SAS® Event Stream Processing: SAS® Event Stream Processing Studio の使用

2026.07

このドキュメントは、ソフトウェアの追加バージョンに適用される場合があります。このドキュメントを SAS Help Center で開き、バナーのバージョンをクリックすると、使用できるすべてのバージョンが表示されます。

SAS Event Stream Processing Studio の概要

SAS Event Stream Processing Studio とは

SAS Event Stream Processing Studio は、Web ベースのクライアントで、SAS Event Stream Processing Studio Modeler を使用して、イベントストリーム処理モデルの作成、編集、インポート、パブリッシュ、テストを行うことができます。SAS Event Stream Processing Studio Modeler は、モデルをデータフローダイアグラムとして表示します。これにより、ウィンドウがどのように関連して互いに流れ込むかを表示および制御することができます。

SAS Event Stream Processing Studio へのアクセス

他の SAS Event Stream Processing クライアントまたは他のアプリケーションを SAS Viya プラットフォームを使用している場合、  をクリックし、**ストリーミングプロジェクトの設計**を選択すると SAS Event Stream Processing Studio にアクセスできます。アプリケーションメニューにあるアプリケーションリストは、システムがどのようにインストールされたかと、アクセス許可に依存します。

それ以外の場合は、SAS Event Stream Processing Studio にアクセスしてください。

1 次の URL を開きます。

https://*name-of-ingress-host*/SASEventStreamProcessingStudio

変数 *nam-of-ingress-host* は、Kubernetes クラスター Ingress 用に設定されているホスト名を指定します。

2 ユーザー ID とパスワードを入力し、**サインイン**をクリックします。

注: スタンドアロン版の SAS Event Stream Processing が配置されている場合、ユーザーにサインインを要求するようにシステムが構成されていない可能性があります。この場合、サインインするためのページは表示されません。このステップをスキップします。

サポートされている Web ブラウザの情報と SAS Event Stream Processing Studio へのアクセスに使用するデバイスが要件を満たしているかどうかについては、“Client Requirements” (*System Requirements for the SAS Viya Platform*)を参照してください。

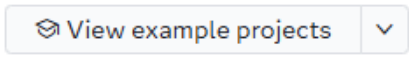
SAS Event Stream Processing Studio では、セッション状態を維持するために Cookie を使用します。

ようこそページとその他のオンボーディングリソースの使用

SAS Event Stream Processing Studio または SAS Event Stream Manager に初めてアクセスすると、**SAS Event Stream Processing にようこそ**ページが表示されます。**SAS Event Stream Processing にようこそ**ページのリソースを使用して、SAS Event Stream Processing および Web ベースのクライアントについて学習してください。

最初に **SAS Event Stream Processing にようこそ**ページから SAS Event Stream Processing Studio に移動する場合、**SAS Event Stream Processing Studio へようこそ**ウィンドウが表示されます。このウィンドウのリンクを使用して、さらなるオンボーディングリソースにアクセスします。

- サンプルプロジェクトを含むユーザーインターフェイスのツアー
- さらなるプロジェクト例

プロジェクトページで、 

をクリックして、ツアーとサンプルにアクセスすることもできます。

アプリケーションバーのユーザーメニューから、各リリースの新機能に関する情報など、他のリソースにアクセスできます。詳細については、“ツールバー”を参照してください。

SAS Event Stream Processing Studio を Progressive Web App としてインストール

SAS Event Stream Processing Studio は、Progressive Web App(PWA)としてインストールできます。詳細については、“Progressive Web App としての SAS Event Stream ProcessingWeb ベースクライアントの使用” (*SAS Event Stream Processing: 概要*)を参照してください。

PWA をインストールする機能は、SAS Event Stream Processing Studio 内からは利用できません。しかし、他のウェブアプリから PWA をインストールし、PWA 内から SAS Event Stream Processing Studio にアクセスすることができます。

PWA をインストールするには、次のとおりにします。

1 Chromium ベースの Web ブラウザ(Chrome など)で、他の SAS Web アプリを開きます。

2 次のいずれかの操作を行います。

- アドレスバーの末尾にある適切なアイコンをクリックして、 **SAS のインストール**を選択します。
- Web ブラウザの**詳細**メニューを開き、**SAS のインストール**を選択します。

