SYBASE®
An SAP® Company

スタジオ・ユーザーズ・ガイド

Sybase Event Stream Processor 5.0

ドキュメント ID： DC01738-01-0500-01
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。

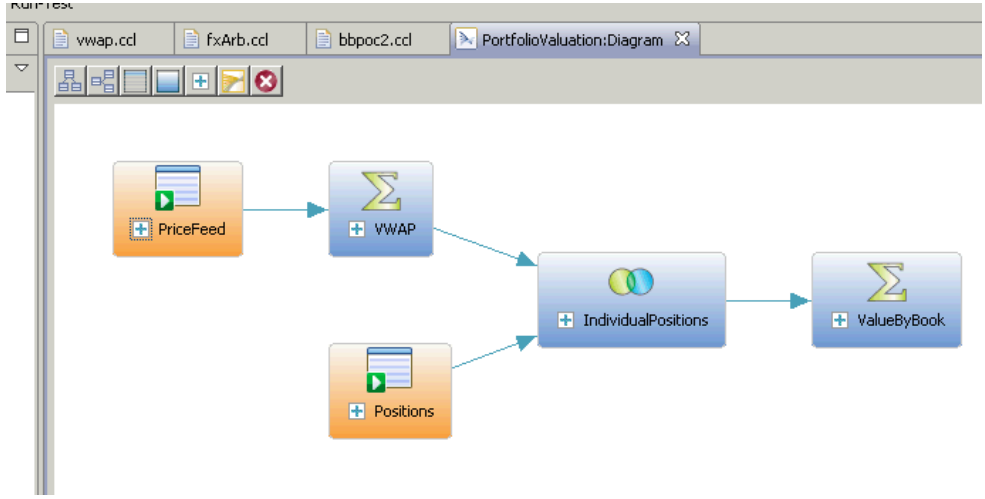
Event Stream Processor は、インメモリ・データベースではありません。インメモリ・データベースは、要求される処理速度を達成するために、メモリ内で動作し、すべてのデータをメモリに保持します。Event Stream Processor は、インメモリ・データベースにいくつかの点で類似していますが、インメモリ・データベースと異なり、オンデマンド・クエリを効果的に処理するように設計されており、連続するイベント駆動型クエリに最適化されたデータフロー・アーキテクチャを使用します。

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。

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。

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。

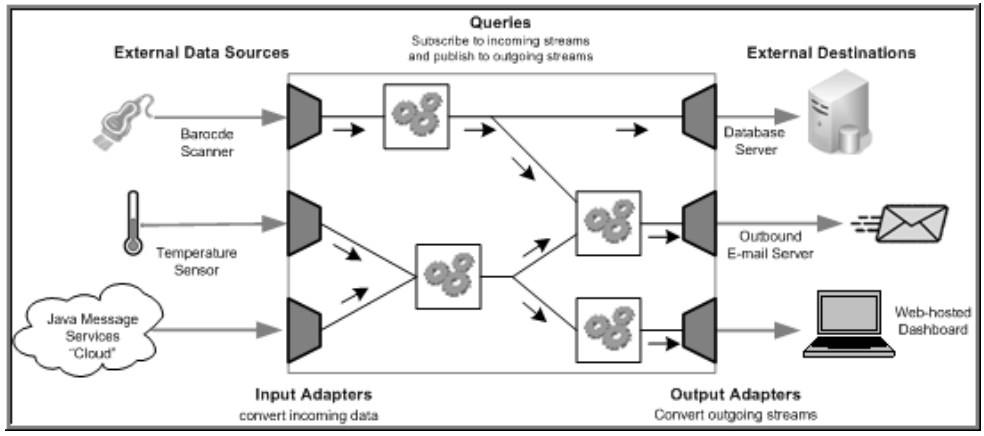
外部ソースまたは外部送信先には、以下があります。

- 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ダウン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。

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。

参照：

- *Flex 演算子* (61 ページ)

第 2 章 ESP スタジオの起動

プロジェクトの作成を開始するには、ESP スタジオを起動して、ワークスペースの基本を確認し、必要に応じてサンプルを実行してから独自のプロジェクトを作成します。

ESP スタジオの起動

デスクトップのショートカット、Windows の [スタート] メニュー、またはコマンド・ラインから、ESP スタジオを起動します。
デスクトップまたはワークステーションから起動するには、次の順序に従います。

プラットフォーム	方法
Windows	• コンピュータ・デスクトップ上の [Sybase ESP Studio] ショートカットをダブルクリックします。または、 • [スタート] > [すべてのプログラム] > [Sybase] > [Event Stream Processor] > [Studio] > [Studio] を選択します。
Linux または UNIX	• コンピュータ・デスクトップ上の [Sybase ESP Studio] ショートカットをダブルクリックします。または、 • コマンド・ラインに `$ESP_HOME/studio/esp-studio/esp_studio` と入力します。

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

参照：

- *第 3 章、「ビジュアル・エディタのオーサリング」* (21 ページ)
- *第 6 章、「プロジェクトの実行とテスト」* (83 ページ)

File Explorer

File Explorer を使用してプロジェクトを編成したり、プロジェクト間を移動したりします。File Explorer では、フォルダとファイルのツリー構造の階層が表示されます。

[File Explorer] ビューでは、プロジェクト・ファイルの編成、ファイルへの移動のほかに、ファイルを基準とするさまざまなアクションを実行できます。

- 新規 CCL ファイルの作成
- 新規プロジェクトの作成
- 既存のファイルの編集
- ファイルの削除
- 新規フォルダの作成

参照：
- *ビジュアル・エディタでのプロジェクトの編集* (27 ページ)
- *CCL エディタでの編集* (67 ページ)

[Learning] パースペクティブ

[Learning] パースペクティブでは、スタジオを使い始める場合にサンプル・プロジェクトを使用して一般的な作業を実行できます。

[Learning] パースペクティブは、次の 3 つの異なる方法で開くことができます。

- [Welcome] 画面の [Open Example] ショートカット
- パースペクティブのショートカット・バーの [Learning] ボタン
- [Window] > [Open Perspective] > [Learning] を選択する

[Learning] パースペクティブの左側の [Examples] ビューには、現在使用可能なすべてのサンプルがリストされます。各サンプル項目の [LOAD] ボタンをクリックすると、サンプルが実行されます。

[Learning] パースペクティブの右側の [Description] ビューには、選択したサンプルの詳細な説明が表示されます。

[Learning] パースペクティブでのサンプルの実行

付属のサンプルをロードして実行し、[Server] ビュー、[Stream] ビュー、ビジュアル・エディタなど、スタジオの重要な機能をデモンストレーションします。

1. [Examples] ビューでサンプル・プロジェクトを選択して、[LOAD] をクリックします。

2. 次のどちらかを行います。
   - [Proceed] をクリックしてサンプル・プロジェクトを開始し、プロンプトに従って残りの手順を実行します。または、
   - [Run in silent mode] をオンにして、[Proceed] をクリックし、バックグラウンドでプロセスを実行します。

   この手順が完了すると、サンプル・プロジェクトがビジュアル・エディタに表示されます。
3. [Proceed] をクリックしてサンプル・ストリームをサブスクライブします。ストリームが [Stream] ビューに表示されます。
4. [Proceed] をクリックしてサンプル・データをパブリッシュし、サーバにアップロードします。
5. (オプション) [Step by Step Example] ビューの [Example project] メニューで、プロジェクトを選択します。

   プロジェクトの名前がワークスペース内の既存のプロジェクトと同じであると、スタジオは既存のプロジェクトもサンプルかどうかを判断します。サンプルの場合、スタジオはプロジェクトをロードします。既存のプロジェクトがサンプルではないとエラーになるので、最初から選択されていたサンプル・プロジェクトの名前を変更するか、エラーの原因となった既存のプロジェクトを削除します。
6. (オプション) ステップをクリックして、これまでのアクションを確認します。

   [Step by Step Example] ビューに、サンプルで実行されるアクションがリストされ、アクションを起動するためのクイック・リンクが表示されます。アクションを起動するには、プロジェクトを実行しておきます。

プロジェクトの作成

スタジオを使用して、ESP サーバで実行できる新しいプロジェクトを作成します。

継続クエリはプロジェクトで構成され、入力、出力に加え、スキーマなど、イベント・データを処理するためのオプションも定義します。

1. [File] > [New] > [Project...] を選択します。
2. 次のように、有効なプロジェクト名を入力します。
   - 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。

   たとえば、`MyFirstProject` と入力します。

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。

参照：
- *プロジェクトを開く* (18 ページ)
- *既存のプロジェクトのインポート* (19 ページ)
- *ビジュアル・エディタでのプロジェクトの編集* (27 ページ)
- *CCL エディタとビジュアル・エディタ間の切り替え* (65 ページ)

AleriML モデルから CCL プロジェクトへの変換

スタジオでは、AleriML データ・モデルを新しい CCL プロジェクトに変換するか、データを既存のプロジェクトに追加できます。

変換エラーが発生するとダイアログ・ボックスに表示されます。各エラーは、行情報とカラム情報と共に、別々の行に表示されます。

Aleri モデルと ESP プロジェクトの相違点については、『移行ガイド』を参照してください。

AleriML モデルから新規プロジェクトへの変換

[File] メニューで、新規プロジェクトの AleriML 変換機能にアクセスします。

1. スタジオで任意のビューから、[File] メニューを開きます。
2. [Convert Aleri Data Model] を選択します。データ・モデルを新しい CCL プロジェクトに変換するか、データ・ファイルを既存のプロジェクトに追加するかを選択します。
3. [Convert to new project] を選択して、[Next] をクリックします。
4. [Aleri data model] で、変換するデータ・モデルの名前を選択するか入力します。選択したモデルに基づいて、[CCL file name] フィールドと [Project name] フィールドにデータが挿入されます。

CCL ファイル名とプロジェクト名は、データがフィールドに挿入された後で上書きできます。

5. [Location] でデフォルトの場所をそのまま使用するか、新規プロジェクト・フォルダの保存先ディレクトリを選択します。
6. [Finish] をクリックして変換を完了します。

参照：

- *既存のプロジェクトに追加する AleriML モデルの変換* (18 ページ)

既存のプロジェクトに追加する AleriML モデルの変換

[File] メニューまたは File Explorer から、既存のプロジェクトの AleriML 変換機能にアクセスします。

1. 次のどちらかを行います。
   - スタジオの任意のパースペクティブから [File] メニューを開きます。または、
   - [Authoring] パースペクティブの [File Explorer] ビューでプロジェクトを右クリックします。
2. [Convert Aleri Data Model] を選択します。[File] メニューから変換オプションにアクセスした場合は、[Convert to existing project] を選択して [Next] をクリックします。

   注意： プロジェクトは現在のワークスペースに配置しておきます。
3. [Aleri data model] で、データ・モデルの名前を選択するか入力します。

   モデルの名前に基づいて、[CCL file name] フィールドと [Project name] フィールドにデータが挿入されます。
4. [Finish] をクリックして変換を完了します。

参照：

- *AleriML モデルから新規プロジェクトへの変換* (17 ページ)

プロジェクトを開く

Event Stream Processor のプロジェクトがワークスペースに既にある場合は、File Explorer から開きます。

1. File Explorer で、プロジェクト・フォルダを展開してプロジェクト・ファイルを表示します。
2. ファイルをダブルクリックするとファイルが開き、編集できるようになります。

- `.cclnotation` ファイルはビジュアル・エディタに表示されます。
- `.ccl` ファイルは CCL エディタに表示されます。

同じプロジェクトの `.cclnotation` ファイルと `.ccl` ファイルの両方を同時に開いておくことはできません。

参照：
- *プロジェクトの作成* (16 ページ)
- *既存のプロジェクトのインポート* (19 ページ)
- *ビジュアル・エディタでのプロジェクトの編集* (27 ページ)
- *CCL エディタとビジュアル・エディタ間の切り替え* (65 ページ)

既存のプロジェクトのインポート

別の場所にある Event Stream Processor のプロジェクトをワークスペースにインポートします。

1. [File] > [Open] > [Project] を選択します。
2. プロジェクトのルート・ディレクトリを見つけて移動します。
3. (オプション) [Copy projects into workspace] を選択します。
   - [Copy projects into workspace] を選択すると、プロジェクトがワークスペースにコピーされ、そこから表示されます。変更はコピーのみに加えられます。
   - このオプションがオフの場合は、元の場所にあるプロジェクトが表示されます。
4. [Finish] をクリックします。

参照：
- *プロジェクトの作成* (16 ページ)
- *プロジェクトを開く* (18 ページ)
- *ビジュアル・エディタでのプロジェクトの編集* (27 ページ)
- *CCL エディタとビジュアル・エディタ間の切り替え* (65 ページ)

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。

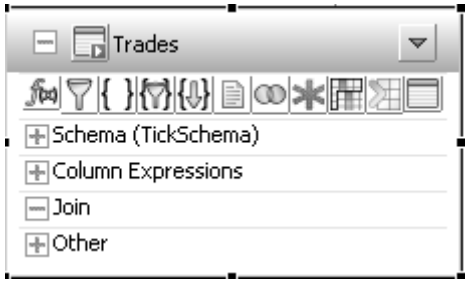
各プロジェクトには、少なくとも 1 つの図があります。1 つのプロジェクトに複数の図を設定できます。プロジェクト間で図を共有することはできません。

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。

参照：

- *図形のリファレンス* (24 ページ)
- *図の表示変更* (29 ページ)

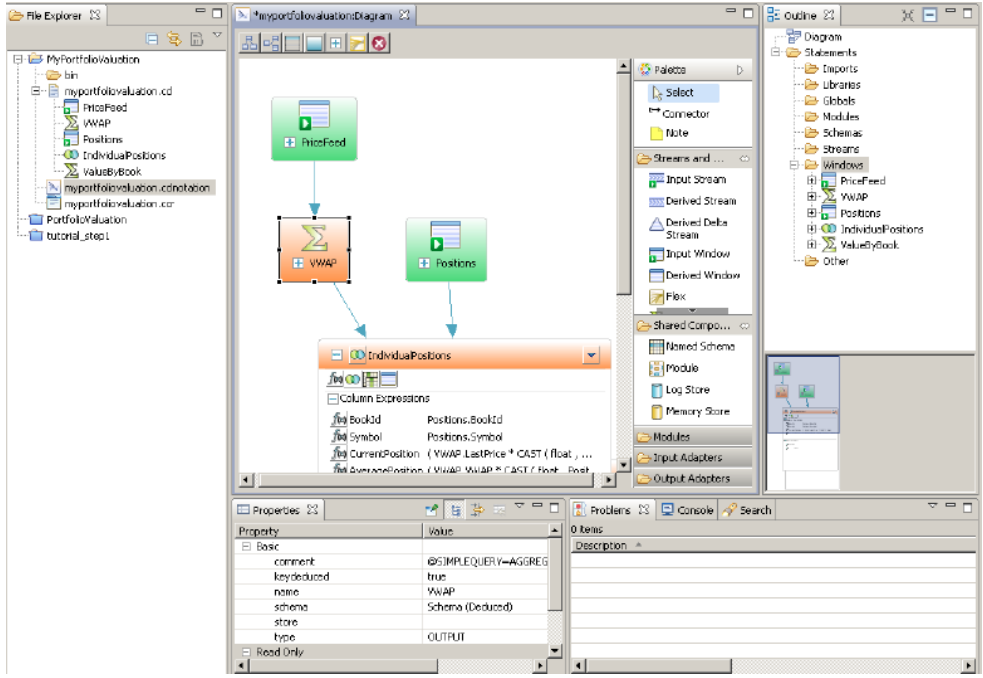
ビジュアル・オーサリング環境

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



図形のリファレンス

パレット内の各図形では、特定のタイプのストリームまたはウィンドウ、アダプタ、接続、再利用可能なスキーマまたはモジュール、ストアが作成され、これらでデータ・フローが作成されます。

表 2 : パレット内の図形

図形	目的	使用法
Connector	ストリームやウィンドウ間のフローを作成します。また、ストリームや共有コンポーネント間の集約を有効化したり、図形の間に注釈をアタッチしたりします。	図内で図形を接続できるのは、基本となる CCL で接続が許可されている場合にかぎります。
Note	図上でのみコメントを作成します。このコメントは CCL ファイルには表示されません。	
Input Stream	入力アダプタまたはパブリッシャからの受信データに継続クエリを適用します。	ストリームはデータを保持しないため、ストアはありません。入力ストリームのデータにはキーが設定されません。
Derived Stream Derived Stream (Output)	別のストリームまたは入力ウィンドウからの受信データに継続クエリを適用して新しいストリームを生成します。	ストリームはデータを保持しないため、キーはありません。ストリームは「挿入のみ」です。つまり、出力は挿入のみで構成されます。入力はストリームまたはストリームとウィンドウのジョインとします。 デフォルトでは、新しいストリームは (派生ストリームを含め) ローカルですが、そのプロパティを出力に変更して外部サブスクライバで表示できるようになります。
Derived Window Derived Window (Output)	別のストリームまたはウィンドウからの受信データに継続クエリを適用します。データを保持するので、保持ルールを設定できます。	すべてのローにユニークなキーが設定されるようにデータにキーを設定します。挿入、更新、削除を、ローカルと出力の両方として処理します。ツールバーを使用してウィンドウを出力に変更して、外部クライアントで表示できるようになります。

図形	目的	使用法
 Derived Delta Stream  Derived Delta Stream (Output)	状態を保持する必要はないが、挿入、更新、削除の各操作は保存する必要があるウィンドウから下流のストリームに継続クエリを適用します。	計算、フィルタ処理、またはユニオンを実行する必要があるが、状態は保持する必要がない場合に使用できます。必要に応じて、ツールバーを使用して派生デルタ・ストリームを出力に変更します。
 Input Window	ストリームが入力ストリームから受信するデータの量を制限します。これは、次の場合にイベント・ストリームのエントリ・ポイントとして使用します。 • ストリームに、挿入、更新、削除の各イベントがある場合。または、 • 一連の受信イベントを保持する必要がある場合。	ウィンドウのサイズは、入力レコードの固定数を使用したロー数、またはレコードを保持する指定期間で設定できます。ウィンドウにはキーを設定します。つまり、すべてのローにユニークなキー値を設定します。
 Flex	カスタムの SPLASH スクリプトを使用して受信イベントを処理するプログラム可能な演算子。	
 Aggregate	1 つのソースから入力を取得し、共通の属性を使用してレコードをグループ化します。グループごとに 1 つの出力レコードを生成します。sum()、count() などのような集合関数を使用します。	常に新しいウィンドウが作成されます。GROUP BY 要素とプライマリ・キーが必要です。必要に応じて、保持ルールを使用してウィンドウのサイズを設定できます。
 Compute	1 つのソースから入力を取得し、受信したレコードごとに新しいレコードを計算します。イベントのスキーマを変更して、新しいフィールドを計算したり既存のフィールドを変更したりできます。	入力がストリームの場合は派生ストリームが生成されます。入力がウィンドウの場合は派生デルタ・ストリームが生成されます。
 Filter	1 つのソースから入力を取得し、フィルタを適用します。フィルタ条件に一致するレコードのストリームを作成します。	入力がストリームの場合は派生ストリームが生成されます。入力がウィンドウの場合は派生デルタ・ストリームが生成されます。

図形	目的	使用法
 Join	2 つ以上のソースから入力を取得し、共通のデータ要素に基づいてそれらをジョインします。	サポートされるジョインの詳細については、このマニュアルと『CCL プログラマーズ・ガイド』の関連情報を参照してください。
 Pattern	2 つ以上のソースから入力を取得し、イベントのパターンを検出します。パターンが検出されるたびに 1 つの出力レコードが生成されます。	
 Union	2 つ以上のソースからの入力をマージします。入力レコードごとに 1 つの出力レコードが生成されます。	すべての入力が共通のスキーマを持っています。
 Named Schema	ストリームとウィンドウで参照できるカラム構造の再利用可能な定義。	
 Module	CCL **CREATE MODULE** 文を表します。プロジェクト内の 1 つ以上の場所で使用できる新しいモジュールを作成します。	モジュールには、プロジェクトと同じ要素をすべて入れることができます。モジュールは再利用可能です。
 Log Store	ウィンドウに保持されているデータを保存します。ディスクベースでリカバリできますが、メモリ・ストアより低速です。	デフォルトでは、新しいウィンドウはメモリ・ストアに割り当てられます。ウィンドウ内のデータをリカバリ可能にする必要がある場合は、ログ・ストアを作成してウィンドウを割り当てます。
 Memory Store	ウィンドウに保持されているデータを保存します。	ログ・ストアより高速ですが、シャットダウン後にデータをリカバリできません。 • (デフォルト) 他のストアが指定されていない場合、CCL コンパイラによって暗黙的に作成されます。 • (オプション) パフォーマンスを最適化するには、明示的に作成し、特定のストアにウィンドウを割り当てます。

