キーを押して、`.ccl` ファイルとプロジェクトを保存します。

サンプル・プロジェクトの CCL

次に、スタジオ・ビジュアル・エディタの Portfolio Valuation サンプル・プロジェクトの CCL を示します。読みやすいように改行しています。

```
CREATE INPUT WINDOW PriceFeed
  SCHEMA ( Id integer , Symbol string , Price float , Shares integer ,
TradeTime date )
  PRIMARY KEY ( Id) ;

/**@SIMPLEQUERY=AGGREGATE*/
CREATE OUTPUT WINDOW VWAP PRIMARY KEY DEDUCED AS
  SELECT
       PriceFeed.Symbol Symbol ,
       PriceFeed.Price LastPrice ,
       ( sum ( ( PriceFeed.Price *  CAST ( FLOAT ,
PriceFeed.Shares ) ) ) /
           CAST ( float , sum ( PriceFeed.Shares ) ) )  VWAP ,
       PriceFeed.TradeTime LastTime
  FROM PriceFeed
  GROUP BY PriceFeed.Symbol ;

CREATE INPUT WINDOW Positions
  SCHEMA ( BookID string , Symbol string , SharesHeld integer )
  PRIMARY KEY ( BookID, Symbol) ;

/**@SIMPLEQUERY=JOIN*/
CREATE OUTPUT WINDOW IndividualPositions
  SCHEMA ( BookID string , Symbol string , CurrentPosition float ,
AveragePosition float )
  PRIMARY KEY ( BookID, Symbol)
  AS
  SELECT
      Positions.BookId BookId ,
      Positions.Symbol Symbol ,
      ( VWAP.LastPrice * CAST ( float , Positions.SharesHeld ) )
CurrentPosition ,
      ( VWAP.VWAP * CAST ( float , Positions.SharesHeld ) )
AveragePosition
```

```
  FROM Positions , VWAP
  WHERE Positions.Symbol = VWAP.Symbol;

/**@SIMPLEQUERY=AGGREGATE*/
CREATE OUTPUT WINDOW ValueByBook PRIMARY KEY DEDUCED AS
  SELECT
     IndividualPositions.BookID BookID ,
     sum ( IndividualPositions.CurrentPosition ) CurrentPosition ,
     sum ( IndividualPositions.AveragePosition ) AveragePosition
  FROM IndividualPositions GROUP BY IndividualPositions.BookID ;
```

第 5 章　プロジェクトの実行とテスト

ESP スタジオを使用すると、プロジェクトを実行し、全面的にテストできます。

開発時には、ESP スタジオを使用して、コンパイルされた任意のプロジェクトのローカルまたはリモートのサーバに対する実行、プロジェクト内で定義されているストリームとウィンドウを通して流れるデータの観察、クエリの実行、デバッグ・ツールの使用ができます。 プロジェクト設定とライセンスによって、プロジェクトを実行する際に使用できるサーバ接続のタイプが決定されます。 一部のアダプタでは、特別なライセンス要件を満たす必要があります。

ライセンスでサポートされている場合は、ESP スタジオ内ですぐにローカル・クラスタに接続してプロジェクトを実行できます。

運用環境では、通常、ESP スタジオではなく、『管理者ガイド』で説明されているコマンドライン・ユーティリティやプロシージャを使用して、プロジェクトをリモート・サーバ上で実行します。

[Run-Test] パースペクティブ

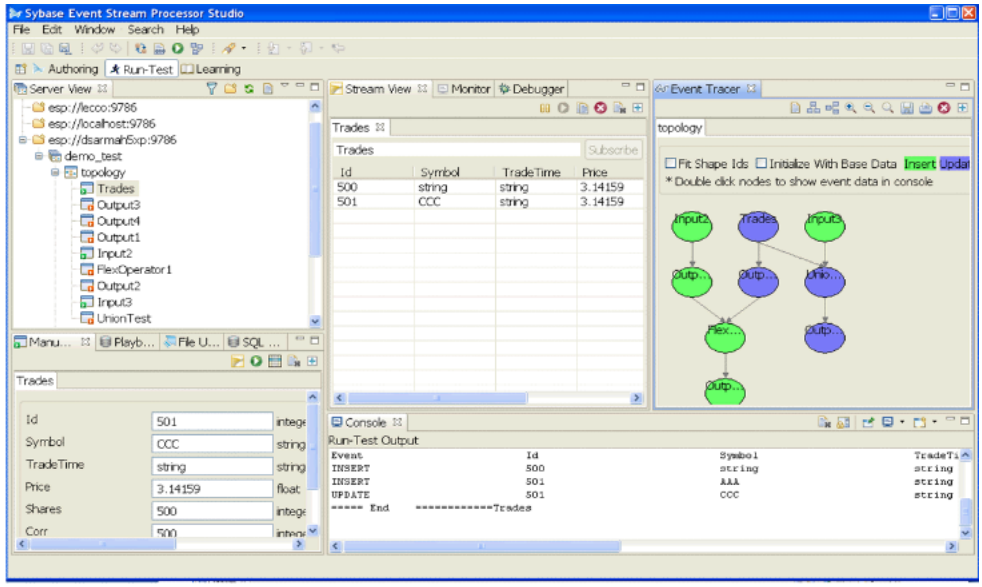
[Run-Test] パースペクティブで、プロジェクトのテスト、デバッグ、微調整が行えるツールにアクセスします。