> **注:** **SAS のインストール** オプションが他のメニューオプションの下にある場合があります。例えば、Microsoft Edge ブラウザでは、**アプリ**の下にあります。

Web アプリが SAS というデスクトッププログラムとしてインストールされました。一つのウェブアプリケーションが PWA としてインストールされた後、他のすべての SAS アプリケーションが PWA でアクセスできるようになります。

PWA をアンインストールするには、次のとおりにします。

1 PWA を開きます。

2 PWA メニューで **SAS のアンインストール**を選択します。

ユーザーインターフェイスについて

ページ

ページは、ユーザーインターフェイスの最上位コンテナーです。他のすべてのユーザーインターフェイスエレメントは、ページ内に含まれています。

SAS Event Stream Processing Studio には、次のメインページが含まれています。

- モデルを含むプロジェクトの作成、編集、インポート、エクスポート、または削除を行うことができる**プロジェクト**ページ。
- ESP サーバーの追加、編集、インポート、エクスポート、または削除を行うことができる **ESP サーバー**ページ。
- **カスタムウィンドウ**ページでは、カスタムウィンドウを作成および管理できます。

また、SAS Event Stream Processing Studio に初めてアクセスすると、**ようこそ**ページが表示されます。詳細については、“ようこそページとその他のオンボーディングリソースの使用”を参照してください。

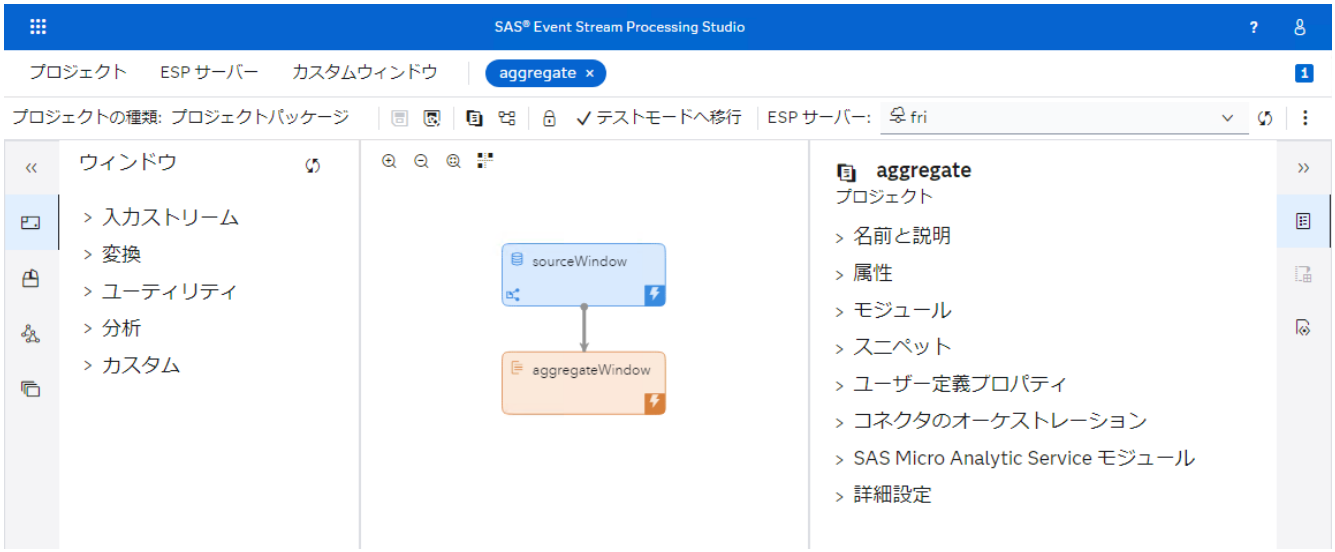
図 1　プロジェクトページ

プロジェクトページで実行できるタスクの詳細については、“プロジェクトの操作の概要”を参照してください。

ペイン

同じページ内でさまざまなタイプの情報を表示できる *ペイン*。次の図では、SAS Event Stream Processing Studio Modeler の開いたプロジェクトが表示されます。左側の**ウィンドウ**ペインには、プロジェクトに追加できるウィンドウが表示されます。真ん中にはワークスペースがあります。右側のペインには、プロジェクトで現在選択されているアイテムのプロパティが 表示されます。