図形	目的	使用法
Input Adapters	入力ストリームまたは入力ウィンドウを外部データ・ソースに接続します。	入力ストリームまたは入力ウィンドウのどちらかに接続します。スキーマ検出を使用する、つまりソースからスキーマをインポートするには、入力アダプタを追加してから、スキーマ検出を使用し、インポートしたスキーマを使用して、接続先の入力ストリームまたは入力ウィンドウを作成します。
Output Adapters	出力ストリームまたは出力ウィンドウを送信先に接続します。	出力ストリームまたは出力ウィンドウのどちらかに接続します。

参照：

- *単純なクエリ* (36 ページ)
- *図形の図への追加* (28 ページ)
- *要素の接続* (47 ページ)
- *ジョインのサポート* (42 ページ)

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：

- *プロジェクトの作成* (16 ページ)
- *プロジェクトを開く* (18 ページ)
- *既存のプロジェクトのインポート* (19 ページ)
- *CCL エディタとビジュアル・エディタ間の切り替え* (65 ページ)
- *File Explorer* (15 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。

参照：
- *単純なクエリ* (36 ページ)
- *図形のリファレンス* (24 ページ)
- *要素の削除* (50 ページ)
- *ビジュアル・エディタでのキーボード・ショートカット* (65 ページ)

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。

参照：
- *スタジオ環境設定の編集* (115 ページ)

単純なプロジェクトの作成

次の手順に従って、ESP スタジオ・ビジュアル・エディタで単純なプロジェクトを完全に作成します。各作業をクリックすると、リンク先にジャンプします。

前提条件
プロジェクトを作成します。

手順
一部の作業は省略可能です。作業の順序は厳密ではなく、プロジェクトごとに詳細は異なります。

ヒント： 入力から開始し、データ・フローに従って、左から右、または上から下に作業を進めます。この方式を使用して、カラムとカラム式を入力ストリームから新しいクエリにコピーできます。

1. *プロジェクトへのアダプタの追加*
   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。
2. *スキーマの検出*
   ビジュアル・エディタの Schema Discovery ボタンを使用して、アダプタからデータのフォーマットに基づいてスキーマを検出し (自動的に) 作成します。
3. *プロジェクトへの入力ストリームまたは入力ウィンドウの追加*
   入力ストリームと入力ウィンドウは、プロジェクトの外部のソースからデータを受け取ります。
4. *単純なクエリの追加*
   希望するタイプの単純なクエリを選択し、ビジュアル・エディタのツールを使用してそのクエリを作成します。
5. *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。
6. *キー・カラムの設定*
   ビジュアル・エディタのデルタ・ストリーム、ウィンドウ、Flex 演算子の各図形の [Column] コンパートメントで、プライマリ・キーを設定します。
7. *ウィンドウ、ストリーム、デルタ・ストリームのカラム式の編集*
   インライン・エディタまたはダイアログベースの式エディタを使用して、ウィンドウ、ストリーム、デルタ・ストリームのカラム式を変更します。

参照：
- *要素の削除* (50 ページ)
- *プロジェクトの作成* (16 ページ)

プロジェクトへのアダプタの追加

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。

1. パレットの [Input Adapters] または [Output Adapters] のコンパートメントを開き、上下の各矢印を使用してアダプタのリストをスクロールします。

2. パレットでアダプタの図形をクリックしてから、図の中をクリックします。
3. アダプタをストリームまたはウィンドウにアタッチします。次のどちらかを行います。
   - スキーマ検出を使用して、ストリームまたはウィンドウを自動的に生成してアタッチします (この方法は、スキーマ検出をサポートするアダプタにおすすめします)。または、
   - ストリームまたはウィンドウを作成してから、次のアダプタにアタッチします。
     - **入力アダプタ** – [Connector] ツールをクリックし、図で [Adapter] の図形をクリックしてから、ストリームまたはウィンドウをクリックします。
     - **出力アダプタ** – [Connector] ツールをクリックし、図でストリームまたはウィンドウをクリックしてから、[Adapter] の図形をクリックします。
4. (オプション) アダプタの名前を編集します。
5. (オプション) アダプタのプロパティを編集します。次のどちらかを行います。
   - [Use named property set] を選択して、プロジェクト設定ファイルにある名前付きプロパティ・セットを使用してから、そのプロパティ・セットに含まれていないプロパティを設定します。または、
   - [Set properties locally] を選択して、アダプタ・プロパティを手動で設定します。

参照：
- *スキーマの検出* (32 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：
- *付録 A、「スキーマ検出をサポートするアダプタ」* (119 ページ)
- *スキーマの検出* (32 ページ)

スキーマの検出

ビジュアル・エディタの [Schema Discovery] ボタンを使用して、アダプタからデータのフォーマットに基づいてスキーマを検出し (自動的に) 作成します。

前提条件
アダプタを図に追加します。

手順
[Authoring] パースペクティブで、次の作業を行います。

1. スキーマ検出用のアダプタを設定します。アダプタの図形で、[Edit Properties] をクリックしてダイアログに入力します。
   - 名前付きプロパティ・セットを選択します。または、
   - [Set properties locally] を選択して [Basic] タブと (必要に応じて) [Advanced] タブにプロパティ値を入力します。必須プロパティは赤で表示されます。

   たとえば、CSV ファイル・インプット・アダプタのスキーマ検出を使用するには、まずアダプタの [Directory] プロパティと [File] プロパティを設定して、アダプタが読み取る必要があるデータ・ファイルの絶対パスを指定します。

   注意： 名前付きプロパティ・セットを作成するには、プロジェクト設定ファイルでアダプタのプロパティを編集します。
2. アダプタ・ツールバーの [Schema Discovery] をクリックします。
   - スキーマが正常に検出されると、ダイアログが表示され、そこでスキーマを確認して選択できます。
   - 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。この場合は、次の作業を実行できます。
     - スキーマ検出のためにアダプタのプロパティが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。
3. スキーマを選択して [Next] をクリックします。
4. 要素を作成するためのダイアログで、オプションを選択します。

アダプタの状態	使用可能なオプション
アダプタはストリームにもウィンドウにもアタッチされていない。	• **[Create a new input stream.] –** 新しいストリームを作成してアダプタにアタッチし、ストリームのインライン・スキーマを作成して、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を設定します。 • **[Create a new input window.] –** 新しいウィンドウを作成してアダプタにアタッチし、ウィンドウのインライン・スキーマを作成して、アダプタから検出されたウィンドウを使用してウィンドウを設定します。 • **[Create a new named schema.] –** 新しい名前付きスキーマを作成し、アダプタから検出されたスキーマを使用して設定します。
アダプタはストリームまたはウィンドウに既にアタッチされている。	• **[Apply the schema to the connecting stream or window.] –** 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またはウィンドウを設定します。 • **[Create a new named schema.] –** 新しい名前付きスキーマを作成し、アダプタから検出されたスキーマを使用して設定します。

5. [Finish] をクリックします。

参照：

- *スキーマ検出* (31 ページ)
- *付録 A、「スキーマ検出をサポートするアダプタ」* (119 ページ)
- *プロジェクトへのアダプタの追加* (30 ページ)

プロジェクトへの入力ストリームまたは入力ウィンドウの追加

入力ストリームと入力ウィンドウは、プロジェクトの外部のソースからデータを受け取ります。

入力ストリームまたは入力ウィンドウを作成するには、スキーマ検出をサポートするアダプタを追加して、外部データ・ソースのスキーマを自動的に継承するストリームまたはウィンドウを生成します。必要に応じて、カラムを追加できます。

1. ビジュアル・エディタのワークスペースの [Streams and Windows] カテゴリーの下にある [Palette] メニューで、次のどちらかを選択します。
   - [Input Stream] 
   - [Input Window] 
2. 図でロケーションを選択し、クリックすると、図形が追加されます。

3. 次のどちらかの手順に従って、入力ストリームまたは入力ウィンドウの名前を設定します。
   - 図形の名前をクリックして編集します。または、
   - 冗長モードで、名前の隣にある [Edit] アイコンをクリックします。
4. [Add Column] をクリックして新しい各カラムをスキーマに追加してから、キー・カラムを設定し、カラム式を編集します。
5. カラムを削除するには、カラムを選択して [Delete] キーを押します。
6. (ウィンドウの場合は省略可能、ストリームの場合は実行不可) [Set Keep Policy] を選択してオプションを選びます。
7. (オプション) ポリシーをダブルクリックしてパラメータを編集します。

参照：
- *単純なクエリの追加* (36 ページ)
- *keep ポリシー* (34 ページ)
- *ウィンドウ、ストリーム、デルタ・ストリームのカラム式の編集* (48 ページ)
- *エイジング・ポリシーの設定* (62 ページ)

keep ポリシー

keep ポリシーによって、ウィンドウにローを保持する基準を指定します。

ウィンドウを生成する単純なクエリを含め、メモリベースのストアを使用するウィンドウには、keep ポリシーを設定できます。これは、保存ポリシーとも呼ばれます。

入力ウィンドウでは、ログ・ファイルベースのストアでの保持のみがサポートされます。

表 3 : keep ポリシーのオプション

オプション	説明
All rows	ウィンドウにすべてのローを保持します (デフォルト)。
Last row	ウィンドウに最後のローのみを保持します。

オプション	説明

Count	次のどちらかを行います。
	• 保持するローの絶対数を入力します。または、
	• [Select] を選んで、これまでに宣言した変数またはパラメータを選択して、ウィンドウに保持する特定の範囲のローを指定します。
	ヒント： リストが空の場合に、カウントの基準をパラメータまたは変数にするには、CCL エディタに切り替えて、CCL の先頭の DECLARE ブロックでパラメータまたは変数を定義します。たとえば、次のように定義します。
	`DECLARE`
	`integer test :=50;`
	`end;`
	戻って、それを選択します。

Slack	カウント基準のポリシーでは、ローが最大数 ([Count] 値) に達した場合に削除するローの数を設定します。デフォルトは 1 です。つまり、ウィンドウ内の行数が *count_value* になると、新しいローが追加されるたびに最も古いローが削除されます。
Time	ウィンドウでの時間制限を設定して、ウィンドウでのローの保持時間を決める期間を指定します。[Ctrl] キーを押しながら [Space] キーを押して、時間の単位を選びます。

スラック
スラックとは、メモリ・ストアに必要なクリーニングの頻度を下げてパフォーマンスを向上させるための高度な機能です。スラックでは、ウィンドウ内の最大ロー数として N + S を設定します。ここで、N は保持サイズ (カウントの設定) で、S はスラックです。ウィンドウ内のロー数が N + S に達すると、システムは S 個のローを消去します。スラックの値が大きいほど、必要なクリーニングの回数が減るので、パフォーマンスが向上します。

スラックのデフォルト値は 1 です。スラックが 1 の場合は、ウィンドウ内のレコードが最大数に達すると、新しいレコードが挿入されるたびに最も古いレコードが削除されます。これは、パフォーマンスに重大な影響を与えます。スラックが 1 より大きい、たとえば Y の場合、ウィンドウには最大 X + Y 個のレコードが累積されます。次のレコードが挿入されると、Y 個のレコードが削除されます。スラックの値が大きいほど、ローを絶えず削除する必要が減るので、パフォーマンスが向上します。

参照：
- *単純なクエリの作成と変更：Aggregate* (39 ページ)
- *単純なクエリの作成と変更：Join* (41 ページ)

単純なクエリの追加

希望するタイプの単純なクエリを選択し、ビジュアル・エディタのツールを使用してそのクエリを作成します。

参照：

- *単純なクエリの作成と変更：Filter* (38 ページ)
- *単純なクエリの作成と変更：Aggregate* (39 ページ)
- *単純なクエリの作成と変更：Compute* (40 ページ)
- *単純なクエリの作成と変更：Join* (41 ページ)
- *単純なクエリの作成と変更：Union* (45 ページ)
- *単純なクエリの作成と変更：Pattern* (46 ページ)
- *単純なクエリ* (36 ページ)
- *プロジェクトへの入力ストリームまたは入力ウィンドウの追加* (33 ページ)
- *要素の接続* (47 ページ)

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。

表 4 : 単純なクエリと同等の CCL (概要)

単純なクエリ	CCL
Filter	WHERE 句
Aggregate	GROUP BY 句
Join	FROM 句
Compute	SELECT FROM 句と DELTA STREAM
Union	UNION 句
Pattern	MATCHING 句

CCL 文から単純なクエリへの変換
CCL でクエリを作成してから、それらをビジュアル・エディタで単純なクエリのタイプとして表示するには、**CREATE STREAM** 文、**CREATE WINDOW** 文、または **CREATE DELTA STREAM** 文の直前に、次の形式でコメントを挿入します。

```
/**@SIMPLEQUERY=QUERY_TYPE*/
```

ここで、*QUERY_TYPE* はビジュアル・エディタに表示する図形の名前です。

たとえば、コメント `/**@SIMPLEQUERY=AGGREGATE*/` を挿入すると、**CREATE WINDOW** 文はビジュアル・エディタで [Aggregate] の図形にマップされます。

CREATE WINDOW 文の直前にこのコメントがないと、ビジュアル・エディタには汎用の [Derived Window] の図形が表示されます。

注意： CCL エディタとビジュアル・エディタで同時に CCL コードを変更することはできません。 ビジュアル・エディタが開いている場合、CCL エディタは読み取り専用になります。

単純なクエリから CCL 文への変換
パレットで単純なクエリを作成した場合、それによって作成される CCL 要素は次のルールに基づきます。

- フィルタ・オブジェクトの入力がストリームの場合、フィルタ・オブジェクトではストリームが作成されます。ソースがウィンドウ、デルタ・ストリーム、またはフレックス・ストリームの場合、フィルタ・オブジェクトではデルタ・ストリームが作成されます。
- 集約オブジェクトではすべてウィンドウが作成されます。
- 計算オブジェクトの入力がストリームの場合、計算オブジェクトではストリームが作成されます。ソースがウィンドウ、デルタ・ストリーム、またはフレックス・ストリームの場合、計算オブジェクトではデルタ・ストリームが作成されます。

- ジョイン・オブジェクトがストリームからのみ入力を取得する場合、ジョイン・オブジェクトではストリームが作成されます。ソースが 1 つ以上のウィンドウ、デルタ・ストリーム、またはフレックス・ストリームの場合、ジョイン・オブジェクトではウィンドウが作成されます。ストリームとウィンドウのジョインの場合、ジョイン・オブジェクトではストリームが作成されます。
- ユニオン・オブジェクトの入力がストリームの場合、ユニオン・オブジェクトではストリームが作成されます。ソースがウィンドウ、デルタ・ストリーム、またはフレックス・ストリームの場合、ユニオン・オブジェクトではデルタ・ストリームが作成されます。
- パターン・オブジェクトではすべてストリームが作成されます。

参照：
- *図形のリファレンス* (24 ページ)
- *図形の図への追加* (28 ページ)
- *要素の接続* (47 ページ)
- *CCL でのクエリ* (70 ページ)
- *単純なクエリの作成と変更：Filter* (38 ページ)
- *単純なクエリの作成と変更：Aggregate* (39 ページ)
- *単純なクエリの作成と変更：Compute* (40 ページ)
- *単純なクエリの作成と変更：Join* (41 ページ)
- *単純なクエリの作成と変更：Union* (45 ページ)
- *単純なクエリの作成と変更：Pattern* (46 ページ)

単純なクエリの作成と変更：Filter

特定の特性を持つイベントのみを渡す単純なクエリを生成します。フィルタでは、CCL の **WHERE** 句を使用します。

1. ビジュアル・エディタのパレットの [Streams and Windows] で、[Filter] ( ) をクリックします。
2. 図でロケーションを選択し、クリックすると、図形が追加されます。
3. フィルタ・オブジェクトを適切なストリームまたはウィンドウにアタッチします。

   フィルタ・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。フィルタ・オブジェクトに割り当てることのできる入力は 1 つのみです。
4. フィルタ式の値を編集するには、値を選択し、必要に応じて変更します。デフォルト値は 1 です。

   「1」に評価される式は true で、すべてのレコードが渡されます。0 (ゼロ) は false です。

5. (オプション) [Toggle] オプションを使用して、フィルタ・オブジェクトを [LOCAL] または [OUTPUT] として指定します。

参照：
- *単純なクエリの作成と変更：Aggregate* (39 ページ)
- *単純なクエリの作成と変更：Compute* (40 ページ)
- *単純なクエリの作成と変更：Join* (41 ページ)
- *単純なクエリの作成と変更：Union* (45 ページ)
- *単純なクエリの作成と変更：Pattern* (46 ページ)
- *単純なクエリ* (36 ページ)

単純なクエリの作成と変更：Aggregate

データを組み合わせる単純なクエリを生成します。CCL の **GROUP BY** 句、**GROUP FILTER** 句、**GROUP ORDER** 句と類似しています。

1. ビジュアル・エディタのパレットの [Streams and Windows] で、[Aggregate] ( ) を選択します。
2. 図でロケーションを選択し、クリックすると、図形が追加されます。
3. [Aggregate] の図形を入力に接続します。
   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。
4. 次のように、カラムを追加します。
   a) 図形ツールバーの [Copy Columns from Input] ( ) をクリックして、[Aggregate] 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。
   c) カラム式を編集するには、ダブルクリックしてインライン・エディタを開きます。または、式を選択して、[Ctrl] キーを押しながら [F2] キーを押すと、選択した式がポップアップ・エディタに表示されます。
5. 図形ツールバーの [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。

6. (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。これによって **KEEP** 句が定義されます。

7. (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。

参照：
- *単純なクエリの作成と変更：Filter* (38 ページ)
- *単純なクエリの作成と変更：Compute* (40 ページ)
- *単純なクエリの作成と変更：Join* (41 ページ)
- *単純なクエリの作成と変更：Union* (45 ページ)
- *単純なクエリの作成と変更：Pattern* (46 ページ)
- *単純なクエリ* (36 ページ)
- *keep ポリシー* (34 ページ)

単純なクエリの作成と変更：Compute

各受信レコードのスキーマまたはフィールドの値を変換する単純なクエリを生成します。受信イベントごとに、カラム式で定義された複数のフィールドから 1 つの新しい出力イベントが生成されます。

1. ビジュアル・エディタのパレットの [Streams and Windows] で、[Compute] ( ) を選択します。
2. 図でロケーションを選択し、クリックすると、図形が追加されます。
3. 計算オブジェクトを、このクエリに入力を提供するストリームまたはウィンドウにアタッチします。

   計算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。計算オブジェクトに割り当てることのできる入力は 1 つのみです。複数の入力ソースに接続しようとすると、ブロックされます。
4. 次のように、カラムを追加します。
   a) 図形ツールバーの [Copy Columns from Input] ( ) をクリックして、このクエリのスキーマに入力フィールドをコピーします。
   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。
   c) カラム式を編集するには、ダブルクリックしてインライン・エディタを開きます。または、式を選択して、[Ctrl] キーを押しながら [F2] キーを押すと、選択した式がポップアップ・エディタに表示されます。
5. 必要に応じて、カラム式 を追加します。
6. カラム式を変更するには、カラム式を選択して直接変更するか、[Properties] ビューで対応するフィールドを編集します。
7. [Toggle] オプションを使用して、計算オブジェクトを [LOCAL] または [INPUT] として指定します。

参照：
- *単純なクエリの作成と変更：Filter* (38 ページ)
- *単純なクエリの作成と変更：Aggregate* (39 ページ)
- *単純なクエリの作成と変更：Join* (41 ページ)
- *単純なクエリの作成と変更：Union* (45 ページ)
- *単純なクエリの作成と変更：Pattern* (46 ページ)
- *単純なクエリ* (36 ページ)

単純なクエリの作成と変更：Join

複数の入力イベントのフィールドを組み合わせて 1 つの出力イベントにする単純なクエリを生成します。

1. ビジュアル・エディタのパレットの [Streams and Windows] で、[Join] ( ) を選択します。
2. 図でロケーションを選択し、クリックすると、図形が追加されます。
3. ジョイン・オブジェクトを、ジョインに入力を提供する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 2 つ以上のストリーム、ウィンドウ、または Flex 演算子に接続します。ジョイン・オブジェクトは 2 つ以上のオブジェクトから入力を取得できますが、生成できる出力は 1 つのみです。