[Run-Test] パースペクティブでは、次を実行できます。

- [Server] ビューからサーバの起動と接続
- プロジェクトの実行
- [File Upload] ビューでデータ・ファイルをサーバにアップロードするか、[Manual Input] ビューでデータをストリームに手動で入力して、テスト・データを入力
- [Server] ビューでストリームにサブスクライブ
- ストリームまたはウィンドウからデータを発行
- [SQL Query] ビューで実行中のプロジェクトに対してクエリの実行
- Event Tracer とデバッガを使用してのブレークポイントとウォッチポイントの設定、プロジェクトを通してのデータ・フローのトレース
- [Monitor] ビューでのパフォーマンスのモニタ
- [Playback] ビューで、流れているデータをプレイバック・ファイルに記録し、キャプチャされたデータを実行中の Event Stream Processor インスタンスでプレイバック

次の図は、プロジェクトのテスト時に、[Run-Test] パースペクティブに表示されるビュー・コンテンツのサンプルを示します。この図は、このマニュアルの手順で作成されるサンプル・プロジェクトとは関係ありません。

図 6 : [Run-Test] パースペクティブ



[Server] ビュー

[Server] ビューには、プロジェクトの接続と実行に利用できるサーバが表示されます。

以下を実行できます。

- プロジェクトの接続、ローカル・クラスタまたはリモート・クラスタの有効化
- 利用可能な接続のリストへの新しいサーバ URL の追加、既存サーバの削除、リストされているすべてのサーバの再接続
- [Monitor] ビューまたは [Event Tracer] ビューでのサーバの表示
- ワークスペースへのプロジェクトのロード
- メタデータ・ストリームのフィルタ処理 (デフォルト)

メタデータ・ストリームは自動的に作成され、通常、現在実行しているプロジェクトに関する正常性とパフォーマンスの情報を取得するために、運用システムの管理者によって使用されます。各ストリームに含まれる情報の詳細については、『管理者ガイド』の「メタデータ・ストリーム」を参照してください。

ローカル・クラスタへの接続

ESP スタジオをローカル・クラスタに接続します。

サーバ接続が、[Run-Test] パースペクティブに表示されます。

1. [Authoring] パースペクティブを選択します。
2. [Run Project] を選択し、プロジェクトを選択します。[OK] をクリックします。
   [Run-Test] パースペクティブに [Server] ビューが開き、プロジェクトの接続が表示されます。接続が成功すると、サーバ・フォルダの下にサーバ・ストリームが表示され、コンソールにはプロジェクトのサーバ・ログが表示されます。

   接続が失敗すると、サーバ接続エラー・ダイアログ・ボックスが表示されます。

ESP スタジオはノード (クラスタ・マネージャ) として機能し、自動的にプロジェクトをローカル・クラスタに接続します。

ストリームまたはウィンドウでのデータの表示

ストリーム・ビューでは、ストリームまたはウィンドウのデータ・フローを、情報が変化すると更新される動的ビューで確認できます。

1. [Run-Test] パースペクティブで、[Server] ビューからストリームまたはウィンドウを選択します。
2. 出力ストリームまたは出力ウィンドウを右クリックし、[Show in] > [Stream Viewer] を選びます。
   新規イベントをすべて表示するタブが [Stream] ビューに表示されます。ウィンドウを選択した場合は、そのウィンドウで現在保持されているローがすべて表示されます。
3. (オプション) ストリーム・ビュー上部のボタンを使用して、サブスクリプションを選択してから、オプションを選択します。
   - [Close Subscription URL] — 現在のストリーム・ビューを切断して閉じます。
   - [Clear contents and pause subscription].
   - [Show current subscription in new view] — 現在のテーブルが表示された新規ストリーム・ビューを開きます。

プロジェクトを実行、テストするためのその他のツール

ESP スタジオには、プロジェクトをテストするためのツールがその他にも多数含まれています ([Run-Test] パースペクティブにもあります)。

プロジェクトの実行、構成、モニタリング、クエリ、デバッグに関する、このガイドで扱われている内容以上の詳細については、『スタジオ・ユーザーズ・ガイド』を参照してください。

索引

記号

A

C

E

G

O

S

V

あ

い

ひ

ふ

め

ゆ

れ

わ