図 2　2 つの縦ペインの例



SAS Event Stream Processing Modeler の詳細については、“SAS Event Stream Processing Studio Modeler の使用”を参照してください。

横ペインの例については、“タイル”を参照してください。

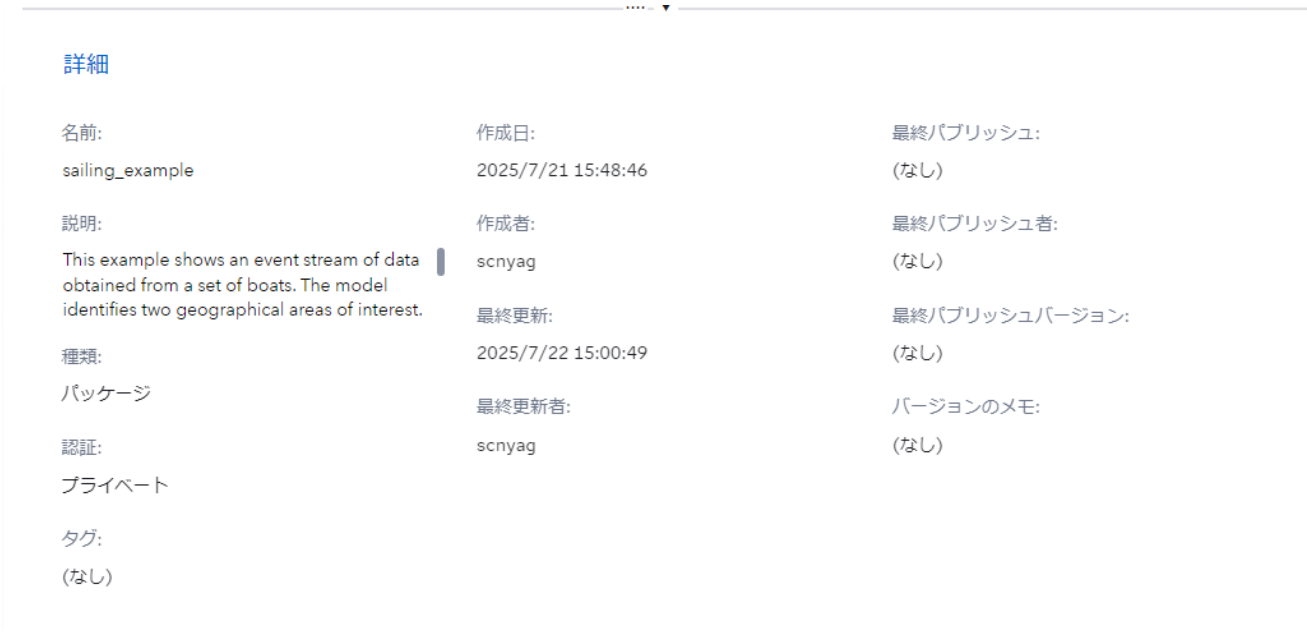
ペインのサイズを変更するには、境界線を適切な方向にドラッグします。横ペインのサイズを変更するには、境界線を上または下にドラッグします。縦ペインのサイズを変更するには、境界線を左右にドラッグします。

プロジェクト、**ESP サーバー**、および**カスタムウィンドウ**ページには、非表示にできる水平ペインが含まれています。ペインを非表示にするには、ツールバーで をクリックします。または、枠線で ▾ をクリックしてペインを非表示にすることもできます。もう一度ペインを表示するには、 をクリックします。または、ユーザーインターフェイスの下部にある ▴ をクリックします。