   注意： ジョインには、ストリーム、ウィンドウ、デルタ・ストリームを含めることができます。ただし、ジョインにデルタ・ストリームを含めることができるのは **KEEP** 句が指定され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。

   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。
4. [Copy Columns from Input] を使用して、このクエリの出力に含める入力フィールドを選択します。
5. 必要に応じて、カラム式 を追加します。
6. カラム式を編集するには、ダブルクリックしてインライン・エディタを開きます。または、式を選択して、[Ctrl] キーを押しながら [F2] キーを押すと、選択した式がポップアップ・エディタに表示されます。
   または、[Properties] ビューで対応するフィールドを編集します。
7. [Add Join Condition] をクリックして、複数の異なるソースにまたがる受信イベントの一致に使用するカラムを指定します。

[Edit Join Expression] ダイアログに入力して、ジョイン・タイプ、ON 句のデータ・ソース、その他のジョイン制約を定義します。

必要なカラムが [Edit Join Expression] ダイアログに表示されない場合は、ジョイン・オブジェクトが正しい入力ソースに接続されていることを確認します。

8. カラムをそれ自体にジョインするには、図形ツールバーの [Add Input Alias] をクリックします。

   各ジョイン条件にユニークな名前を指定するには、カラムのエイリアスが必要です。
9. (オプション) [Toggle] オプションを使用して、ジョイン・オブジェクトを [LOCAL] または [OUTPUT] として指定します。
10. (オプション) [Set Keep Policy] を選択してオプションを選びます。

    keep ポリシーを編集するには、[Inputs] メニューで入力ウィンドウまたは入力ストリームを右クリックします。keep ポリシーを追加するには [Set Keep Policy] を選択します。keep ポリシーを削除するには [Delete Keep Policy] を選択します。

参照：

- *単純なクエリの作成と変更：Filter* (38 ページ)
- *単純なクエリの作成と変更：Aggregate* (39 ページ)
- *単純なクエリの作成と変更：Compute* (40 ページ)
- *単純なクエリの作成と変更：Union* (45 ページ)
- *単純なクエリの作成と変更：Pattern* (46 ページ)
- *単純なクエリ* (36 ページ)
- *keep ポリシー* (34 ページ)
- *ジョインのサポート* (42 ページ)

ジョインのサポート

単純なジョイン・クエリに入れる属性の組み合わせを指定します。

ESP スタジオで作成する単純なジョイン・クエリのタイプを決めるには、この説明を参考にして、ジョインのコンポーネントをアタッチする方法と、[Edit Join Expression] ダイアログ・ボックスで変更する設定を判断します。

注意： CCL エディタでカンマ区切り構文を使用してジョインを作成してから、ビジュアル・エディタの [Edit Join Expression] ダイアログを使用して ON 句を追加した場合、カンマ区切り構文で最初に作成された WHERE 句は削除されません。これは結果には影響しませんが、パフォーマンスに悪影響を与えます。

ストリーム、ウィンドウ、またはデルタ・ストリームは、ジョインを構成できます。ただし、デルタ・ストリームをジョインに構成できるのは、 keep ポリシーが定義されている場合のみです。1 つのジョインには、任意の数のウィンドウとデ

ルタ・ストリームを (対応する keep ポリシーを使用して) 構成できますが、ストリームは 1 つしか構成できません。セルフ・ジョインもサポートされます。たとえば、各インスタンスにエイリアスを指定することによって、同じウィンドウまたはデルタ・ストリームを 1 つのジョインに複数回構成できます。

ストリームとウィンドウのジョインでは、ターゲットとして、集約が指定されたストリームまたはウィンドウを指定できます。ストリームとウィンドウのジョインはキーを指定せず、ウィンドウはキーを必要とするので、ウィンドウをターゲットとして使用するには集約が必要です。集約時に、**GROUP BY** カラムが、ターゲット・ウィンドウのキーを自動的に生成します。この制限は、デルタ・ストリームとウィンドウのジョインには適用されません。これは、 keep ポリシー を使用すると、デルタ・ストリームが名前なしウィンドウに変換されるためです。

Event Stream Processor では、次のすべてのジョイン・タイプがサポートされます。

ジョイン・タイプ	説明
内部ジョイン	ジョインがレコードを生成するには、ジョインの両側から 1 つずつのレコードが必要である。
左外部ジョイン	レコードが右側 (内側) にあるかどうかにかかわらず、ジョインの左側 (外側) からレコードが生成される。右側にレコードが存在しない場合、内側からのカラムは NULL 値になります。
右外部ジョイン	左外部ジョインとは反対に、右側がジョインの外側で、左側がジョインの内部である。
全外部ジョイン	ジョインの右側または左側に一致があるかどうかにかかわらず、レコードが生成される。

Event Stream Processor では、以下のカーディナリティもサポートされます。

タイプ	説明
1 対 1	ジョインの一方の側のキーは、ジョインの他方の側のキーに完全にマッピングされる。1 つの受信ローから出力として生成されるのは、1 つのローのみです。
1 対多	「1」の側からの 1 つのレコードは、「多」の側の複数のレコードとジョインする。ジョインの「1」の側とは、すべてのプライマリ・キーがジョインの他方の側にマッピングされる側です。レコードがジョインの「1」の側に来ると常に、多数のローが出力として生成されます。

タイプ	説明
多対多	ジョインの両側のキーは、ジョインの他方の側のキーに完全にはマッピングされない。ジョインのいずれかの側にローが来ると、複数のローが出力として生成される可能性があります。

キー・フィールド・ルール
キー・フィールド・ルールは、ローの挿入が重複しているかキー・フィールドが NULL の場合に、ローが拒否されないようにするために必要です。

- ターゲットのキー・フィールドは常に、ジョインの「多」の側のキーから完全に抽出される。多対多の関係では、キーはジョインの両側のキーから抽出されます。
- 1 対 1 の関係では、キーはジョインのいずれかの側のキーから完全に抽出される。
- 外部ジョインでは、キー・フィールドはジョインの外側から抽出される。ジョインの外側が関係の「多」の側ではない場合、エラーが生成されます。
- 全外部ジョインでは、すべてのソースとターゲットでキー・カラムの数と型が同一であることが必要である。また、キー・カラムには、対応するキー・カラムをソースに含む **FIRSTNONNULL** 式が必要です。

ジョインの結果がウィンドウである場合、特定のルールによって、ターゲット・ウィンドウのプライマリ・キーを形成するカラムが決定されます。複数テーブルのジョインでは、概念上は各ジョインがペアで生成されてから、ジョインの結果が別のストリームまたはウィンドウとジョインされ、以下同様の処理が行われるため、同じルールが適用されます。

次の表では、この情報を、ジョイン・タイプに関連して説明しています。

	1 対 1	1 対多	多対 1	多対多
INNER	少なくとも一方の側からのキーが射影リスト (キーが複合キーである場合は、射影リストの組み合わせ) に含まれる必要あり。	右側からのキーが射影リストに含まれる必要あり。	左側からのキーが射影リストに含まれる必要あり。	両側からのキーが射影リストに含まれる必要あり。
LEFT	左側からのキーのみが含まれる必要あり。	不可。	左側からのキーが射影リストに含まれる必要あり。	不可。

	1 対 1	1 対多	多対 1	多対多
RIGHT	右側からのキーのみが含まれる必要あり。	右側からのキーが射影リストに含まれる必要あり。	不可。	不可。
OUTER	キーは、両側からのキーの各ペアに対して **FIRSTNONNULL**() を使用して生成される必要あり。	不可。	不可。	不可。

これらのオプションは、[Edit Join Expression] ダイアログ・ボックスの [Options] ペインで定義できます。

ネスト・ジョイン
Event Stream Processor では、ネスト・ジョインを実装する場合にいくつかの重要な機能に注意する必要があります。ネスト・ジョインの構文は CCL ではサポートされていますが、ビジュアル・エディタではネスト・ジョインの作成も編集もできません。ネスト・ジョインが CCL ファイルに定義されている場合に、ビジュアル・エディタに切り替えると、空のジョイン・コンパートメントが表示されます。

参照：
- *単純なクエリの作成と変更：Join* (41 ページ)

単純なクエリの作成と変更：Union
ユニオン・オブジェクトを使用して、2 つ以上の入力ストリームまたは入力ウィンドウを組み合わせて 1 つの出力にします。すべての入力のスキーマが一致するようにします。

1. ビジュアル・エディタのパレットの [Streams and Windows] で、[Union] () を選択します。
2. 図でロケーションを選択し、クリックすると、図形が追加されます。
3. ユニオン・オブジェクトを 2 つ以上の入力にアタッチします。入力には、ストリーム、ウィンドウ、または Flex 演算子を選択できます。
4. (オプション) [Toggle]  オプションを使用して、ユニオン・オブジェクトを [LOCAL] または [OUTPUT] として指定します。

参照：
- *単純なクエリの作成と変更：Filter* (38 ページ)
- *単純なクエリの作成と変更：Aggregate* (39 ページ)

- *単純なクエリの作成と変更：Compute* (40 ページ)
- *単純なクエリの作成と変更：Join* (41 ページ)
- *単純なクエリの作成と変更：Pattern* (46 ページ)
- *単純なクエリ* (36 ページ)

単純なクエリの作成と変更：Pattern

パターン一致クエリを実行します。このクエリでは、1 つ以上の入力で受信イベントの特定のパターンを監視し、パターンが検出されたら出力イベントを生成します。パターンでは、CCL の **MATCHING** 句を使用します。

1. ビジュアル・エディタのパレットの [Streams and Windows] で、[Pattern] () をクリックします。
2. 図でロケーションを選択し、クリックすると、図形が追加されます。
3. パレットの図形を、クエリへの入力である 1 つ以上のストリームまたはウィンドウに接続します。
4. 次のように、カラムを追加します。
   a) 図形ツールバーの [Copy Columns from Input] () をクリックして、パターン・クエリのスキーマにコピーするカラムを選択します。

      これが、パターンが検出されたときに生成される新しいイベントのスキーマになります。
   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。
   c) カラム式を編集するには、ダブルクリックしてインライン・エディタを開きます。または、式を選択して、[Ctrl] キーを押しながら [F2] キーを押すと、選択した式がポップアップ・エディタに表示されます。
5. 次のように、パターン式を作成して編集します。
   a) [Add Pattern] をクリックします。
   b) イベントのエイリアスを入力します。
   c) 時間間隔またはパラメータのどちらかを入力します。
   d) 式を定義するには、[Pattern] を右クリックしてイベントを追加します。式の要素を続けて右クリックして演算子を追加し、イベント式を調整します。[Next] をクリックします。
   e) [Add] をクリックしてジョイン条件を追加します。

   有効なパターン式の詳細については、『CCL プログラマーズ・ガイド』の「ON 句：パターン一致構文」を参照してください。
6. (オプション) [Toggle] オプションを使用して、パターン・オブジェクトを [LOCAL] または [OUTPUT] として指定します。

参照：
- *単純なクエリの作成と変更：Filter* (38 ページ)
- *単純なクエリの作成と変更：Aggregate* (39 ページ)
- *単純なクエリの作成と変更：Compute* (40 ページ)
- *単純なクエリの作成と変更：Join* (41 ページ)
- *単純なクエリの作成と変更：Union* (45 ページ)
- *単純なクエリ* (36 ページ)

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。

参照：
- *単純なクエリ* (36 ページ)
- *図形のリファレンス* (24 ページ)
- *単純なクエリの追加* (36 ページ)

キー・カラムの設定

ビジュアル・エディタのデルタ・ストリーム、ウィンドウ、Flex 演算子の各図形の [Column] コンパートメントで、プライマリ・キーを設定します。

複数のカラムをプライマリ・キーとして指定できます。ビジュアル・エディタで、プライマリ・キーは のアイコンで表示されます。抽出されたプライマリ・キー

は のアイコンで表示されます。ターゲット要素に **PRIMARY KEY DEDUCED** フラグが設定されていると、抽出されたキーが計算されます。

注意： **PRIMARY KEY DEDUCED** は、デルタ・ストリームとウィンドウでのみサポートされます。これらの要素の抽出されたキーのプロパティは [Properties] ビューで変更できます。

1. 希望するクエリ・オブジェクト (デルタ・ストリーム、ウィンドウ、または Flex の図形) の [Columns] コンパートメントを展開します。
2. カラム名の左にあるアイコンをクリックすると、そのカラムがプライマリ・キーになります。
   単一キーのアイコン によって、そのカラムはプライマリ・キーとして指定されます。
3. 抽出されたプライマリ・キーを使用してクエリ・オブジェクトのプライマリ・キーを設定するには、ターゲット・ストリームまたはターゲット・ウィンドウ内の任意のカラムまたは抽出されたキーをクリックします。
   これで、最初に選択したカラムと、その他のすべての抽出されたキーのカラムがプライマリ・キーになります。さらに、ターゲット・ストリームまたはターゲット・ウィンドウは **PRIMARY KEY DEDUCED** ではなくなります。
4. カラムからプライマリ・キーの指定を削除するには、カラム名の左にある をクリックします。
   単一キーのアイコンに代わってカラムのアイコンが表示され、カラムがプライマリ・キーの一部ではなくなったことが示されます。

ウィンドウ、ストリーム、デルタ・ストリームのカラム式の編集

インライン・エディタまたはダイアログベースの式エディタを使用して、ウィンドウ、ストリーム、デルタ・ストリームのカラム式を変更します。

1. (オプション) カラム式を追加するには、図形ツールバーの [Add Column Expressions] をクリックします。
2. [Column Expressions] コンパートメントを展開します。
3. 次の手順のいずれかに従って、カラム式を変更します。
   - [F2] キーを押して、インライン・エディタを開きます。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] キーを押して、式エディタを開きます。[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、式を手動で入力します。
   - [Properties] ビューで式を修正します。

参照：

- *カラム式* (49 ページ)

カラム式

カラム式では、入力カラムの値、カラム値の相互の関係、または計算式に基づいて、結果が算出されます。組み込みまたはユーザ定義の関数、定数、パラメータ、または変数を使用できます。

単純式

CCL の単純式は、定数、NULL、またはカラムを指定します。定数は、数またはテキスト文字列です。リテラル NULL は null 値を意味します。NULL は他の式の一部にはなりませんが、NULL 自体は式です。

カラム名を単独で、またはそのストリームまたはウィンドウの名前と一緒に指定できます。カラムとストリームまたはウィンドウの両方を指定するには、"stream_name.column_name" という形式を使用します。

単純式の例を次に示します。

- `stocks.volume`
- `'this is a string'`
- `26`

複合式

CCL の複合式は、単純式または複合式の組み合わせです。複合式には、演算子、関数、CCL の単純式 (定数、カラム、NULL) を記述できます。

カッコを使用して、式のコンポーネントの優先順位を変更できます。

複合式の例を次に示します。

- `sqrt (9) + 1`
- `('example' + 'test' + 'string')`
- `( length ('example') *10 ) + pi()`

式でのカラム・エイリアス

各式では、カラムのユニークな名前またはエイリアスが定義されます。

PortfolioValuation の例で、VWAP という派生ウィンドウは、カラム Symbol、Price、TradeTime を持つ入力ストリーム (PriceFeed) から入力を取得し、集約式が含まれます。この派生ウィンドウ (ビジュアル・エディタで単純な集約クエリとして作成) のカラム・エイリアスは、次のとおりです。

エイリアス	カラム式
Symbol	PriceFeed.Symbol
LastPrice	PriceFeed.Price

エイリアス	カラム式
VWAP	(sum ((PriceFeed.Price * CAST (FLOAT , PriceFeed.Shares))) / CAST (float , sum (PriceFeed.Shares)))
LastTime	PriceFeed.TradeTime

式でのデータ型
カラム式のデータ型は、明示的に作成されたインライン・スキーマ、または入力アダプタから検出されたスキーマのどちらかから継承されます。カラム式エディタではなく、スキーマ・エディタでサポートされているデータ型から選択します。

式では、文字列データを `'my_string_data'` のように単一引用符で囲みます。

大文字と小文字の区別

- すべての識別子は大文字と小文字を区別します。これには、ストリーム、ウィンドウ、パラメータ、変数、スキーマ、カラムの各名前が含まれます。
- キーワードは大文字と小文字を区別しないので、識別名として使用できません。
- 組み込み関数の名前 (キーワードを除く) とユーザ定義関数は、大文字と小文字が区別されますが、組み込み関数の中には、`setOpcode` と `setopcode` のように、小文字の形式と大文字と小文字の混合形式の両方の名前を持つものもあります。

参照：
- *関数* (71 ページ)
- *演算子* (71 ページ)
- *ウィンドウ、ストリーム、デルタ・ストリームのカラム式の編集* (48 ページ)

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。

参照：
- *図形の図への追加* (28 ページ)
- *ビジュアル・エディタでのキーボード・ショートカット* (65 ページ)

プロジェクトへの高度な機能の追加

複雑な演算と式、再利用可能なモジュールと名前付きスキーマ、カスタム・アダプタを追加して、プロジェクトを完成させます。

これらの高度な機能はすべてオプションです。

複雑なクエリ

汎用の派生ストリーム、派生ウィンドウ、派生デルタ・ストリームの図形を使用して、単純なクエリの図形を使用して作成できる継続クエリよりも複雑な継続クエリをビジュアル・エディタで作成します。

派生ストリーム、派生ウィンドウ、または派生デルタ・ストリームは、入力をアダプタから直接取得するのではなく、別のストリームまたはウィンドウから取得し、それに継続クエリを適用します。ビジュアル・エディタでは、単純なクエリはすべて派生ストリームまたは派生ウィンドウのタイプです。

たとえば、ジョイン条件のセットとパターン一致式の両方を適用する継続クエリを作成するには、汎用の派生ウィンドウを使用します。

データの入力、出力、保持の各要件に従って、挿入、更新、削除の各操作を保存するために、図形のタイプを選択します。

表 5 : 派生ストリーム、派生ウィンドウ、派生デルタ・ストリームの各ルール

図形	入力	出力	状態の保持	挿入、更新、削除の保存
派生ストリーム	別のストリームまたは入力ウィンドウ	ストリーム	なし	なし
派生ウィンドウ	別のストリームまたはウィンドウ	ウィンドウ	keep ポリシーで定義 (デフォルトは keep all rows)	あり

図形	入力	出力	状態の保持	挿入、更新、削除の保存
派生デルタ・ストリーム	別のウィンドウ	ストリーム	なし	あり **注意：** デルタ・ストリームは挿入または削除のどちらかのみを受け入れます。

参照：
- *ジョインのサポート* (42 ページ)
- *挿入、更新、削除* (6 ページ)
- *ウィンドウ、ストリーム、デルタ・ストリームのカラム式の編集* (48 ページ)

モジュール性

Sybase Event Stream Processor のモジュールは再利用できます。モジュールはプロジェクト全体で複数回ロードしたり使用したりできます。

モジュール性とは、プロジェクトの要素をモジュールと呼ばれる自己完結型の再利用可能なコンポーネントに編成することです。これによって、入力と出力が適切に定義され、よく繰り返されるデータ処理プロシージャをカプセル化できます。

モジュールは、インポート・ファイルやメイン・プロジェクトなどの他のオブジェクトと共に独自のスコープを持ちます。スコープは、変数または定義へのアクセスが許可される範囲を定義します。スコープで宣言された変数、オブジェクト、または定義は、スコープ内でのみアクセス可能です。親スコープと呼ばれる外側のスコープや、その他の外部スコープからはアクセスできません。親スコープは、モジュールでも、メイン・プロジェクトでも可能です。たとえば、モジュール A がモジュール B をロードし、メイン・プロジェクトがモジュール A をロードする場合、モジュール A のスコープはモジュール B の親スコープです。モジュール A の親スコープはメイン・プロジェクトです。

モジュールには、明示的に宣言された入力と出力があります。モジュールへの入力は、親スコープのストリームまたはウィンドウと関連付けられ、モジュールの出力は、識別子を使用して親スコープに公開されます。モジュールを再利用すると、そのモジュール内のストリーム、変数、パラメータ、またはその他のオブジェクトがレプリケートされるので、モジュールの各バージョンは他のバージョンとは別個に存在します。

モジュールは他のモジュール内にロードできるので、モジュール A はモジュール B をロードでき、モジュール B はモジュール C をロードできる、というようになります。ただし、モジュール依存性ループは無効になります。たとえば、モジュール A がモジュール B をロードし、モジュール B がモジュール A をロードす

ると、CCL コンパイラは、モジュール A と B 間の依存性ループを示すエラーを生成します。

CREATE MODULE 文はモジュールを作成します。モジュールは、プロジェクトに複数回ロードでき、モジュールの入力と出力はより大きいプロジェクトのさまざまなパーツにバインドできます。**LOAD MODULE** 文を使用すると、定義済みのモジュールをプロジェクト全体で 1 回以上再利用できます。

注意： モジュール関連のコンパイル・エラーはすべて致命的です。

モジュールの作成

ビジュアル・エディタで新しいモジュールを既存のプロジェクトに追加します。

複数のプロジェクト間で幅広く再利用する予定がない場合は、プロジェクト内にモジュールを直接作成します。

1. ビジュアル・エディタのパレットの [Shared Components] で、[Module] () を選択します。
2. 図でロケーションを選択し、クリックすると、図形が追加されます。
3. ビジュアル・エディタのどこかをクリックしてモジュールを置きます。

次のステップ

モジュールを開き、そのプロパティを編集してモジュールを完成させ、入力と出力を設定します。必要に応じて、プロジェクトにモジュールをロードします。

参照：

- *モジュールの編集* (53 ページ)
- *モジュール・ファイルの作成* (54 ページ)
- *別の CCL ファイルからの定義のインポート* (55 ページ)
- *プロジェクト内でのモジュールの使用* (56 ページ)
- *ロード・モジュールの設定* (56 ページ)

モジュールの編集

基本的なモジュール・プロパティの編集と、モジュールの入力、出力、インポートの各機能について説明します。

前提条件

モジュールを作成します。

手順

具体的なモジュール入力とモジュール出力は、プロジェクトの開発者が決定します。インポートしたモジュールでは編集が制限されますが、モジュール入力とモジュール出力の各ノードは変更できます。

1. ビジュアル・エディタで、編集するモジュールを選択します。
2. 次のいずれかを行って、モジュール名をこのモジュールのスコープにあるすべてのオブジェクト名の間でユニークになるように編集します。
   - モジュール名をクリックします。
   - 冗長モードで、[Edit] をクリックします。
   - モジュールを選択し、[Properties] ビューで [name] の値を変更します。

   デフォルトでは、[Properties] ビューは [Authoring] パースペクティブの左下にあります。
3. [Add Module Inputs] () をクリックします。
4. [Module Inputs] ダイアログで、追加または削除する入力を選択して、[OK] をクリックします。
5. [Add Module Outputs] () を選択します。
6. [Module Outputs] ダイアログで、追加または削除する出力を選択して、[OK] をクリックします。
7. **CREATE MODULE** 文にアクセスしてその内容を編集するには、[Open Module Diagram] () を選択します。
8. 表示され図でモジュールを編集します。
9. [Properties] ビューでコメントを追加します。

参照：

- *モジュールの作成* (53 ページ)
- *モジュール・ファイルの作成* (54 ページ)
- *別の CCL ファイルからの定義のインポート* (55 ページ)
- *プロジェクト内でのモジュールの使用* (56 ページ)
- *ロード・モジュールの設定* (56 ページ)

モジュール・ファイルの作成

プロジェクトにインポートできる新しい別個のモジュール・ファイルを作成します。

プロジェクト内にモジュールを作成するか、別個のファイルにモジュールを作成してからプロジェクトにインポートすることができます。特定のモジュールを別のプロジェクトで再利用する可能性がある場合は、別個のモジュール・ファイル

を作成します。モジュール・ファイルとは、**CREATE MODULE** 文を別個に保持する CCL ファイルです。

1. [File] > [New] > [CCL Module File] を選択します。
2. ファイル名を入力します。
   これがモジュール名になります。名前は、このモジュールのスコープにあるすべてのオブジェクト名の間でユニークになるようにします。
3. (オプション) 別のフォルダを指定します。
   デフォルトでは、モジュールは現在のプロジェクトのワークスペースに作成されます。
4. 必要に応じてモジュールを変更し、保存します。
   CCL を編集するには、『CCL プログラマーズ・ガイド』の「CREATE MODULE 文」を参照してください。

参照：
- *モジュールの作成* (53 ページ)
- *モジュールの編集* (53 ページ)
- *別の CCL ファイルからの定義のインポート* (55 ページ)
- *プロジェクト内でのモジュールの使用* (56 ページ)
- *ロード・モジュールの設定* (56 ページ)

別の CCL ファイルからの定義のインポート

モジュール・ファイルをインポートしてプロジェクトでモジュールを使用します。

この処理を行うには、CCL エディタで **IMPORT** 文を使用するか、次に説明するように、ビジュアル・エディタの [Outline] ビューを使用します。

1. [Authoring] タブを選択します。
2. [Switch to Visual] をクリックするか、[F4] キーを押して、ビジュアル・エディタを開きます。
3. [Outline] ビューが表示されない場合は、[Window] > [Show View] > [Outline] を選択するか、[Alt] キーと [Shift] キーを押しながら [O] キーを押します。
4. [Outline] ビューで [Statements] リストを展開します。
5. [Imports] 文を右クリックして [Create Child] > [Import] を選択します。
6. インポートするモジュールを 1 つ以上選択して、[OK] をクリックします。
7. インポートしたモジュールが表示されるまで、インポート・ファイルを展開します。
8. モジュールをクリックして図のどこかにドラッグします。

参照：
- *モジュールの作成* (53 ページ)
- *モジュールの編集* (53 ページ)
- *モジュール・ファイルの作成* (54 ページ)
- *プロジェクト内でのモジュールの使用* (56 ページ)
- *ロード・モジュールの設定* (56 ページ)

プロジェクト内でのモジュールの使用
プロジェクト内に定義済みモジュールのインスタンスを作成して、モジュールの入力と出力をプロジェクトのストリームまたはウィンドウにバインドできます。

既存のモジュール (プロジェクト内で作成されたもの、またはインポートされたもの) はプロジェクトのどこででも使用できます。モジュールをプロジェクト内で使用する (プロジェクトにロードする) には、バインドを設定してモジュールの入力と出力をプロジェクトのストリームまたはウィンドウにアタッチし、モジュールで使用するパラメータを設定します。

1. ビジュアル・エディタのパレットの [Module] ドロワで、プロジェクトに追加するモジュールを見つけて選択します。

   パレットに、メインの CCL ファイルまたはインポートされた CCL ファイルのどちらかで、現在のプロジェクトに定義されているすべてのモジュールがリストされます。**CREATE MODULE** 文がない場合、パレットのドロワは空です。
2. 図のどこかをクリックしてロード・モジュールを置きます。

参照：
- *モジュールの作成* (53 ページ)
- *モジュールの編集* (53 ページ)
- *モジュール・ファイルの作成* (54 ページ)
- *別の CCL ファイルからの定義のインポート* (55 ページ)
- *ロード・モジュールの設定* (56 ページ)

ロード・モジュールの設定
入力バインドと出力バインドを追加または削除して、ロード・モジュールのプロパティを変更します。

既存のモジュール定義を使用して新しいモジュール・インスタンスを作成すると、アクティブなモジュールが作成されます。

1. 図で、編集するロード・モジュールを選択します。
2. ロード・モジュールの名前を編集するには、次のどちらかを行います。

- ロード・モジュール名をクリックします。
- 冗長モードで、[Edit] をクリックします。

3. [Add Input Binding] (一番左の) を選択して、入力バインドを追加します。
4. [Add Output Binding] ( ) を選択して、出力バインドを追加します。
5. ロード・モジュール内のバインドを個別に選択し、[Properties] ウィンドウで次のオプションのいずれかを変更して、入力バインドまたは出力バインドをさらに変更します。

プロパティ	値
inputStreamOrWindow	リストから使用可能な入力ストリームまたは入力ウィンドウのコンポーネントを選択します。
streamOrWindowInModule	既存のストリーム入力またはウィンドウ入力とバインドする使用可能なストリームまたはウィンドウを選択します。
comment (出力のみ)	出力ストリームのコメントまたは説明を追加します。
name (出力のみ)	出力ストリームに名前を追加します。

6. [Add Parameter Binding] ( ) をクリックします。
7. [Properties] ウィンドウで次のパラメータ・バインド値を編集します。
   - [parameterInModule]：文字列の分類。デフォルトでは NULL です。
   - [parameterValue]：パラメータ・バインドの定義に必要な値句。
8. [Add Store Binding] ( ) をクリックします。
9. [Properties] ウィンドウで使用可能なフィールドを選択して変更し、ストア・バインドを編集します。
   - **[storeInModule] –** 文字列の分類。デフォルトでは NULL です。
   - **[storeValue] –** パラメータ・バインドを定義する値句。
10. ロード・モジュールの内側で使用されている入力ウィンドウまたは出力ウィンドウにアクセスするには、[Open Module Diagram] ( ) を選択します。

参照：
- *モジュールの作成* (53 ページ)
- *モジュールの編集* (53 ページ)
- *モジュール・ファイルの作成* (54 ページ)
- *別の CCL ファイルからの定義のインポート* (55 ページ)
- *プロジェクト内でのモジュールの使用* (56 ページ)

ストア

ストアのデフォルト値を設定するか、ログ・ストアまたはメモリ・ストアを選択してウィンドウからのデータをどのように保存するかを指定します。

各ウィンドウはストアに割り当てられ、そのストアで保存レコードが保持されます。デフォルトでは、ウィンドウはすべてメモリ・ストアに割り当てられます。追加ストアを作成することで、データをリカバリしたりパフォーマンスを最適化したりできるようになります。ウィンドウは特定のストアに割り当てることができます。

また、**CREATE DEFAULT STORE** 文を使用してデフォルトのストアを明示的に作成することもできます。デフォルトのストア設定を定義して、新しいウィンドウを特定のストア・タイプに割り当てない場合のストアのタイプと場所を指定できます。

ログ・ストア
永続性を実装するにはログ・ストアを使用して、すべてのデータをディスクに記録します。これによって、障害が発生した場合にデータを確実にリカバリできます。

ログ・ストアの作成には **CREATE LOG STORE** 文を使用します。ログ・ストアをデフォルトのストアとして設定するには、**CREATE DEFAULT STORE** 文を使用して、デフォルトのメモリ・ストアを無効にします。

メモリ・ストア
メモリ・ストアでは永続性は使用されず、データはすべてメモリに保存されます。メモリ・ストアには、プロジェクトが実行されているかぎり、前回のサーバの起動時からのプロジェクトのクエリの状態が保持されます。 クエリの状態はディスクではなくメモリに保持されるため、メモリ・ストアへのアクセスはログ・ストアより高速です。

メモリ・ストアの作成には **CREATE MEMORY STORE** 文を使用します。デフォルトのストアが定義されていない場合、新しいウィンドウは自動的にメモリ・ストアに割り当てられます。上記の関連文のどちらでも、特定のメモリ・ストアの動作を指定したり、デフォルトのストアを設定したりできます。

ログ・ストアの作成

サーバのシャットダウンや障害が発生した場合にウィンドウにデータを復元できるようにログ・ストアを作成します。

前提条件
最適なパフォーマンスを確保するため、ログ・ストアのサイズ、数、場所については、システム管理者に相談してください。

手順

1. ビジュアル・エディタのパレットの [Shared Components] で、[Log Store] をクリックします。
2. 図でロケーションを選択し、クリックすると、図形が追加されます。
3. ログ・ストアをウィンドウに接続します。
4. [Set Store Properties] をクリックして、プロパティ値を変更します。

注意： 次の表には、最初に [Properties] ダイアログに表示されるプロパティ名を示し、その後にストア図形の [Properties] コンパートメントに表示されるプロパティ名を示します。

表 6 : ログ・ストアのプロパティ

プロパティ	説明
Filename (FILENAME)	ログ・ストア・ファイルを書き込むフォルダの絶対パスまたは相対パス。相対パスをおすすめします。
Max Size (GB) (MAXFILESIZE)	ログ・ストア・ファイルの最大サイズ (MB 単位)。デフォルトは 8MB です。
Sweep Amount (%) (SWEEPAMOUNT)	単一のパスでクリーンアップできる、メガバイト単位のデータ量。デフォルトは **maxfilesize** の 20% です。
Reserve Percentage (%) (RESERVEPCT)	空き領域として保持するログの割合。デフォルトは 20% です。
Ck Count (CKCOUNT)	作成されるレコードの最大数。この数を超えると、中間メタデータが作成されます。デフォルトは 10,000 です。
Sync (SYNC)	ストリームが更新されるたびにストリームと同期して永続化データを更新するかどうかを指定します。 true の値では、システムで受信確認されるすべてのレコードが永続化されますが、パフォーマンスが犠牲になります。false の値では、パフォーマンスは向上しますが、受信確認されても、まだ永続化されていないデータを失う可能性があります。デフォルトは false です。

5. (オプション) [Default] を選択して、これをプロジェクト (またはモジュール) のデフォルト・ストアに設定します。

参照：

- *メモリ・ストアの作成* (60 ページ)

メモリ・ストアの作成

前回のサーバ起動時から、メモリに継続クエリの状態を保持するメモリ・ストアを作成します。

前提条件

最適なパフォーマンスを確保するため、メモリ・ストアのタイプ、数、インデックスの各値については、システム管理者に相談してください。

手順

1. ビジュアル・エディタのパレットの [Shared Components] で、[Memory Store] をクリックします。
2. 図でロケーションを選択し、クリックすると、図形が追加されます。
3. メモリ・ストアをウィンドウに接続します。
4. ストアに、プロジェクトまたはモジュールのスコープ内でユニークな名前を指定します。
5. (オプション) [Set Store Properties] をクリックして、プロパティ値を変更します。

表 7 : メモリ・ストアのプロパティ

プロパティ	説明
Index Size Hint (KB) (INDEXSIZEHINT)	(オプション) ハッシュ・インデックスを使用する場合に、ハッシュ・テーブル内の要素の初期数を指定します。値は 1024 の倍数で指定します。この値が大きいほど多くのメモリが消費されますが、遅延時間のスパイクが発生する可能性は減ります。デフォルトは 8KB です。
Index Kind (INDEXTYPE)	格納される要素のインデックス・メカニズムのタイプ。デフォルトは [Tree] です。 バイナリ・ツリーには [Tree] を使用します。バイナリ・ツリーは、メモリの使用量の点で予測可能であると同時に、速度の点で一貫性があります。 ハッシュ・テーブルは高速ですが、メモリの消費量が増えることが多いため、ハッシュ・テーブルには [Tree] を使用します。

6. (オプション) [Default] を選択して、これをプロジェクト (またはモジュール) のデフォルト・ストアに設定します。

参照：
- *ログ・ストアの作成* (58 ページ)

Flex 演算子

Flex 演算子はカスタム演算子で、これを使用して、受信イベントに対して動作する SPLASH スクリプトを作成できます。

Flex 演算子では、受信イベントに適用できるタイプのビジネス・ロジックを拡張し、標準の CCL や SQL クエリで実行できる処理以上のことを実行できます。SPLASH で個々のイベント・ハンドラを作成できるようにすることで CCL を拡張します。

Flex 演算子は、ウィンドウやストリームの組み合わせを入力として取得し、アタッチ先の SPLASH スクリプトにあるロジックに従って出力ストリームまたは出力ウィンドウを生成します。

参照：
- *SPLASH* (11 ページ)

ビジュアル・エディタでの Flex 演算子の作成

SPLASH で作成されたイベント・ハンドラをプロジェクトに追加するための Flex 演算子を作成します。

1. ビジュアル・エディタのパレットの [Streams and Windows] で、[Flex] ( ) を選択します。
2. 図のどこかをクリックして Flex 演算子を置きます。
3. 次のどちらかの手順に従って、Flex 演算子の名前を設定します。
   - クリックして [F2] キーを押し、演算子の名前を編集します。または、
   - 冗長モードで、名前の隣の編集アイコン をクリックします。
4. Flex の図形を適切な入力ストリームまたは入力ウィンドウに接続します。

   注意： ストリームまたはウィンドウを Flex 演算子に接続すると、デフォルトでは、ソースが Flex の図形への入力として追加され、ソースのストリームまたはウィンドウから On Input メソッドが作成されます。
5. [Add Columns] をクリックして Flex 演算子で生成されたイベントのスキーマを定義します。
6. [Add On Input Method] ボタンを使用して On Input メソッドを追加します。

   各メソッドは SPLASH スクリプトで、関連付けられた入力でイベントが受信されると呼び出されます。言い換えれば、これらはイベント・ハンドラです。

a) 各メソッドの SPLASH スクリプトを編集するには、Flex の図形が選択されていることを確認し、[F4] キーを押して CCL エディタに切り替えます。カーソルが CREATE FLEX 文に置かれた状態で、CCL エディタが開きます。
b) SPLASH スクリプトを編集します。
c) [F4] キーを押して、ビジュアル・エディタに再び切り替えます。

7. (オプション) エイジング・ポリシーを追加します。
8. (オプション) [Set Output Keep Policy] をクリックし、keep ポリシーのオプションを設定します。

参照：

- *keep ポリシー* (34 ページ)
- *エイジング・ポリシーの設定* (62 ページ)

ビジュアル・エディタでのスキーマの作成

多くのストリームやウィンドウから参照できる共有スキーマ・オブジェクトを作成します。

1. [Shared Components] の下の [Palette] メニューで、[Named Schema] () を選択します。
2. ビジュアル・エディタのどこかをクリックしてスキーマを置きます。
3. 次のどちらかの手順に従って、スキーマの名前を設定します。
   - 名前ラベルをダブルクリックします。または、
   - [Properties] ウィンドウ内から名前フィールドを編集します。
4. [Add Columns] () をクリックして個々のカラムを追加します。
5. カラムの名前とデータ型を編集します。

エイジング・ポリシーの設定

エイジング・ポリシーを定義して、一定の期間、レコードが更新されなかった場合のレコードの保持時間を制限します。

エイジング・ポリシーは、ウィンドウやその他のステートフルな要素向けの高度なオプション機能です。

1. [Set Aging Policy] を選択して次の値を設定します。

値	説明
Aging Time	この期間を過ぎると、データ・エイジング・プロセスが開始します。 この期間は、時間、分、秒、ミリ秒、マイクロ秒 単位で指定できます。

値	説明
Aging Field	エイジング期間が経過してレコードに対し何もアクティビティが発生しなかったたびに、または定義されている最大値に達するまで、1 つずつ増分するレコードのフィールド。 デフォルトでは、この値は 1 です。
(オプション) Max Aging Field Value	エイジング・フィールドの最大値で、この値まで増分できます。指定しなかった場合、エイジング・フィールドは 1 回増分されます。
(オプション) Aging Time Field	エイジング・プロセスの起動時間。指定しなかった場合は、内部のロー時間が使用されます。 指定する場合は、フィールドに有効な起動時間を指定します。

2. (オプション) ポリシーをダブルクリックしてパラメータを編集します。

プロジェクトが実行されると、[Aging Time] または [Max Aging Field Value] に達するまでレコードが累積されます。レコードが更新されると、経過時間は 0 にリセットされます。

エラー・ストリームのモニタリング

エラー・ストリームを使用してプロジェクト内の他のストリームでのエラーを追跡するようにプロジェクトを変更します。

エラー・ストリームは、他のストリームからエラーに関する情報を収集します。開発中のプロジェクトのデバッグや運用環境でのプロジェクトのモニタに使用できます。

1. モニタするプロジェクトと特定のストリームを識別します。
2. 複数のエラー・ストリームを使用するかどうかを決定します。各エラー・ストリームに対して、アクセス性を決定します。
3. そのプロジェクト内でエラー・ストリームを作成します。
4. エラー・ストリームからの情報を表示します。

エラー・ストリームの作成

プロジェクト内の他のストリームから、エラーとそれらのエラーの原因であるレコードを収集する、特殊なタイプのストリームを追加します。

開発時にプロジェクトをデバッグするか、運用モードでプロジェクトをモニタするかにかかわらず、[Error Stream] では、他のストリームでのエラーとそれらのエラーの原因であるレコードをリアルタイムで確認できます。

注意： エラー・ストリームで他のエラー・ストリームをモニタすることはできません。

1. ビジュアル・エディタにプロジェクトを開きます。
2. パレットで [Error Stream] の図形をクリックして、図中の何もない領域をクリックします。
3. プラス (+) 記号をクリックします。
   モニタできるプロジェクト内のストリームのリストが表示されます。
4. モニタするストリームを指定します。[Select All] をクリックするか、モニタする各ストリームのチェックボックスをオンにして、[OK] をクリックします。
   指定したすべてのストリームが、エラー情報を送信していることを示す赤い線で [Error Stream] に接続されます。