タイル

*タイル*は、ペイン内に存在する情報の完結型ブロックであり、時にはページ上に直接存在することもあります。

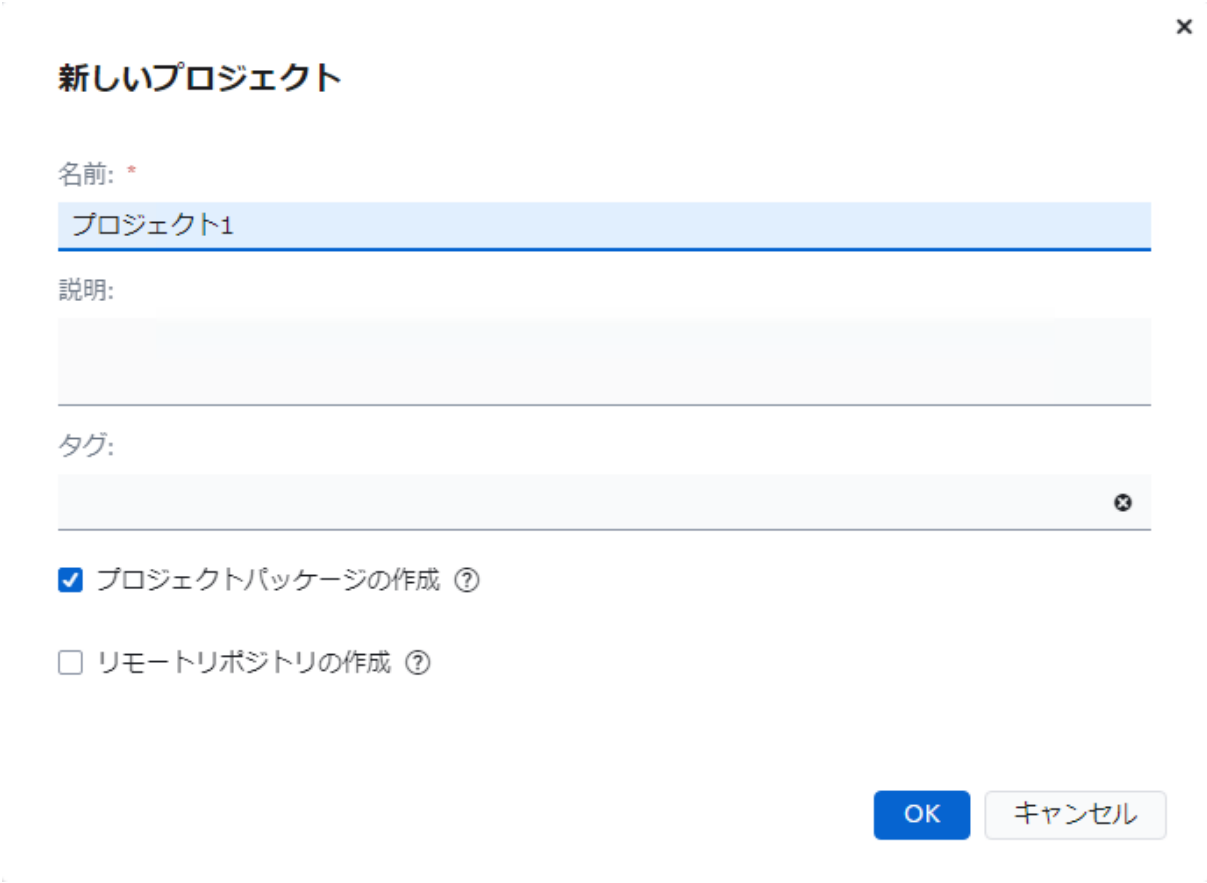
図 3　横ペイン内の単一タイルの例



ウィンドウ

*ウィンドウ*とは、フロー形式でユーザー定義アクションを表示するためのユーザーインターフェイスエレメントです。ウィンドウは、通常、アクションを実行するための手段を提供し、ウィンドウを起動したページに戻るために閉じることができます。次の図は、SAS Event Stream Processing Studio でプロジェクトの新規作成に使用されるウィンドウを示しています。

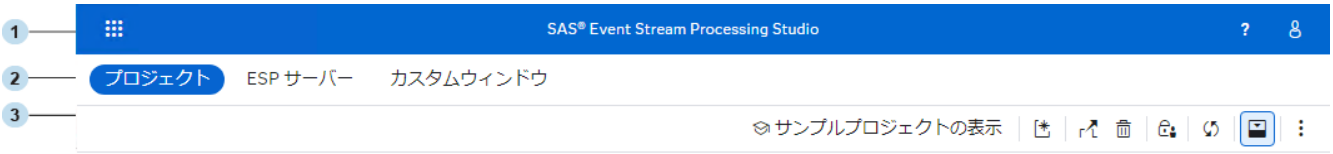
図 4　新しいプロジェクトウィンドウ



注: SAS Event Stream Processing Studio のユーザーインターフェイスエレメント *ウィンドウ*は、SAS Event Stream Processing ウィンドウと同じ意味を持ちません。SAS Event Stream Processing では、ウィンドウは連続クエリのコンポーネントです。連続クエリには、ソースウィンドウと 1 つ以上の派生ウィンドウが含まれます。SAS Event Stream Processing ウィンドウは、関連する方向を持つエッジによって接続されます。SAS Event Stream Processing Studio ユーザーインターフェイス要素ウィンドウと SAS Event Stream Processing ウィンドウの両方が含まれます。