エラー・ストリーム・データの表示

デフォルトでは、エラー・ストリームはローカルですが、その情報はプロジェクトの外部で入手できます。

運用段階では、プロジェクトのモニタは外部で実行される可能性があります。

1. ビジュアル・エディタにプロジェクトを開きます。
2. プロジェクトで発生したエラーをリアルタイムでモニタするには、[Error Stream] で [Type] アイコンをクリックして [LOCAL] から [OUTPUT] に切り替えます。
3. アドホック SQL クエリを有効にするには、プロジェクトで、エラー・ストリームからの下流となるストリームにウィンドウ (たとえば、ErrorState) を追加します。
   ErrorState ウィンドウにエラー・ストリームの状態が保存されるので、**esp_query** ユーティリティを使用してその状態をクエリできます。

エラー・ストリームの変更

エラーとそれらのエラーの原因であるレコードをエラー・ストリームが収集するプロジェクト内のストリームを変更します。

開発段階でプロジェクトをデバッグしたり、運用モードでプロジェクトをモニタしたりする場合に、[Error Stream] を使用してモニタする特定のストリームを変更できます。

注意： エラー・ストリームで他のエラー・ストリームをモニタすることはできません。

1. ビジュアル・エディタにプロジェクトを開きます。
2. ワークスペースで [Error Stream] の図形を見つけて、入力ストリームのリストを確認します。
3. エラー・ストリームに入力ストリームを追加します。プラス (+) 記号をクリックし、ポップアップ・リストでモニタする各ストリームのチェックボックスを

オンにして、[OK] をクリックします。または、パレットの [Connector] を使用して、エラー・ストリームに入力ストリームを接続します。
赤い線で各ストリームが [Error Stream] に接続され、新しいストリーム名が [Inputs] リストに表示されます。

4. エラー・ストリームから入力ストリームを削除します。赤い円で囲まれた X をクリックし、ポップアップ・リストで削除する各ストリームのチェックボックスをオンにして、[OK] をクリックします。
ストリームを [Error Stream] に接続する赤い線と、[Inputs] リスト上のストリーム名が削除されます。

CCL エディタとビジュアル・エディタ間の切り替え

プロジェクトの作成と編集でスタジオの柔軟性を最大限に活用するには、2 つのエディタ間で切り替えます。

- CCL エディタからビジュアル・エディタに切り替えるには、右クリックして [Switch to Visual] を選択する ([F4] キーを押す) か、メイン・ツールバーの  をクリックします。
- ビジュアル・エディタから CCL エディタに切り替えるには、右クリックして [Switch to Text] を選択する ([F4] キーを押す) か、メイン・ツールバーの  をクリックします。

参照：

- *プロジェクトの作成* (16 ページ)
- *プロジェクトを開く* (18 ページ)
- *既存のプロジェクトのインポート* (19 ページ)
- *ビジュアル・エディタでのプロジェクトの編集* (27 ページ)

ビジュアル・エディタでのキーボード・ショートカット

キーボード・ショートカットを使用すれば、ビジュアル・エディタ内のさまざまな機能に簡単にアクセスできます。

次の表に、よく使用されるキーボード・ショートカットをリストします。リスト全体を表示するには、[Help] > [Key Assist] を選択します ([Ctrl] キーと [Shift] キーを押しながら [L] キーを押します)。

キー	アクション
F2	選択された図形の名前または図形内の要素 (コンテキスト依存) を編集する
F4	CCL エディタとビジュアル・エディタ間を切り替える
F7	コンパイルする
F11	[Authoring] パースペクティブと [Run-Test] パースペクティブ間を切り替える
Insert	コンパートメントに新規項目を挿入する
Delete	選択された要素をプロジェクトから削除する
Ctrl +Delete	選択された要素を図から削除する
Ctrl + A	すべて選択する
Ctrl + N	新規プロジェクトを開く
Ctrl + Y	やり直す
Ctrl + Z	元に戻す
Ctrl + F2	カラム式エディタを開く
Ctrl + Space	カラム式で使用可能なカラムと組み込み関数を表示する
Ctrl + マウス・ホイール	ズームインまたはズームアウトする
Ctrl + Shift + L	キーボード・ショートカットの割り当てをすべてリストする
Alt + U	[Outline] でコンパートメント項目を上に移動する
Alt + D	[Outline] でコンパートメント項目を下に移動する
Alt + T	アイコン・モードと冗長モード間で図形を切り替える

参照：
- *図形の図への追加* (28 ページ)
- *要素の削除* (50 ページ)

第 4 章　CCL エディタのオーサリング

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。

たとえば、次のものを追加できます。

- ビジュアル・エディタの機能を上回る複雑なクエリ
- プロジェクトの変数、パラメータ、データ型、関数を宣言するための DECLARE ブロック
- Flex 演算子で呼び出す SPLASH イベント・ハンドラ
- ユーザ定義関数
- 1 つのプロジェクト内または複数のプロジェクト間で複数回使用できる再利用可能なモジュールとスキーマ

CCL 言語の詳細については、『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。

参照：

- *File Explorer* (15 ページ)
- *CCL エディタとビジュアル・エディタ間の切り替え* (65 ページ)
- *プロジェクトのコンパイル* (83 ページ)

CCL エディタの機能

スタジオの CCL エディタには、CCL コードの編集プロセスを簡素化する機能がいくつか用意されています。

表 8 : CCL エディタの機能

機能	説明
Completion Proposals	ワークスペースで完了プロポーザルをアクティブにします [Ctrl + Space]。
Case-Insensitive Syntax Highlighting	CCL コードの編集時に自動的に実行されます。
Error Validation/Syntax Checking	[Problems] ビューにアクセスして CCL コードのエラーを表示します。
Compile and Report Compilation Errors	[Problems] ビューにアクセスして CCL コードのエラーを表示します。

CCL エディタでのキーボード・ショートカット

キーボード・ショートカットを使用すれば、CCL エディタ内のさまざまな機能に簡単にアクセスできます。

キー	アクション
F3	宣言にジャンプする
F4	ビジュアル・エディタと CCL エディタ間を切り替える
F6	CCL 文の順序を変更する
F7	コンパイルする
F11	[Authoring] パースペクティブと [Run-Test] パースペクティブ間を切り替える
Ctrl + N	新規プロジェクト・ファイルを開く
Ctrl + Y	やり直す
Ctrl + Z	元に戻す
Ctrl + Shift + L	キーボード・ショートカットの割り当てをすべてリストする

テキストの検索

CCL コードでテキストを検索します。

1. [Search] > [File] を選択します。
   検索結果が表示されない場合は、[Search] ビューのリンクから新しい検索を開始することもできます。
2. ダイアログに検索基準を入力します。
3. 次のどちらかを選択します。
   - [Search]。結果が表示されます。または、
   - [Replace]。結果が置換されます。
4. [Search] ビューで結果を確認し、[Search] ツールバーのオプションから選択します。

   ヒント： 一致をダブルクリックすると、CCL エディタでその一致が強調表示されます。

CCL でのクエリ

CCL クエリを派生ストリームまたは派生ウィンドウにアタッチして 1 つ以上の入力からデータを選択し、そのデータを希望する出力に変換します。

標準 SQL で **CREATE VIEW** 文を使用する場合と同じように、CCL は **CREATE STREAM**、**CREATE WINDOW**、**CREATE DELTA STREAM** の各文にクエリを埋め込みます。SQL と異なり、CCL で、SELECT は文ではなく、**CREATE** *object_type* 文内で使用する句です。

ビジュアル・エディタでは単純なクエリと呼ばれるビジュアルなコンポーネントを使用してデータを選択できますが、実際にはこれらのクエリは CCL 文で、アタッチされたクエリを使用してストリームまたはウィンドウを作成します。

CCL でクエリを作成するには、『CCL プログラマーズ・ガイド』を参照してください。

- 「文」で、CREATE STREAM、CREATE WINDOW、CREATE DELTA STREAM の各文を参照し、これらの文でサポートされる句を確認してください。
- 「句」で、構文と使用法の詳細を参照してください。

参照：
- *単純なクエリ* (36 ページ)

CCL エディタでのスキーマの作成

CCL エディタを使用して CREATE SCHEMA 文を入力し、多くのストリームやウィンドウから参照できる共有スキーマ・オブジェクトをユーザに提供します。
CCL エディタで、CREATE SCHEMA 文に対する有効な CCL を入力します。

- テキストを手動で入力します。
- [Create] > [Schema] を選択し、必要に応じて CCL コード案を編集します。

たとえば、次の文では、4 つのカラムを持つ、SchemaTrades1 という共有スキーマ・オブジェクトが作成されます。

```
CREATE SCHEMA        SchemaTrades1 (
Symbol STRING ,
Seller STRING ,
Buyer STRING ,
```

関数

関数とは、特定のタスクを実行する自己完結型の再利用可能なコード・ブロックです。

Sybase Event Stream Processor では、次の関数がサポートされます。

- 組み込み関数 - 集合関数、スカラ関数など
- ユーザ定義の SPLASH 関数
- ユーザ定義の外部関数

組み込み関数とはこのソフトウェアに付属する関数で、一般的な算術演算、集約、データ型変換、セキュリティの関数が含まれます。

演算の評価順序
関数内の演算は右から左に評価されます。これは、変数が別の演算に依存し、関数の実行前に変数を渡しておかなければ、その演算で予期しない結果が生じる可能性がある場合に重要です。例を示します。

```
integer a := 1;
integer b := 2;
max( a + b, ++a );
```

組み込み関数 **max()** は、カンマで区切られた値リストの最大値を返します。++a が最初に評価されるので 4 が返されます。そのため、予想されていた `max(3, 2)` ではなく `max(4, 2)` が実行されます。

演算子

CCL では、さまざまなタイプの数値演算子、非数値演算子、論理演算子がサポートされています。

算術演算子
算術演算子は、数値の符号切り替え、加算、減算、乗算、または除算に使用します。数値型に適用できますが、数値型の混在もサポートされます。算術演算子では 1 つまたは 2 つの引数を指定できます。単項演算子は引数と同じデータ型を返します。二項演算子は、数値的に優先順位が最も高い引数を選び、残りの引数をそのデータ型に暗黙的に変換して、その型を返します。

演算子	意味	使用例
+	加算	3+4

演算子	意味	使用例
-	減算	7-3
*	乗算	3*4
/	除算	8/2
%	剰余 (余り)	8%3
^	指数	4^3
-	符号切り替え	-3
++	インクリメント 前置インクリメント (++*argument*) 値は、増分されてから引数として渡されます。 後置インクリメント (*argument*++) 値は、渡されてから増分されます。	++a (前置インクリメント) a++ (後置インクリメント)
--	デクリメント 前置デクリメント (--*argument*) 値は、減分されてから引数として渡されます。 後置デクリメント (*argument*--) 値は、渡されてから減分されます。	--a (前置デクリメント) a-- (後置デクリメント)

比較演算子
比較演算子は、2 つの式を比較します。比較の結果は、TRUE、FALSE、または null となります。

比較演算子は、次の構文を使用します。

```
expression1 comparison_operator expression2
```

演算子	意味	使用例
=	等しい	a0=a1
!=	等しくない	a0!=a1
<>	等しくない	a0<>a1
>	より大きい	a0!>a1
>=	以上	a0!>=a1
<	より小さい	a0!<a1

演算子	意味	使用例
<=	以下	a0!<=a1
IN	値のリストのメンバ。値が式リストの値にある場合、結果は TRUE となります。	a0 IN (a1, a2, a3)

論理演算子

演算子	意味	使用例
AND	すべての式が TRUE の場合は TRUE を、その他の場合は FALSE を返す。	(a < 10) AND (b > 12)
NOT	すべての式が FALSE の場合は TRUE を、その他の場合は TRUE を返す。	NOT (a = 5)
OR	いずれかの式が TRUE の場合は TRUE を、その他の場合は FALSE を返す。	(b = 8) OR (b = 6)
XOR	1 つの式が TRUE でその他が FALSE の場合は、TRUE を返す。両方共に TRUE または FALSE の場合、FALSE を返します。	(b = 8) XOR (a > 14)

文字列演算子

演算子	意味	使用例
+	文字列を連結し、別の文字列を返す。 **注意：** + 演算子は、データ型の混在 (整数と文字列など) をサポートしません。	'go' + 'cart'

LIKE 演算子

カラム式と **WHERE** 句式で使用できます。LIKE は LIKE 演算子と REGEXP_LIKE 演算子の使用をサポートします。これらの演算子は、文字列式を、相互に非常に類似しているが正確には一致していない文字列に一致させます。

演算子	構文と意味	使用例
LIKE	**WHERE** 句の文字列式を、相互に非常に類似しているが正確には一致していない文字列に一致させる。 `compare_expression LIKE pattern_match_expression` LIKE 演算子は、**compare_expression** が **pattern_match_expression** に一致する場合は TRUE を、一致しない場合は FALSE を返します。式にはワイルドカードを使用できます。ここで、パーセント記号 (%) は任意の長さの文字列に一致し、アンダースコア (_) は単一の文字に一致します。	Trades.StockName LIKE "%Corp%"

[] 演算子

[] 演算子は、辞書とベクトルのコンテキストでのみサポートされます。

演算子	構文と意味	使用例
[]	ストリームまたはウィンドウ内の現在のローではないローを対象に関数を実行することを可能にする。 `stream-or-window-name[index].column` **stream-or-window-name** は、ストリームまたはウィンドウの名前で、**column** はストリームまたはウィンドウ内のカラムを示します。**index** はリテラル、パラメータ、または演算子を指定できる式で、整数に評価されます。この整数は、現在のローまたはウィンドウのソート順を基準とする、ストリームまたはウィンドウのローを示します。	MyNamedWindow[1].MyColumn

演算子の評価順序

複数の演算子のある式を評価する場合、優先度の高い演算子が低い演算子よりも先に評価されます。優先度が同じである場合、式内で左側から右側への順で評価されます。カッコを使用して演算子の優先度を変更できます。エンジンは、カッコの内側の式をその外側の式よりも先に評価します。

注意： ^ 演算子は、右から左に評価されます。したがって、a ^ b ^ c = a ^ (b ^ c) であり、(a ^ b) ^ c ではありません。

演算子の優先度を以下に示します。同一行の演算子は、同じ優先度を持ちます。

- +.- (単項演算子の場合)
- ^
- *、/、%
- +、- (二項演算子と連結の場合)
- =、!=、<>、<、>、<=、>= (比較演算子)
- LIKE、IN、IS NULL、IS NOT NULL
- NOT
- AND
- OR、XOR

第 5 章　スタジオでのプロジェクトの接続

[Run Project] をクリックしてプロジェクトを接続します。ローカル・クラスタが直ちに起動されます。

ローカル・クラスタが既に稼働している場合は、[Run Project] をクリックしてプロジェクトを追加します。

リモート・クラスタは ESP スタジオの外部で起動されます。リモート・クラスタに接続するには、クラスタの URL を入力します。

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。

参照：
- *リモート・クラスタへの接続* (78 ページ)
- *サーバの認証の変更* (79 ページ)
- *スタジオでのサーバ認証* (79 ページ)
- *複数のプロジェクトの実行* (80 ページ)

リモート・クラスタへの接続

スタジオ環境設定を使用して、リモート・クラスタの接続と認証方法を管理します。

前提条件
管理者が ESP スタジオの外側でリモート・クラスタを起動します。

手順

1. 新しいリモート・クラスタ接続を追加するには、[Server] ビューのツールバーの [New Server URL] を選択します。

   注意： また、[Studio Preferences] > [Sybase Event Stream Processor Studio] > [Run Test Preferences] を選択して新しい接続を追加することもできます。[New] を選択します。
2. クラスタ・マネージャの値を追加します。
3. (オプション) クラスタ・マネージャ接続の暗号化を有効にするには、[SSL] を選択します。
4. 認証方法を選択します。

認証方法	オプション
None	認証に関するフィールドは無効になります。
RSA	• **[RSA User]：** – キー・エイリアスを指定します。 • **[RSA Password]：** – キーストアのパスワードを指定します。 • **[RSA Key store]：** – プライベート・キーがあるキーストアのファイル名を指定します。
User/ Password	認証に関するフィールドは無効になります。

5. [OK] をクリックします。

[Run-Test] パースペクティブの [Server] ビューから、保存されているサーバ接続のリストにアクセスします。認証方法に応じて、スタジオは直ちに接続を試みるか ([None] モードと [RSA] モードの場合)、[User/Password] 認証が設定されている各クラスタではログイン・ダイアログが表示されます。

注意： リスト内のすべてのサーバに接続するには、[Server] ビューのツールバーの [Reconnect All] を選択します。

参照：
- *ローカル・クラスタへの接続* (77 ページ)
- *サーバの認証の変更* (79 ページ)
- *スタジオでのサーバ認証* (79 ページ)
- *複数のプロジェクトの実行* (80 ページ)

スタジオでのサーバ認証

スタジオには、セキュリティ認証として [None]、[User/Password]、[RSA] の 3 つのオプションがあります。

スタジオで使用される認証情報は認証方法によって異なります。[None] の場合は、ユーザ名もパスワードも不要です。[LDAP] と [KERBEROS] の場合は、ユーザ名とパスワードが必要ですが、スタジオでは保存されません。

Event Stream Processor でサポートされる認証方法の詳細については、『管理者ガイド』を参照してください。

参照：
- *ローカル・クラスタへの接続* (77 ページ)
- *リモート・クラスタへの接続* (78 ページ)
- *サーバの認証の変更* (79 ページ)

サーバの認証の変更

既に設定されているサーバの認証設定を変更します。

1. [Server] ビューで、[Studio Preferences] > [Sybase Event Stream Processor Studio] > [Run Test Preferences] を選択します。
2. 既存のサーバ接続を選択します。
3. [Edit] をクリックして認証オプションを変更します。
4. [OK] をクリックして変更内容を保存します。

参照：
- *ローカル・クラスタへの接続* (77 ページ)
- *リモート・クラスタへの接続* (78 ページ)
- *スタジオでのサーバ認証* (79 ページ)

複数のプロジェクトの実行

スタジオでは、ユーザはローカル・クラスタまたはリモート・クラスタに接続して、さまざまなワークスペースで複数のプロジェクトを実行できます。

クラスタは 1 つ以上のワークスペースで構成され、各ワークスペースには 1 つ以上のプロジェクトが含まれます。これらのプロジェクトは実行したり停止したりできます。すべてのワークスペースは 1 つのサーバ内にあるので、ユーザは複数のプロジェクトを同時に処理できます。

クラスタ内で作業するには、ユーザは管理者からライセンス情報または認証情報のどちらかを取得する必要があります。

ローカル・クラスタ
ローカル・クラスタでは、ユーザはローカル・マシンからプロジェクトを処理できます。インターネット・アクセスは必要ありません。

リモート・クラスタ
リモート・クラスタを使用すれば、ユーザはデフォルト・サーバより高性能のサーバに接続できます。手動入力、プレイバック、その他のスタジオ機能を使用できます。また、リモート・クラスタでは、ユーザはクラスタ内のプロジェクトを他のユーザと共有することもできます。

リモート・クラスタは、接続権限をユーザに付与する管理者がセットアップします。

参照：
- *ローカル・クラスタへの接続* (77 ページ)
- *リモート・クラスタへの接続* (78 ページ)
- *複数のワークスペースの処理* (80 ページ)
- *プロジェクトの実行* (84 ページ)

複数のワークスペースの処理

複数のワークスペース内の複数のプロジェクトを同時に処理するには、新しいワークスペースを作成し、そこに新規プロジェクトや既存のプロジェクトを追加します。

ワークスペースでは、そこで実行されるプロジェクトに保護下の名前空間が提供されます。したがって、同じプロジェクトの 2 つのインスタンスを、同じクラスタ上の異なるワークスペースでそれぞれ実行でき、名前の競合を回避できます。

1. [Server] ビューで、サーバの URL を右クリックします。
2. [Create Workspace] を選択します。
3. 次のように、ワークスペースの有効な名前を入力します。
   - 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。
4. [OK] をクリックします。
   新しいワークスペースは空です。そこで新しいプロジェクトを作成するか、既存のプロジェクトをロードすることができます。
5. 既存のプロジェクトをワークスペースに追加するには、次の手順に従います。
   a) [Load Project(s) into Workspace] を選択します。
   b) ワークスペースに追加するコンパイル済みの CCX ファイルを選択して、[OK] をクリックします。
6. (オプション) プロジェクトを右クリックして開始または停止します。
7. (オプション) ワークスペースからプロジェクトを削除するには、削除するプロジェクトを右クリックして、[Remove Project] を選択します。

参照：

- *複数のプロジェクトの実行* (80 ページ)

第 6 章　プロジェクトの実行とテスト

プロジェクトをテストするには、サーバでプロジェクトをコンパイルして実行し、ストリームにアクセスしてフィルタ処理を行い、データを保存して Sybase Event Stream Processor サーバにアップロードして、プロジェクト設定を作成します。

[Run-Test] パースペクティブの開始

[Run-Test] パースペクティブのツールバーとビューを使用して、Event Stream Processor プロジェクトのテスト、モニタ、デバッグ、検査を簡素化します。スタジオのメイン・ウィンドウの上部にある [Run-Test] タブをクリックすると、[Run-Test] パースペクティブが表示されます。

[Run-Test] タブが表示されない場合は、メイン・メニューから [Window] > [Open Perspective] > [Run-Test] を選択します。

プロジェクトのコンパイル

CCL コードから実行可能 `.ccx` ファイルを生成します。Event Stream Processor で実行する実行可能ファイルを生成するには、CCL コードをコンパイルします。

1. (オプション) CCL コンパイラ・オプションを設定します。
   a) [Edit] > [Preferences] を選択します。
   b) ツリー・ビューを展開して [Sybase Event Stream Processor] > [Run Test Preferences] > [Compiler Options] を選択します。
   c) コンパイル済みプロジェクト用のディレクトリを変更するには、[Change] をクリックしてディレクトリを選択し、[OK] をクリックします。
   d) その他の変更内容を確認するには、[OK] をクリックします。

   注意： デフォルトでは、コンパイル用ディレクトリは `bin` に設定されます。つまり、`.ccx` ファイルは、プロジェクトのディレクトリを基準とするサブディレクトリに作成されます。
2. [Authoring] パースペクティブの [File Explorer] でツリー・ビューを展開すると、プロジェクトの `.ccl` ファイルが表示されます。
3. コンパイルして実行する `.ccl` プロジェクトを選択して開きます。
4. [Run Project] を選択します。

プロジェクトは自動的にコンパイルされて実行されます。[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。正常に接続されると、サーバ・フォルダの下にサーバ・ストリームが表示されます。接続が失敗すると、サーバ接続エラーのダイアログが表示されます。

スタジオでは、プロジェクトに属するすべてのオープン・ファイルの保存、プロジェクトのコンパイルと実行、`.ccx` ファイル (コンパイル済みの実行可能ファイル) の作成がサイレントに実行されます。コンパイル・エラーは、エラーの種類に応じて、[Problems] ビューまたは [Console] ビューに表示されます。

プロジェクトの実行

プロジェクトを実行すると、ローカル・クラスタまたは別の接続先クラスタのどちらかでプロジェクトが自動的に開始します。

前提条件
デフォルト以外のワークスペースでプロジェクトを実行するには、1 つ以上の接続先ワークスペースが使用可能であることを確認します。

手順

1. 実行する `.ccl` ファイルを選択して開きます。
   エディタが開いていない場合は、実行するプロジェクトを選びます。
2. 次のいずれかの手順に従って、プロジェクトを実行します。
   - メイン・ツールバー ([Authoring] パースペクティブまたは [Run-Test] パースペクティブのどちらか) の [Run Project] をクリックしてデフォルトのワークスペースでプロジェクトを実行します。または、
   - [Run Project] ツールの隣にあるドロップダウン矢印をクリックして [Run Project in Workspace] を選択します。次に、このプロジェクトの実行先のワークスペースを選択します。

   プロジェクトが実行され、[Run-Test] パースペクティブに結果が表示されます。

参照：
- *複数のプロジェクトの実行* (80 ページ)

[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。

参照：

- *第 5 章、「スタジオでのプロジェクトの接続」* (77 ページ)
- *パフォーマンス・モニタ* (99 ページ)
- *[Event Tracer] ビュー* (105 ページ)

プロジェクト設定

プロジェクト設定は XML ドキュメントです。これによって、ストリームの URI バインド、アダプタ・プロパティ、パラメータ値、高度な配備オプションなど、プロジェクトの具体的なランタイム・プロパティを制御します。

プロジェクト設定ファイルの作成と編集は、アタッチ先のプロジェクトとは別個に行います。このファイルは、`.ccr` ファイル拡張子で識別されます。プロジェクト設定ファイルは、[Authoring] パースペクティブの [File Explorer] ビューに表示して編集します。

設定ファイルは CCL の外側ですべてのランタイム・プロパティを維持します。したがって、ランタイム・プロパティを変更する一方で、バージョン管理下で CCL ファイルと CCX ファイルを維持できます。これで、CCL ファイルと CCX ファイルを変更せずにテスト環境から運用環境にプロジェクトを移行できます。

デフォルトでは、新しいプロジェクトが作成されると、新しいプロジェクト設定ファイルも作成されます。Aleri モデルを Event Stream Processor プロジェクトに変換した場合も、新しい設定ファイルが作成されます。1 つのプロジェクトに複数の設定ファイルをアタッチできるので、新しいプロジェクト設定を手動で作成できます。

プロジェクト設定の作成

プロジェクト設定を作成し、設定プロパティを編集します。新しいプロジェクトを作成すると、プロジェクト設定ファイルが自動的に生成されます。

1. [File] - [New] - [Project Configuration] を選択します。
2. 新しい設定ファイルの保存先フォルダを選択し、ファイル名を割り当てます。
3. [Finish] をクリックします。
   CCR プロジェクト設定エディタのウィンドウが表示されます。

参照：

- *既存のプロジェクト設定を開く* (86 ページ)
- *プロジェクト設定ファイルの編集* (86 ページ)

既存のプロジェクト設定を開く

既存のプロジェクト設定ファイルを開きます。

デフォルトでは、新しいプロジェクトを作成するとプロジェクト設定が作成されるので、プロジェクトごとに少なくとも 1 つの既存のプロジェクト設定があります。

1. [Window] - [Open Perspective] - [Authoring] を選択するか、[Authoring] タブをクリックします。
2. [Window] - [Show View] - [File Explorer] を選択します。
3. プロジェクト設定ファイルを見つけます。このファイルは `<projectname>.ccr` として表示されます。ファイルをダブルクリックして開きます。

参照：

- *プロジェクト設定の作成* (86 ページ)
- *プロジェクト設定ファイルの編集* (86 ページ)

プロジェクト設定ファイルの編集

CCR プロジェクト設定エディタは 5 つの別々の設定カテゴリーに分割されています。

次の設定プロパティを定義および変更します。

1. [Cluster Parameters]：プロジェクトのバインドを設定し、プロジェクトのバインド先のクラスタのプロパティを定義します。
2. [Bindings]：バインドを設定し、データ・ソースへのバインドを定義します。

3. [Adapter Properties]：入力アダプタか出力アダプタまたはその両方の設定を制御するアダプタ・プロパティ・セットを設定します。
4. [Viewing parameters]：値の表示と割り当てを行います。
5. [Advanced options]：配備プロパティ、プロジェクト・オプション、プロジェクト・インスタンス、プロジェクトの結び付きを変更します。

1. *プロジェクト設定でのクラスタ・パラメータの編集*
   プロジェクトのクラスタを設定し、クラスタ設定を行います。
2. *プロジェクト設定でのバインドの編集*
   あるプロジェクトの入力ストリームまたは入力ウィンドウと他のプロジェクトのデータ・ソースとのバインドを設定します。
3. *プロジェクト設定でのアダプタ・プロパティ・セットの編集*
   CCR プロジェクト設定エディタを使用して、プロジェクト設定ファイルでアダプタ・プロパティ・セットを設定します。プロパティ・セットとは、プロジェクト設定ファイルに保存されている再利用可能な一連のプロパティです。
4. *プロジェクト設定でのパラメータの設定*
   パラメータの定義を編集し、削除されたパラメータを除去します。
5. *プロジェクト設定での詳細オプションの編集*
   プロジェクト設定ファイルで、プロジェクト配備プロパティ、プロジェクト・オプション、プロジェクト・インスタンスを変更します。

参照：
- *プロジェクト設定の作成* (86 ページ)
- *既存のプロジェクト設定を開く* (86 ページ)
- *プロジェクト間のバインド* (87 ページ)
- *クラスタ* (88 ページ)

プロジェクト間のバインド
あるプロジェクトの出力を別のプロジェクトの入力にバインドします。バインドは実行時に指定されます。

プロジェクトを別のプロジェクトにバインドして、プロジェクト A の入力ストリームまたは入力ウィンドウがプロジェクト B の出力ストリームまたは出力ウィンドウから入力を受け取るように設定できます。プロジェクトのバインドは概念的には入力アダプタを入力ストリームまたは入力ウィンドウにアタッチする処理と似ていますが、一方のプロジェクトの出力ともう一方のプロジェクトの入力の間に直接接続が確立されるので、より効率的です。

バインドは、ローカルでも、同じクラスタ内でも設定できます。また、あるクラスタ内のプロジェクトを別のクラスタ内のプロジェクトに接続することもできます。バインド情報は、プロジェクト設定ファイル内に CCL で指定します。このた

め、実行時にバインド参照を変更できるので、プロジェクトを複数の環境で使用できるようになります。

参照：
- *プロジェクト設定ファイルの編集* (86 ページ)
- *クラスタ* (88 ページ)
- *ストリームのバインド・ルール* (90 ページ)
- *プロジェクト設定でのバインドの編集* (89 ページ)

クラスタ

スタジオでは、ユーザはプロジェクトを接続できる入力用のクラスタを指定できます。ローカル・クラスタでも、リモート・クラスタでも追加できます。これらはバインドを設定する場合に使用されます。

クラスタでは、処理作業を複数のサーバに分割してパフォーマンスを向上させます。リモート・クラスタには子クラスタ・マネージャを割り当てることができ、これを使用してユーザはクラスタ・ノードのパラメータを設定できます。クラスタ・ノードはクラスタ・マネージャの子コンポーネントとして表示されます。ローカル・クラスタには子クラスタ・マネージャを割り当てられません。

ユーザは、プロジェクト設定ファイルの [Cluster] タブでローカル・クラスタまたはリモート・クラスタを追加したり削除したりできます。

[Cluster] タブには、[All Clusters] と [Cluster Details] の 2 つのペインがあります。[All Clusters] ペインには、作成されたすべてのクラスタが表示されます。このペインで 1 つのクラスタを選択して、そのクラスタのプロパティを編集できます。

[Cluster Details] ペイン内でクラスタ・プロパティの編集に使用できるオプションは、次のとおりです。

フィールド	説明
Name	クラスタに適用された名前を変更するには、このフィールドを編集します。
Type	ローカル (サーバ情報が必要ない) とリモート (サーバ情報が必要) のクラスタ接続オプション間を切り替えます。
User Name	クラスタにユーザ名を適用し、必要に応じて暗号化します。
Password	クラスタにパスワードを適用し、必要に応じて暗号化します。

注意： プロジェクトを接続できるクラスタを指定することは、クラスタに接続することとは異なります。接続については、「ローカル・クラスタへの接続」と「リモート・クラスタへの接続」を参照してください。

参照：
- *プロジェクト設定ファイルの編集* (86 ページ)
- *プロジェクト間のバインド* (87 ページ)
- *プロジェクト設定でのクラスタ・パラメータの編集* (89 ページ)

プロジェクト設定でのクラスタ・パラメータの編集
プロジェクトのクラスタを設定し、クラスタ設定を行います。

1. CCR プロジェクト設定エディタのウィンドウで [Cluster] タブを選択します。
2. クラスタを追加するには、[Add] をクリックします。
3. クラスタを編集するには、クラスタをクリックして、[Cluster Details] ペインで個々のプロパティを変更します。
4. (オプション) クラスタのユーザ名またはパスワードを暗号化するには、次の手順に従います。
   a) ユーザ名とパスワードを入力して [Encrypt] をクリックします。
   b) [Cluster URI] (ホスト名とポート番号 (`<HOST>:<PORT>`) で構成)、クレデンシャルのフィールド (必須の場合) など、必須フィールドに入力します。
   c) [Encrypt] をクリックします。
   クラスタのユーザ名またはパスワードが暗号化されて、ランダムに選択された暗号化文字がフィールドに挿入されます。

   注意： 暗号化をリセットするには、該当するフィールドの横の [Encrypt] をクリックします。[Reset] をクリックします。
5. マスタ・クラスタ・ノードと子クラスタ・ノードを追加するには、次の手順に従います。
   a) [Cluster Properties] で、タイプとして [remote] を選択します。
   b) クラスタを右クリックして、[New] - [Cluster Manager] を選択します。
   c) クラスタ・ノードを選択し、[Cluster Manager Details] ペインの [Cluster Manager] フィールドにホストとポートに関する情報を追加して、各クラスタ・ノードを設定します。

参照：
- *クラスタ* (88 ページ)

プロジェクト設定でのバインドの編集
あるプロジェクトの入力ストリームまたは入力ウィンドウと他のプロジェクトのデータ・ソースとのバインドを設定します。

現在のプロジェクトの入力ストリームに使用する他のプロジェクトのデータ・ソースにバインドを割り当てることができます。

1. CCR プロジェクト設定エディタで [Bindings] タブを選択します。
2. バインドを追加するには、[Add] をクリックします。
3. バインド項目を個別に設定するには、CCR プロジェクト設定エディタの右側にある [Binding Details] ペインを使用します。

フィールド	説明
Binding name	(オプション) バインドに名前を付けます。
Local stream/ window	バインドとストリームにタイトルを付けます。バインド名は、このフィールドに挿入した名前になります。
Cluster	バインド先のクラスタを選択します。 **注意：** 1 つ以上のクラスタを事前に定義しておきます。
Workspace	ワークスペースのデータ (たとえば、ws1) を入力します。
Project	アクセス対象のプロジェクト (たとえば、project1) を入力します。
Remote stream	リモート・ストリームの情報 (たとえば、remoteStream1) を入力します。

4. バインドを削除するには、バインドを選択して、[Remove] をクリックします。

参照：
- *プロジェクト設定でのアダプタ・プロパティ・セットの編集* (91 ページ)
- *プロジェクト間のバインド* (87 ページ)
- *ストリームのバインド・ルール* (90 ページ)

ストリームのバインド・ルール
バインドを確実に成功させるには、バインド・ルールに従います。

Event Stream Processor では、出力ストリームから入力ストリームへのバインドのみがサポートされます。ストリームは、プロジェクト間でのみバインドでき、プロジェクト内ではバインドできません。バインドを実行するには、両方のストリームが次の条件を満たす必要があります。

- スキーマに互換性があること。
- 各フィールド名のデータ型が同じであること。
- カラムの順序が同じであること。
- カラムの数が同じであること。

プロジェクト内のカラム名とストリーム名は、一致する必要はありません。

参照：
- *プロジェクト間のバインド* (87 ページ)

- *プロジェクト設定でのバインドの編集* (89 ページ)

プロジェクト設定でのアダプタ・プロパティ・セットの編集

CCR プロジェクト設定エディタを使用して、プロジェクト設定ファイルでアダプタ・プロパティ・セットを設定します。プロパティ・セットとは、プロジェクト設定ファイルに保存されている再利用可能な一連のプロパティです。

プロパティ・セットはツリー形式で表示され、個々のプロパティ定義はプロパティ・セットの子として表示されます。

1. CCR プロジェクト設定エディタで [Adapter Properties] タブを選択します。
2. 新しいアダプタ・プロパティ・ノードを作成するには、[Add] をクリックします。
3. [Property Set Details] ペインでプロパティ・ノードの名前を定義します。
4. プロパティ・セットに新しいプロパティを追加するには、セットを右クリックして、[New] - [Property] を選択します。

   注意： プロパティ・セットにはプロパティ項目を必要な数だけ追加できます。
5. プロパティを設定するには、次の手順に従います。
   a) [Property Details] ペインでプロパティの名前を定義します。
   b) プロパティの値を入力します。
6. (オプション) プロパティの値を暗号化するには、次の手順に従います。
   a) プロパティの値を選択して、[Encrypt] をクリックします。
   b) [Cluster URI]、クレデンシャルのフィールドなど、必須フィールドに入力します。
   c) [Encrypt] をクリックします。
      値と、関連フィールドに、ランダムに選択された暗号化文字が挿入されます。

      注意： 暗号化をリセットするには、該当するフィールドの横の [Encrypt] をクリックします。必要に応じて値を変更し、[Reset] をクリックします。
7. [All Adapter Properties] リストから項目を削除するには、次の手順に従います。
   - プロパティ・セットを右クリックして、[Remove] を選択します。または、
   - プロパティを右クリックして、[Delete] を選択します。

参照：

- *プロジェクト設定でのバインドの編集* (89 ページ)

プロジェクト設定でのパラメータの設定
パラメータの定義を編集し、削除されたパラメータを除去します。

パラメータ定義のリストには、プロジェクト・フォルダの CCL ドキュメント内のパラメータに基づいて自動的に値が挿入されます。パラメータ定義の値は変更できます。CCL ドキュメントから定義が削除された場合は、パラメータも除去できます。

1. CCR プロジェクト設定エディタで [Parameters] タブを選択します。
2. パラメータの値を変更するには、[Parameter Details] ペインでパラメータをクリックして値を変更します。

   注意： パラメータの [Name] フィールドは変更できません。
3. 削除されたパラメータ定義をリストから除去するには、リストの上部にある [Remove] を選択します。

   注意： (removed) のマークが付いているパラメータ定義は元の CCL ファイルから削除されているので、パラメータ定義リストから除去できます。

参照：
- *プロジェクト設定での詳細オプションの編集* (94 ページ)

高度なプロジェクト配備オプション
プロジェクト配備オプションでは、クラスタへのプロジェクトの配備方法と実行時のプロジェクトの動作方法を指定します。プロジェクト・オプション、アクティブ／アクティブ・インスタンス、フェールオーバ間隔、プロジェクト配備タイプ・オプションを含め、これらのパラメータは、CCR ファイルに手動で設定するか、ESP スタジオで設定します。

アクティブ／アクティブの配備
アクティブ／アクティブの配備は、CCR ファイルにプロジェクトを `ha-project` として定義した場合のみ使用できます。アクティブ／アクティブの配備とは、プロジェクトの 2 つのインスタンスをクラスタ内で同時に実行することです。プロジェクトの 2 つのインスタンスは 2 つの異なるホストでクラスタ・マネージャによって開始されます。

プロジェクトの一方のインスタンスがプライマリ・インスタンスとして選択されます。インスタンスの一方が既にアクティブな場合、そのインスタンスがプライマリ・インスタンスです。障害が発生したインスタンスが再開すると、そのインスタンスはセカンダリ・ポジションとなり、現行のインスタンスで障害が発生するか現行のインスタンスが停止するまで、そのポジションを維持します。

プロジェクト・オプション
プロジェクト・オプションはプロジェクトのランタイム・パラメータとして使用され、使用可能なオプション名の事前定義リストが含まれます。これには、ほとんどのコマンド・ライン・エントリが反映されています。

インスタンス
使用可能なインスタンスの数は、ユーザが選択した配備タイプ ([HA] (高可用性) または [Non-HA]) によって異なります。プロジェクトが HA (アクティブ／アクティブ) モードで設定されている場合は、プライマリとセカンダリの 2 つのインスタンスが作成されます。フェールオーバ間隔、間隔あたりの障害数など、インスタンスごとに結び付きとコールド・フェールオーバに関するオプションを設定できます。

フェールオーバ
プロジェクトは、適切に実行されなかったり、適切に実行が停止されなかったりすると、障害が発生します。障害が発生したプロジェクトまたはサーバが別のサーバに切り替わって処理を続行した場合、フェールオーバが発生します。フェールオーバによってプロジェクトが再開されることもあります (定義されている場合)。再開は、障害の間隔と間隔あたりの再開数の定義に基づいて制限できます。フェールオーバ・オプションにアクセスするにはインスタンス設定を使用します。次のようなオプションがあります。

フィールド	説明
Failover	[enabled] または [disabled]。[disabled] の場合、プロジェクトのフェールオーバによる再開は実行できません。[enabled] の場合、[Failure interval] と [Failures per interval] の各フィールドにアクセスでき、再開を実行できます。
Failures per interval	プロジェクトが所定の間隔内で再開を試行できる回数を指定します。この数は、プロジェクトを手動で開始したり、障害の発生時点からの期間が間隔より長いために障害がリストから消去されたりした場合に、ゼロにリセットできます。
Failure interval	(オプション) ここには、間隔の時間を秒単位で指定します。ブランクのままにすると、間隔の時間は無制限に設定されます。

結び付き
Affinities (結び付き) では、クラスタ内でプロジェクトを実行するかどうかを制限します。結び付きには次の 2 つのタイプがあります。

- コントローラ - アクティブ／アクティブと非アクティブ／アクティブの設定に使用します。各コントローラには複数の結び付きを割り当てることができますが、strong positive として設定できるコントローラの結び付きは 1 つのみです。
- インスタンス - アクティブ／アクティブの設定にのみ使用します。インスタンスの場合は、それぞれ別々のプロジェクト・サーバに適用できる 2 つの結び付きが作成されます。