ツールバー

図 5　メインツールバー

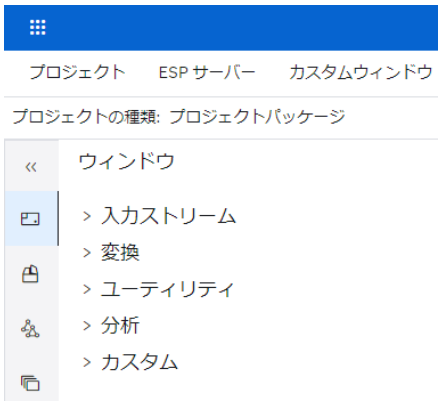


次の表に示すように、SAS Event Stream Processing Studio には 3 つの主要なツールバーがあります。

アイテムの番号	説明	説明
1	アプリケーションバー	■ アプリケーションメニューを表示し、SAS Viya プラットフォーム内の他のアプリケーションにアクセスします。アプリケーションメニューとその中にあるアプリケーションリストの存在は、システムがどのようにインストールされたかと、アクセス許可に依存します。 ■ ヘルプへのアクセスを提供します。 ■ 名前またはユーザー ID の最初の文字を示すユーザーアイコンを 表示します。ユーザーアイコンをクリックすると、次の機能にアクセスできます。 □ 表示名またはユーザー ID を完全に表示します。表示名が設定されていない場合は、デフォルトでユーザー ID が表示されます。ユーザー名とパスワードを使用してサインインする必要がないように SAS Event Stream Processing Studio が設定されている場合、ユーザー ID は表示されません。 □ 各リリースの新機能、設定、および製品情報に関する情報を表示します。 □ SAS Event Stream Processing Studio からのサインアウト(該当する場合)。
2	ナビゲーションバー	■ メインの SAS Event Stream Processing Studio ページへのアクセスを提供します。**プロジェクト**、**ESP サーバー**、および**カスタムウィンドウ**。 ■ 現在開いている各プロジェクト、プロジェクトバージョンまたはモデルテストへのアクセスを提供します。ナビゲーションオーバーフローメニューボタンでは、現在開いているこれらのページの総数が表示されます。たとえば、1 つのページが開いている場合、次のアイコンが表示されます。 1 .
3	ツールバー	■ 開いているアイテムに 関連付けられたボタンまたはタブが含まれます ■ プロジェクトに関連付けられたアクションを実行できます。たとえば、ツールバーの⋮をクリックして**プロジェクトのインポート**を選択すると、プロジェクトをインポートできます。

水平ツールバーに加えて、SAS Event Stream Processing Studio 垂直ツールバーが含まれています。次の図は、垂直ツールバーの例を示しています。この例のツールバーは、プロジェクトを開いたときに表示されます。このツールバーを使用すると、**ウィンドウ**ペイン、**プロジェクトパッケージ**ペイン、および**プロジェクトバージョン**ペインを切り替えることができます。

図 6　垂直ツールバーの例



並べ替えとフィルタリング

大量の情報を扱いやすくするために、表に表示されている項目をソートしてフィルタ処理することができます。また、列を表示、非表示、並べ替えることもできます。

アイテムのリストを昇順または降順で並べ替えることができます。昇順でソートするには、並べ替える列の見出しをクリックします。降順でソートするには、列をもう一度クリックします。並べ替えを削除するには、列を 3 回クリックします。

列の情報のサブセットのみを表示するフィルタ条件を作成できます。フィルタ条件を作成するには、フィルタ条件を適用する列の⋮をクリックし、**フィルタ**を選択してフィルタ条件を入力します。