結び付きごとに定義するパラメータは、次のとおりです。

フィールド	説明
name	結び付きのオブジェクトの名前、つまり結び付きの設定対象であるコントローラ名またはインスタンス名を入力します。インスタンスの結び付きの場合、1 つのインスタンスの結び付きは 2 つ目のインスタンスを参照します。
strength	[strong] または [weak]。strong の場合、プロジェクトは特定のコントローラで実行する必要があります。その他のコントローラは使用できません。weak の場合、プロジェクトは定義されたコントローラで優先して開始されます。ただし、そのコントローラを使用できない場合は、別の使用可能なコントローラで開始できます。
charge	[positive] または [negative]。positive の場合、プロジェクトはコントローラで実行されます。negative の場合、プロジェクトはコントローラで実行されません。

参照：

- *プロジェクト設定での詳細オプションの編集* (94 ページ)

プロジェクト設定での詳細オプションの編集

プロジェクト設定ファイルで、プロジェクト配備プロパティ、プロジェクト・オプション、プロジェクト・インスタンスを変更します。

1. CCR プロジェクト設定エディタで [Advanced] タブを選択します。
2. プロジェクト配備項目がない場合は、[Add] を選択します。
3. [Project Deployment Detail] ウィンドウでプロジェクト配備タイプを選択します。オプションは次のとおりです。

タイプ	説明
Non-HA	配備タイプが [Non-HA] の場合は、プロジェクト配備項目の下に 1 つのプロジェクト・オプション項目と 1 つのインスタンス項目が子として作成されます。

タイプ	説明
HA	配備タイプが [HA] の場合は、プロジェクト配備項目の下に 1 つのプロジェクト・オプション項目と 2 つのインスタンス項目が子として作成されます。[HA] では、インスタンス間のホット・プロジェクト・フェールオーバが実現されます。

4. オプションを追加するには、プロジェクト・オプション項目を右クリックして [New] - [option] を選択します。

   次の表に、ESP スタジオの [Project Configuration] ビューを使用して設定できる使用可能なすべてのプロジェクト・オプションをまとめます。

プロジェクト・オプション	説明
on-error-discard-record	true の場合、計算エラーが発生すると、計算対象のレコードは破棄されます。false に設定した場合、計算対象外のすべてのカラムに NULL が埋め込まれ、レコード処理は続行します。デフォルト値は true です。 **注意：** キー・カラムの計算エラーが発生した場合は、このオプションの設定にかかわらずレコードは破棄されます。
on-error-log	true に設定した場合、計算エラーが発生するとエラー・メッセージに記録されます。デフォルト値は true です。
java-classpath	Java クラス・パスを設定します。値は classpath ファイルのファイル・パスです。
java-max-heap	プロジェクトの最大 Java ヒープを設定します。デフォルト値は 256 メガバイトです。
utf8	サーバ上の UTF-8 機能を有効にします (デフォルトでは、この機能はオフです)。デフォルト値は false です。true に設定すると有効になります。
precision	プロジェクトで数字を表す場合の小数点以下の桁数を設定します。デフォルト値は 6 です。
command-port	コマンド・ポート番号を設定します。一般に、この詳細オプションは設定しないでください。ポートが 0、または 1 ～ 65535 の範囲外の場合は、プログラムによって任意のポートが選択されます。特定のポートを定義するには、1 ～ 65535 の値を設定します。
sql-port	SQL ポート番号を設定します。一般に、この詳細オプションは設定しないでください。ポートが 0、または 1 ～ 65535 の範囲外の場合は、プログラムによって任意のポートが選択されます。特定のポートを定義するには、1 ～ 65535 の値を設定します。

プロジェクト・オプション	説明
gateway-port	ゲートウェイ・ポート番号を設定します。一般に、この詳細オプションは設定しないでください。ポートが 0、または 1 ～ 65535 の範囲外の場合は、プログラムによって任意のポートが選択されます。特定のポートを定義するには、1 ～ 65535 の値を設定します。
time-granularity	プロジェクト内の時間粒度を定義します。このオプションでは、実行中の Event Stream Processor からパフォーマンス・レコード (ストリームあたり 1 つとゲートウェア接続あたり 1 つ) のセットを取得する頻度を秒単位で指定します。デフォルトでは、時間粒度は 5 に設定されます。このオプションを 0 に設定すると、モニタは無効になります。この場合は、パフォーマンスも最適化されます。
debug-level	プロジェクトをデバッグする場合のロギング・レベルを 0 ～ 7 の範囲で設定します。各番号の意味は、次のとおりです。 • 0：LOG_EMERG - システムが使用不可能である • 1：LOG_ALERT - アクションを直ちに実行しなければならない • 2：LOG_CRIT - 重大な状態 • 3：LOG_ERR - エラー状態 • 4：LOG_WARNING - 警告状態 • 5：LOG_NORMAL - 正常だが重要な状態である • 6：LOG_INFO - 通知 • 7：LOG_DEBUG - デバッグ・レベル・メッセージ
memory	プロジェクトのメモリ使用の上限を設定します。デフォルトは 0 で、無制限を意味します。
optimize	冗長なストア更新を実行しません。デフォルト値は false で、true に設定すると有効になります。
ignore-config-topology	これを有効にすると、プロジェクト間のトポロジが無視されます。デフォルトは false で、true に設定すると有効になります。
time-interval	ウィンドウでのローの最大表示時間を指定する、定数の間隔式を設定します。デフォルトでは、秒単位で 0 に設定されています。これは、タイマなしを意味します。

注意： 各プロジェクト・オプションを追加できるのは 1 回のみです。実装されたプロジェクト・オプションはドロップダウン・リストで選択できなくなります。

5. オプション項目を設定するには、次のフィールドに入力します。

オプション	説明
Name	前述の表に示されている使用可能なオプションのリストから選択します。
Value	プロパティ・オプションの値を入力します。

6. インスタンス項目の下に結び付きを追加するには、インスタンス項目を右クリックして [New] - [affinity] を選択します。次のフィールドに入力します。

オプション	説明
Name	結び付き項目の名前を入力します。
Strength	強さの度合いを選択します。
Type	タイプを選択します(たとえば、[controller])。
Charge	positive/negative を選択します。

7. [All Advanced Configurations] リストから項目を削除するには、次の手順に従います。
   - プロジェクト配備項目を選択して、[Remove] をクリックします。
   - オプションまたは結び付き項目を右クリックして、[Delete] を選択します。

参照：
- *プロジェクト設定でのパラメータの設定* (92 ページ)
- *高度なプロジェクト配備オプション* (92 ページ)

ストリームの表示

[Stream] ビューには、実行中のプロジェクトの出力ストリーム内のすべてのイベントと出力ウィンドウ内のすべての保存イベントが表示されます。

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て、[Show In] > [StreamViewer] (または [New StreamViewer]) を選択します。
   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. サブスクリプション・リスト、または個々のストリーム・サブスクリプションを操作するには、編集するサブスクリプションを選択し、[Stream] ビューの上部にある次のボタンのいずれかを選択します。
   - [Close Subscription URL] 。切断して、[Stream] ビューを閉じます。

- [Clear] 。コンテンツを消去してサブスクリプションを一時停止します。
- [Show Current Subscription in new View] 。入手可能な場合は、ストリームのパブリッシュ日が表示されます。

4. (オプション) [Stream] ビューのデータを保存するには、[Clipboard Copy] をクリックします。

ESP サーバへのデータのアップロード

アップロードしたデータを使用して、新たに作成されたプロジェクトをテストして実行します。

前提条件

ローカル・クラスタまたはリモート・クラスタのどちらかで、プロジェクトが実行されていることを確認します。

手順

1. [Run-Test] パースペクティブの左下のペインで、[File Upload] ビューを選択します。
2. [Select Server] ( ) をクリックします。
3. データのアップロード先のプロジェクトを選択して、[OK] をクリックします。
4. [Browse] をクリックしてファイル選択ダイアログを開き、アップロードする入力ファイルに移動します。
5. アップロードするファイルを 1 つ以上選択します。

注意： ESP サーバでは、ESP バイナリ (`.bin`)、ESP XML (`.xml`)、カンマ区切り値とテキスト (`.csv` または `.txt`) の各ファイルがサポートされています。

6. [Upload] をクリックします。進行状況バーにアップロード状況が示されます。

   [File Upload] ビューで実行できるその他のアクションは、次のとおりです。

UI コントロール	アクション
[Remove File]	[Input File(s)] メニューからこれまでに選択したファイルを破棄します。
[Cancel]	現在進行中のファイル・アップロードをキャンセルします。 **注意：** アップロードがキャンセルされる前に送信されたデータは処理されます。

UI コントロール	アクション
[Use Transaction]	ファイルを Event Stream Processor サーバへの 1 つのトランザクションとして処理します。
[Record Buffer]	複数のレコードをまとめてグループにし、そのグループを 1 つのトランザクションとして処理します。

ストリームへのデータの手動入力

ストリームまたはウィンドウへの入力として、イベントを手動で作成してパブリッシュします。

プロジェクトへの入力イベントの手動パブリッシュは、プロジェクトをテストする場合に役立ちます。

1. [Run-Test] パースペクティブの左下のペインで、[Manual Input] ビューを選択します。
2. [Select Stream] () をクリックします。
3. [Select Stream] ダイアログで、ストリームを選択して [OK] をクリックします。
4. 必要に応じて、使用可能なデータ・カラムを編集します。
5. データの複数のローを編集するには、[Edit Multiple Rows] () を選択して、変更するローを選択します。
6. ウィンドウにパブリッシュする場合は、opcode を指定します。ストリームにパブリッシュする場合は、イベントの挿入のみがサポートされます。
7. (オプション) [Use Current Date] を選択します。
8. [Publish] をクリックして、イベントをプロジェクトに送信します。

参照：

- *手動入力の設定* (116 ページ)

パフォーマンス・モニタ

[Monitor] ビューには、プロジェクト内の各ストリームとウィンドウ (ローカルのストリームとウィンドウを含む) のキューのサイズ、スループット、CPU 使用率の各ビジュアル・インジケータが表示されます。

各ノードはモデル内のストリームに対応し、直線によってデータが流れる経路が示されます。各ノードの色は、指定に応じて、[QueueDepth] または [Rows Processed (/sec)] のどちらかを表します。

たとえば、[Color Queue Depth] オプションを選択した場合、[(Red) Range >=] フィールドにはデフォルトで 125 が設定され、[(Yellow) Range >=] フィールドにはデフォルトで 20 が設定されます。これは、次のような意味です。

- ストリーム・ノードのキューの深さが 125 以上の場合、ノードは赤で表示されます。
- ストリーム・ノードのキューの深さが 20 ～ 124 の場合、ノードは黄色で表示されます。
- ストリーム・ノードのキューの深さが 20 より小さい場合、ノードは緑で表示されます。
- ノードが白のままの場合は、モニタが Stream Processor からデータを受信していないことを示します。

[Monitor] ビューでは、CPU 使用率もノードの省略記号に黒の円グラフとして示されます。選択したオプションに基づいて、省略記号の残りの部分は赤、黄、または緑で表示されます。完全に黒のノードは、シングル コア CPU では、CPU 使用率が 100% であることを示します。マルチコア環境またはマルチプロセッサ環境で完全に黒のノードは、CPU 使用率が 100% を超えている可能性があります。

カーソルを図中の特定のノードの上に移動すると、そのノードのパフォーマンス統計を確認できます。

モニタの実行

ストリームとウィンドウごとに、キュー・サイズと CPU 使用率の各ビジュアル・インジケータが表示されます。

前提条件

モニタを開始するには、事前にプロジェクトを実行しておきます。パフォーマンス・タイマの間隔を変更して遅延を指定できます。

手順

1. [Run-Test] パースペクティブで [Monitor] ビューを選択します。
2. [Select Running Project] ( ) をクリックします。
3. [OK] をクリックします。
4. [QueueDepth] または [Rows Processed] を選択して、パフォーマンス図で各ノードに色を設定する方法を指定します。次のどちらかを行います。
   - [(Red) Range >=] フィールドに数値を入力するか矢印ボタンを使用して、赤のノードを作成する範囲を選択します。

- [(Yellow) Range >=] フィールドに数値を入力するか矢印ボタンを使用して、黄色のノードを作成する範囲を選択します。

注意： [(Red) Range >=] と [(Yellow) Range >=] のどちらにも入らない範囲内のノードは緑で表示されます。

5. 図の拡大ビューまたは縮小ビューを表示するには、[Zoom In] または [Zoom Out] をクリックします。

参照：
- *イメージとしてのパフォーマンス図の保存* (101 ページ)

イメージとしてのパフォーマンス図の保存

パフォーマンス図を保存します。

[Run-Test] パースペクティブの [Monitor] ウィンドウを使用してパフォーマンス図のプロパティを変更できます。この図は [Event Tracer] ウィンドウに表示され、そのウィンドウからのみ保存できます。

1. [Run-Test] パースペクティブで [Event Tracer] ビューを選択します。
2. [Save] () をクリックします。
3. ファイル名と保存場所を入力します。[Save] をクリックします。
   ファイルは指定した場所に JPEG イメージとして保存されます。

参照：
- *モニタの実行* (100 ページ)

SQL クエリの実行

[SQL Query] ビューで、実行中のプロジェクトの出力ウィンドウに対してスナップショット SQL クエリを実行し、その結果を [Console] に表示します。

1. [Run-Test] パースペクティブの左下のペインで [SQL Query] ビューを選択します。
2. [Select Project] () をクリックします。
3. 表示されたメニューで、クエリ対象のプロジェクトとウィンドウを選択し、[OK] をクリックします。
4. クエリを入力します。
   たとえば、`Select * from <stream>` と入力します。
5. [Execute] をクリックします。

[Console] に結果が表示されます。

[Playback] ビュー

[Playback] ビューでは、送信中のデータがプレイバック・ファイルに記録され、取得されたデータが実行中の Event Stream Processor インスタンスに戻されて再生されます。

表 9 : [Playback] ビューのオプション

機能	説明
[Select playback file]	Event Stream Processor レコーダで使用するファイル形式を選択します。
[Start playback]	現在のプレイバック・ファイルの再生を開始します。
[Stop playback]	プレイバックまたは記録を停止し、関連するファイルを閉じて、関連するプレイバックまたは記録のコンテキストを終了します。
[Start Recording]	記録されたデータを保存するファイルの選択をユーザに求めるプロンプトを表示し、Event Stream Processor レコーダを開始します。
[At timestamp rate]	このスライダは、プレイバック中にプレイバックの速度を変更する場合に使用します。

表 10 : プレイバック・モードのオプション

機能	説明
[Full rate]	[Full rate] には、プレイバックの速度が ESP スタジオで指定されないことを示します。[Full rate] は、ESP スタジオが実行されているコンピュータ、ネットワーク の遅延時間などの要因に依存します。
[Timestamp column]	[Timestamp column] オプションは、指定されたカラムからのタイミング・レート情報を使用して記録ファイルをプレイバックすることを指定します。 これを使用するには、[Timestamp column] に入力します。 プレイバック中に、タイムスタンプによってレコード間の時間間隔が決定されます。 [Use Recorded Time] チェックボックスをオンにした場合、プレイバック・ファイルは、データが記録された時刻を使用して実行されます。 それ以外の場合、プレイバック・ファイルは現在の時刻を使用し、現在運用されているものとして再生されます。

機能	説明
[Rec/ms]	レコード数／ミリ秒 ([rec/ms]) モードでは、レコード数／ミリ秒の速度でプレイバックが実行されます。 このオプションでは、最初に rec/ms 速度を設定してから、後で [At timestamp rate] スライダ・ツールを使用してその速度を変更できます。

ESP スタジオ・レコーダでサポートされるファイル形式は、次のとおりです。

- `.xml`(ESP XML)
- `.csv` (カンマ区切り値)
- `.bin` (ESP バイナリ)
- `.rec` (ESP スタジオ記録ファイル)

Event Stream Processor は `.rec` 形式で記録して、受信データの元のタイミングを保持します。

注意： バイナリ・メッセージはアーキテクチャに依存します。ビッグ・エンディアン・マシンで作成されたバイナリ・メッセージはリトル・エンディアン・マシンで実行されている ESP サーバにロードできません。その逆も同様です。

プレイバック・ファイルへの受信データの記録

Event Stream Processor に流れているデータからプレイバック・ファイルにデータを記録します。これで、データを保存して、後で表示できます。

前提条件
ESP サーバに接続しておきます。また、ストリームとウィンドウを Stream Viewer に表示しておきます。

手順

[Playback] ビューで、次の作業を行います。

1. [Select Project] ( ) をクリックします。
2. 記録するプロジェクトを選択します。
3. [OK] をクリックします。
4. [Record] アイコンをクリックします。
5. 記録するストリームとウィンドウを選択するか、[Select All] をクリックしてプロジェクト内のすべてのストリームとウィンドウを記録します。
6. [OK] をクリックします。
7. 記録の保存先のファイルを選択するか作成します。

8. [OK] をクリックします。
9. 次のどちらかを使用して、選択したストリームにデータを送信します。
   - [Manual Input] ビュー。データを入力してストリームにパブリッシュします。
     または、
   - [File Upload]。既存のデータ・ファイルを取得してストリームにパブリッシュします。
10. [Stop] をクリックして記録を停止します。

参照：
- *記録データの再生* (104 ページ)

記録データの再生

実行中の Event Stream Processor インスタンスで、以前記録したデータを表示して再生します。

注意： プレイバック・ビュー・オプションの選択は、プレイバック対象のファイルを選択する前でも後でもかまいません。

1. [Playback File] ( ) をクリックします。
2. プレイバック対象のファイルを見つけて選択します。
   プレイバック・ファイルが [Playback File History] に追加されます。履歴に登録されているファイルもプレイバックできます。履歴項目をダブルクリックし、プレイバック対象としてアクティブ化します。

   注意： [Remove] ボタンまたは [Delete] キーのどちらかを使用して、履歴から項目を削除できます。 プレイバック履歴の変更は永続的です。
3. [Play] をクリックしてプレイバックを開始します。
   Stream Viewer に、デフォルトでは記録されたときの速度でデータが表示されます。

参照：
- *プレイバック・ファイルへの受信データの記録* (103 ページ)

デバッグ

[Run-Test] パースペクティブには、データ・フローをデバッグするツールと、プロジェクト内のバグの検出と修正に役立つツールの 2 つが用意されています。
Debugger では、ブレークポイントを設定できます。Event Tracer では、各受信イベントがプロジェクトのすべてのストリームとウィンドウに与える影響が示されます。

デバッグ・ツールはプロジェクトの開発時に使用するものであって、Event Stream Processor が運用モードの場合に使用するものではありません。通常、デバッグ機能は無効です。デバッグ機能を使用するには、事前にシステムをトレース・モードにしておきます。

スタジオには、プロジェクトをデバッグするための幅広いツール・セットが用意されていますが、コマンド・ラインからもデバッグできます。『ユーティリティ・ガイド』を参照してください。

[Event Tracer] ビュー

Event Tracer はデータ・フローのデバッグに使用するツールの 1 つです。イベントがプロジェクトの各ストリームとウィンドウに与える影響が示されます。

[Event Tracer] ビューには、モデル内のトランザクション・フローが表示され、それぞれのノード (ストリームまたはウィンドウ) でデータを確認できます。[Event Tracer] ビューに示される複数のノードから 1 つのデータ・フローが描画され、ノード間の関係が示されます。

表 11 : [Event Tracer] ビュー

ボタン	機能
Select Running Project	スタジオからモニタできる実行中のプロジェクトのリストが表示されます。
Layout TopDown	上から下へのデータ・フローに合わせて図形を縦方向に並べ替えます。
Layout Left to Right	左から右へのデータ・フローに合わせて図形を横方向に並べ替えます。
Save	イメージを JPG ファイルとして保存します。
Zoom In	イメージのサイズを拡大します。
Zoom Out	イメージのサイズを縮小します。
Zoom Page	イメージのサイズを元のサイズに戻します。
Print Performance Data to Console	収集したデータをコンソールに出力します。
Close Subscription	サブスクリプションを終了してビューをクリアします。
Show Current Subscription in New View	別のビューに現在のサブスクリプションを表示します。
Fit Shape Ids	図形を展開してストリームまたはウィンドウの名前を表示します。

ボタン	機能
Initialize With Base Data	Event Stream Processor から Event Tracer にすべてのイベント・データを送信します。

参照：
- *ブレークポイントとウォッチ変数によるデバッグ*(107 ページ)

Event Tracer でのデータ・フローのトレース

[Event Tracer] タブまたは [Server] ビューから Event Tracer を実行します。

前提条件

ESP サーバが実行されていることを確認します。

手順

1. [Run-Test] パースペクティブで、次のどちらかの手順に従います。

方法	手順
Event Tracer	1. [Event Tracer] ビューをクリックします。 2. [Select Running Project] ( ) をクリックして、ストリームまたはウィンドウがある実行中のプロジェクトを表示します。 3. Event Tracer の対象である実行中のプロジェクトを選択します。 4. [OK] をクリックします。
[Server] ビュー	1. [Server] ビューを選択します。 2. [Server] ビューを更新するには、[Reconnect All] をクリックします。 3. ストリームがある実行中のプロジェクトを選択します。 4. プロジェクト・ノードを右クリックし、[Show in] > [Event Tracer View] を選択します。

   ビューアに示される複数のノードから 1 つのデータ・フローが描画されます。各ノードでトランザクションが処理されると、トランザクションのタイプを反映するようにノードの色が変化します。
2. ノードをダブルクリックすると、対応するストリームのデータが [Console] ビューに表示されます。
3. テスト・データをロードして、[Event Tracer] タブで各ストリームに与える影響を確認するには、次のどちらかを行います。

- [Server] ビューで、[Upload File] を選択してファイルからデータをアップロードします。または、
- [Manual Input] ビューで、[Select Stream] アイコンをクリックしてトランザクションを個別に手動で入力します。ストリームを選択します。確認するため、[OK] をクリックします。

[Event Tracer] ビューの図形の色が変化します。

トポロジ・ストリームの表示

トポロジ・ストリームからデータフロー図が作成されます。この図では、プロジェクトのノード間の関係が直線のセグメントとして表されます。

1. [Run-Test] パースペクティブで [Event Tracer] ビューを選択します。
2. [Select Running Project] をクリックします。希望するプロジェクトを選択して、[OK] をクリックします。
3. 図全体を表示するには、[Layout top down] または [Layout left to right] を選択します。
4. 特定のノードを表示するには、希望するストリームがあるデータフロー図のセクションを選択します。

ブレークポイントとウォッチ変数によるデバッグ

ESP スタジオでは、Event Stream Processor ストリームを通るデータ・フローのトレース、一時停止、再開、ステップを有効にして実行中のプラットフォーム・インスタンスを制御できます。 また、実行中のアプリケーションにブレークポイントとウォッチ変数を作成することもできます。

ブレークポイントとは、Event Stream Processor モデルでデータのフローを停止するストリームまたはウィンドウの入力または出力内の場所です。ウォッチ変数ではデータを検査します。

表 12 : スタジオのブレークポイント・ボタン

ボタン	機能
Trace On	Event Stream Processor にトレース (デバッグ) の開始を指示します。Event Stream Processor のブレークポイント API を使用するには、このパラメータを設定します。
Trace Off	トレース (デバッグ) を停止します。
Step Platform	実行中の Event Stream Processor を停止します。
Pause Platform	実行中の Event Stream Processor を一時停止します。

ボタン	機能
Enable All Breakpoints	リスト内のすべてのブレークポイントを有効にします。
Disable All Breakpoints	リスト内のすべてのブレークポイントを無効にします。
Insert Breakpoint	[Watch] 表にブレークポイント項目を挿入します。
Insert Watch	[Watch] 表にウォッチ項目を挿入します。
Print Breakpoint Data to Console	現在の Event Stream Processor のブレークポイントと一時停止状態のデータをコンソールに出力します。

次のブレークポイント・コマンドで開始される操作の実行は長時間かかります。これらの各コマンドは、[Cancel Current Step] をクリックして、完了前にキャンセルできます。

表 13 : ブレークポイント・コマンド

ボタン	機能
Step Quiesce from Base	入力キューが空になるまで、すべての派生 (非基準) ストリームを自動的にステップします。
Step Quiesce	すべてクワイス状態になるまで、ストリームとその直接と間接の下位ストリームすべてを自動的にステップします。
Step Transaction	トランザクションの最後まで自動的にステップします。
Step Quiesce Downstream	ストリーム自体ではなく、ストリームの下位ストリームをステップします。

注意： ブレークポイントとウォッチ変数はワークスペースに永続的に保持されます。

参照：

- *[Event Tracer] ビュー* (105 ページ)

ブレークポイント

プロジェクト内のどのストリームのブレークポイントでも挿入できます。

ブレークポイントには、次のようなタイプがあります。

- **ローカル –** ストリームへの入力でブレークします。
- **入力 –** ストリームへの特定の入力ストリームでブレークします (複数の入力ストリームを設定できるのは、フレックス、ジョイン、ユニオンのみです)。

- **出力** – データがストリームからの出力の場合にブレークします。

ブレークポイントはカウンタ (enableEvery) に関連付けることができます。カウンタ (n) がブレークポイントに関連付けられている場合、ブレークポイントは、イベント・フローがそのブレークポイントを通った後でトリガします。 その後、カウンタはゼロにリセットされます。

参照：
- *ブレークポイントの追加* (109 ページ)
- *ウォッチ変数* (110 ページ)
- *ウォッチ変数の追加* (111 ページ)
- *Event Stream Processor の一時停止* (112 ページ)
- *Event Stream Processor のステップ* (112 ページ)

ブレークポイントの追加

ブレークポイントを Event Stream Processor に追加します。

前提条件

ブレークポイントを追加するには、[Run-Test] パースペクティブの [Debugger] ビューを表示します。また、プラットフォームをトレース・モードにします。

手順

1. [Trace On] をクリックします。
2. [Insert Breakpoint] ( ) をクリックします。
3. ブレークポイントの設定先のストリームを選択します。
4. ストリームのタイプを選択します。
5. [enableEvery] フィールドに値を入力して、ブレークポイントをトリガする条件を指定します。
6. [Add] をクリックします。

   選択したストリームが [Insert Breakpoint] ダイアログ・ボックス内の表に表示されます。
7. [OK] をクリックします。

   ブレークポイントが [Debugger] ビューの [Breakpoint] の表に表示されます。
8. 特定のブレークポイントを有効にする、無効にする、または削除するには、ブレークポイントを右クリックして、次のオプションを選択します。
   - [Enable Breakpoint]

- [Disable Breakpoint]
- [Remove Breakpoint]

9. すべてのブレークポイントを有効または無効にするには、[Enable All Breakpoints] または [Disable All Breakpoints] を選択します。
10. すべてのブレークポイントを削除するには、[Breakpoints] の表内を右クリックして、[Remove All Breakpoints] を選択します。
11. [Trace Off] をクリックして Event Stream Processor を実行します。

参照：
- *ブレークポイント* (108 ページ)
- *ウォッチ変数* (110 ページ)
- *ウォッチ変数の追加* (111 ページ)
- *Event Stream Processor の一時停止* (112 ページ)
- *Event Stream Processor のステップ* (112 ページ)

ウォッチ変数

[Debugger] の [Breakpoints] ビューで [Watch] 表にウォッチ変数を挿入して、プロジェクトを流れているときのデータを検査します。

ウォッチは次のものに対応します。

- ストリームの現在の入力
- ストリームの現在の出力
- ストリームのキュー
- ストリームのトランザクション入力
- ストリームのトランザクション出力
- ストリームの出力履歴
- ストリームの入力履歴
- フレックス・ストリームの変数

モニタするウォッチを [Watch] 表に追加してから、Event Stream Processor を実行します。Event Stream Processor が実行されているときに、実行制御イベント (実行、ステップ、一時停止) が Event Stream Processor に送信されると [Watch] 表は動的に更新されます。

参照：
- *ブレークポイント* (108 ページ)
- *ブレークポイントの追加* (109 ページ)
- *ウォッチ変数の追加* (111 ページ)
- *Event Stream Processor の一時停止* (112 ページ)

- *Event Stream Processor のステップ*(112 ページ)

ウォッチ変数の追加

ブレークポイントにウォッチ要素を追加します。

前提条件

ウォッチを追加するには、[Run-Test] パースペクティブの [Debugger] ビューを開きます。プラットフォームをトレース・モードにします。

手順

1. [Trace On] をクリックします。
2. [Watch] 表内を右クリックします。
3. [Add Watch] を選択します。
4. [Watch Choices] ボックスからストリームを選択します。
5. そのストリームに設定するウォッチのタイプを選択します。
6. [Add] をクリックします。
   ウォッチがダイアログ・ボックスの下部にある表に表示されます。
7. [OK] をクリックします。
   ウォッチが [Debugger] ビューの [Watch] 表に表示されます。
8. ウォッチを削除するには、[Watch] 表内を右クリックして、次のどちらかを選択します。
   - [Remove Watch]。1 つの選択したウォッチ変数を削除します。または、
   - [Remove All Watches]。すべてのウォッチ変数を削除します。

参照：

- *ブレークポイント* (108 ページ)
- *ブレークポイントの追加* (109 ページ)
- *ウォッチ変数* (110 ページ)
- *Event Stream Processor の一時停止* (112 ページ)
- *Event Stream Processor のステップ*(112 ページ)

Event Stream Processor の一時停止

Event Stream Processor を一時停止します。

前提条件

Event Stream Processor を一時停止するには、[Run-Test] パースペクティブを開きます。

注意： Event Stream Processor をトレース・モードにします。

手順

1. [Debugger] で、[Pause Platform] ( ) をクリックします。
2. Event Stream Processor を再開するには、[Trace Off] をクリックした後で [Trace On] をクリックします。

参照：

- *ブレークポイント* (108 ページ)
- *ブレークポイントの追加* (109 ページ)
- *ウォッチ変数* (110 ページ)
- *ウォッチ変数の追加* (111 ページ)
- *Event Stream Processor のステップ* (112 ページ)

Event Stream Processor のステップ

Event Stream Processor をシングルステップします。

前提条件

シングルステップを実行するには、[Run-Test] パースペクティブの [Debugger] ビューを開きます。ステップを実行するには、プラットフォームも一時停止します。

手順

1. [Debugger] ビューで、[Step Platform] ( ) をクリックして、プラットフォームをシングルステップに移行します。
2. [Cancel Current Step] をクリックしてアクションを終了します。

参照：

- *ブレークポイント* (108 ページ)
- *ブレークポイントの追加* (109 ページ)

- *ウォッチ変数* (110 ページ)
- *ウォッチ変数の追加* (111 ページ)
- *Event Stream Processor の一時停止* (112 ページ)

第 7 章　スタジオ作業環境のカスタマイズ

希望する方法で作業できるようにスタジオ・インタフェースをカスタマイズします。

注意： Eclipse ベースのアプリケーションとして、ESP スタジオには Sybase Event Stream Processor に固有ではない多くの機能が自動的に組み込まれます。このマニュアルで説明されている機能はスタジオでテスト済みです。その他の Eclipse 機能は予期したとおりに動作しない可能性があります。

たとえば、チーム同期化のパースペクティブはサポートされていません。

スタジオ環境設定の編集

環境設定を編集してスタジオの環境をカスタマイズします。

スタジオの関連ビューからも、これらの環境設定の多くにアクセスできます。

1. [Edit] > [Preferences] を選択します。
2. [Sybase Event Stream Processor Studio] を展開したら、設定する環境設定が表示されるまで展開します。環境設定はすべてオプションです。
   - [CCL Editor Settings] – 構文カラー・オプションとテンプレート・オプションの設定。
   - [Run Test Preferences] – サーバ接続のデフォルトの設定、新規接続の追加、[StreamViewer] ビューと [Server] ビューの制限とフィルタの設定、スタジオでプロジェクトを実行するためのその他のオプションの設定。
   - [Compiler Options] – CCL コンパイラ出力のディレクトリの変更 (デフォルトは *workspace¥project* フォルダ内の `bin` フォルダ)。
   - [Data Input Settings] – ファイル・アップロードと [SQL Query] ビューのオプションの設定。
   - [Manual Input Settings] – money 型を除くすべてのデータ型のデフォルトを含む、[Manual Input] ビューからのパブリッシュ・データの設定の選択。
   - [Manual Input Settings - Money Types] – money(n) データ型のデフォルトの設定。
   - **[Shapes General] –** 図で図形を作成して表示する場合のデフォルトの選択。
3. 各環境設定ダイアログで、次のどちらかを行います。
   - [Apply] をクリックして新しい設定を保存します。または、

- [Restore Defaults] をクリックして行った変更を元に戻します。

現在のダイアログ内の設定のみが適用または元に戻されます。

4. [OK] をクリックして [Preferences] ダイアログを閉じます。

参照：

- *図の表示変更* (29 ページ)

手動入力の設定

[Manual Input] ビューからストリームにパブリッシュするデータのデータ型のデフォルト値と、データをパブリッシュする場合の形式を設定します。

ほとんどのデータ型の設定は [Manual Input Settings] 環境設定にあります。money(n) データ型の設定は [Manual Input Settings - Money Types] 環境設定にあります。

設定	説明
[Publish Multiple Rows]	入力ストリームからのデータをシングル・インスタンスでパブリッシュするか、複数のローとしてパブリッシュするかを指定します。
[Use Current Date]	データを現在の日付でパブリッシュするか、過去の日付を維持するかを指定します。
[binary]	ストリームにパブリッシュするバイナリ値を指定します。この設定を使用して、トレース可能な値をフィールドに挿入して、ストリームのバイナリ値をモニタします。
[boolean]	[True] または [False] に設定できます。
[string]	スタジオが文字列型で受け入れるデフォルト値を指定します。
[integer]	スタジオが整数型で受け入れるデフォルト値を指定します。小数点を使用した値は受け入れられません。
[float]	スタジオが浮動小数点型で受け入れるデフォルト値を指定します。
[long]	スタジオが long 型で受け入れるデフォルト値を指定します。
[interval]	スタジオが interval 型で受け入れるデフォルト値を指定します。
[date]	日付型のデフォルト値を指定します。[Select] を選択してカレンダ・ダイアログを開き、ミリ秒の精度でデフォルトの日付スタンプを選択します。

設定	説明
[bigdatetime]	bigdatetime 型のデフォルト値を指定します。[Select] を選択してカレンダ・ダイアログを開き、ミリ秒の精度でデフォルトの bigdatetime スタンプを選択します。
[timestamp]	timestamp 型のデフォルト値を指定します。[Select] を選択してカレンダ・ダイアログを開き、ミリ秒の精度でデフォルトのタイムスタンプを選択します。
[money(n)]	さまざまな精度の money 型のデフォルト値を指定します。ここで、n は小数点以下の最大桁数です。money 型のデフォルト値は、小数点以下最大 15 桁で設定します。

注意： 間違った文字を入力したり、フィールドに入力できる文字数を超えたりすると、環境設定ウィンドウの上部にエラー・メッセージが表示されます。

参照：
- *ストリームへのデータの手動入力* (99 ページ)

パースペクティブでのビューの並べ替え

ビューをパースペクティブ内の新しい結合場所に移動して、パースペクティブでビューを並べ替えます。

1. 移動するビューのタイトル・バーをクリックします。
2. 左マウス・ボタンを押したまま、ビューを新しい領域にドラッグします。

   ビューを移動すると、ドロップ・カーソル・アイコンの外観が変わるので、ビューを結合できる場所を判断できます。

表 14 : ドロップ・カーソル

ドロップ・カーソル	カーソル名	説明
	Dock Above	カーソル下にあるビューの上に結合します。
	Dock Below	カーソル下にあるビューの下に結合します。
	Dock to the Right	カーソル下にあるビューの右に結合します。

ドロップ・カーソル	カーソル名	説明
	Dock to the Left	カーソル下にあるビューの左に結合します。
	Stack	ビューはカーソル下にあるビューのタブとして表示されます。
	Restricted	ビューは結合できません。たとえば、エディタではビューを結合できません。

3. ビューを適切な位置に置いたら、左マウス・ボタンを離し、ビューを新しい場所にドロップします。

   アプリケーションを閉じると、新しい設定が保存されます。

パースペクティブ・ショートカット・バーの移動

パースペクティブ・ショートカット・バーは、デフォルトではパースペクティブの左上隅に横方向に表示されます。

パースペクティブ・ショートカット・バーは、パースペクティブの右上に横方向に結合するか、左側に縦方向に結合できます。

1. パースペクティブ・ショートカット・バーを右クリックすると、コンテキスト・メニューが開きます。
2. 次のいずれかを実行します。

選択	ショートカット・バーの結合先
[Dock on] > [Top Right]	右上で、メイン・ツールバーの隣に横方向に結合。
[Dock on] > [Top Left]	左上で、メイン・ツールバーの下に縦方向に結合。これがデフォルトです。
[Dock on] > [Left]	右上で、パースペクティブの側面に縦方向に結合。

付録 A スキーマ検出をサポートするアダプタ

スキーマ検出をサポートするアダプタと、スキーマ検出を有効にする場合に使用するプロパティについて説明します。

アダプタ	スキーマ検出のサポート	プロパティ
AtomReader インプット	不可	—
データベース・インプット	可	Database Service アダプタのデータベース接続の取得先であるデータベース・サービスの名前。
データベース・アウトプット	可	Database Service 使用するサービス・エントリの名前。
ファイル CSV インプット	可	Directory アダプタが読み取るデータ・ファイルの絶対パス。
ファイル CSV アウトプット	不可	—
FIX ファイル・インプット	不可	—
FIX ファイル・アウトプット	不可	—
XML ファイル・インプット	可	Directory アダプタが読み取るデータ・ファイルの絶対パス。
XML ファイル・アウトプット	不可	—
FIX インプット	不可	—
FIX アウトプット	不可	—
フレックス・アウトプット	不可	—

アダプタ	スキーマ検出のサポート	プロパティ
HTTP インプット	不可	—
JMS CSV インプット	可	• Delimiter - フィールド・デリミタ。 • Connection Factory - 接続ファクトリ・クラス名。 • JNDI Context Factory - JNDI コンテキスト初期化用のコンテキスト・ファクトリ。 • JNDI URL • Destination Type • Destination Name
JMS CSV アウトプット	不可	—
JMS カスタム・インプット	不可	—
JMS カスタム・アウトプット	不可	—
JMS FIX インプット	不可	—
JMS FIX アウトプット	不可	—
JMS Object Array インプット	可	• Connection Factory - 接続ファクトリ・クラス名。 • JNDI Context Factory - JNDI コンテキスト初期化用のコンテキスト・ファクトリ。 • JNDI URL • Destination Type • Destination Name
JMS Object Array アウトプット	不可	—

アダプタ	スキーマ検出のサポート	プロパティ
JMS XML インプット	可	• Connection Factory - 接続ファクトリ・クラス名。 • JNDI Context Factory - JNDI コンテキスト初期化用のコンテキスト・ファクトリ。 • JNDI URL • Destination Type • Destination Name
JMS XML アウトプット	不可	—
KDB インプット	可	• KDB Server • KDB Port • KDB User • KDB Password
KDB アウトプット	可	• KDB Server • KDB Port • KDB User • KDB Password
ログ・ファイル・インプット	不可	—
ランダム・タプル生成インプット	不可	—
ロイター・マーケットフィード・インプット	可	Discovery Path
ロイター・マーケットフィード・アウトプット	不可	—
ロイター OMM インプット	可	Discovery Path
ロイター OMM アウトプット	不可	—
RTView アウトプット	不可	—
SMTP アウトプット	不可	—
ソケット (クライアント側) CSV インプット	不可	—

アダプタ	スキーマ検出のサポート	プロパティ
ソケット (クライアント側) CSV アウトプット	不可	—
ソケット (クライアント側) XML インプット	不可	—
ソケット (クライアント側) XML アウトプット	不可	—
ソケット (サーバ側) XML インプット	不可	—
ソケット (サーバ側) XML アウトプット	不可	—
ソケット (サーバ側) CSV インプット	不可	—
ソケット (サーバ側) CSV アウトプット	不可	—
ソケット FIX インプット	不可	—
ソケット FIX アウトプット	不可	—
Sybase IQ アウトプット	不可	—
オープン・インプット／アウトプット	不可	—
Tibco Rendezvous インプット	不可	—
Tibco Rendezvous アウトプット	不可	—
NYSE インプット	可	Discovery Directory Path アダプタ検出ディレクトリへの絶対パス。

参照：

- *スキーマ検出* (31 ページ)
- *スキーマの検出* (32 ページ)

索引

記号

A

C

E

F

G

う

え

か

き

く

さ

し

す

ほ

ま

め

も

ゆ

り

れ

ろ

わ