表示する列を構成できます。これを行うには、任意の列の⋮をクリックし、列を選択し、表示しない**列**の選択を解除します。

列を並べ替えることができます。これを行うには、列見出しをクリックしたままにして別の場所にドラッグします。

SAS Event Stream Processing Studio モデラー

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開くと、プロジェクトの内容を含む別のページが表示されます。このプロジェクトページには、SAS Event Stream Processing Studio モデラーが表示され、モデルを視覚的にデザインしてテストすることができます。SAS Event Stream Processing Studio モデラーには XML エディタも含まれています。このエディタを使用してモデルを構築することもできます。

モデラーの詳細については、“SAS Event Stream Processing Studio Modeler の使用”を参照してください。XML エディタの詳細については、“XML エディタの使い方”を参照してください。

プロジェクトの操作

プロジェクトの操作の概要

*プロジェクト*は、1 つ以上の連続クエリで構成されます。SAS Event Stream Processing Studio を使用して、プロジェクトの作成、インポート、エクスポート、削除ができます。**プロジェクト**ページでは、配置内のプロジェクトを、それらの識別の詳細と一緒に表示できます。

次の図にその例を示します。

図 7　プロジェクトページ

注: SAS Event Stream Processing Studio で編集中のプロジェクトは自動的にロックされます。これにより、プロジェクトの変更を他のユーザーが誤って上書きすることがないようにします。詳細については、“プロジェクトのロック”を参照してください。

プロジェクトページのテーブルには、各プロジェクトに関する次の情報が表示されます。

- プロジェクトのアクセスレベル。この列は、SAS Event Stream Processing が他の SAS 製品とともに配置されている場合に表示されます。
- プロジェクトの名前。
- プロジェクトの種類**パッケージ**は、プロジェクトパッケージが存在することを示します。**XML ファイル**は、プロジェクト XML ファイルがプロジェクトパッケージの一部ではないことを意味します。
- プロジェクトに割り当てられているタグの識別。
- プロジェクトが最後に更新された日時。
- プロジェクトを最後に更新したユーザーのユーザー名。
- プロジェクトを所有するユーザーのユーザー名。所有者とは、プロジェクトを作成またはインポートしたユーザーです。所有者の変更はできません。プロジェクトのアクセスレベルを変更できるのは、所有者と SASAdministrators グループのメンバーです。
- プロジェクトがリモートリポジトリに保存されているかどうか。この列は、ファイルベースのバージョン管理が使用されている場合に関連します。詳細については、“ファイルベースのバージョニングと Git”を参照してください。

ツールバー上のいくつかのボタンに関する情報は次のとおりです。

- サンプルプロジェクトを表示してインストールするには、サンプルプロジェクトの表 をクリックします。詳細については、“サンプルプロジェクトのインストール”を参照してください。
- プロジェクトを編集用に開くには、メインテーブルからプロジェクトを選択して、をクリックします。または、メインテーブルでプロジェクトをダブルクリックして、プロジェクトを編集用に開くこともできます。
- プロジェクトを作成またはインポートすると、デフォルトでは制限され、プライベートとマークされます。プロジェクトのアクセスレベルを変更するには、メインテーブルからプロジェクトを選択して をクリックして、**プライベートプロジェクト**または**パブリックプロジェクト**を選択します。複数のプロジェクトのアクセスレベルを同時に変更するには、メインテーブルでプロジェクトを選択する際に Ctrl を押したまま選択します。詳細については、“プロジェクトのアクセス制御”を参照してください。
- メインテーブルを更新するには、をクリックします。
- プロジェクトの追加情報を表示するには、**プロジェクト**ページで関連するプロジェクトを選択してください。次に示すように、この情報を含むタイルが表示されます。この情報を隠すには、 をクリックします。

プロジェクトの作成

1 **プロジェクト**ページで、をクリックします。

新しいプロジェクトウィンドウが表示されます。

2 **新しいプロジェクト**ウィンドウで、次のタスクを実行します。

a **名前**フィールドに、プロジェクトの一意の名前を入力します。

注: 一意のプロジェクト名を入力する必要があります。重複したプロジェクト名はサポートされていません。

プロジェクトがパブリッシュされ、その後に削除された場合、削除したプロジェクトの名前を再利用することはできません。

---

b 必要に応じて、**説明**フィールドにプロジェクトの説明を入力します。

c 必要に応じて、**タグ**フィールドにはプロジェクトを表す識別キーワードを入力します。

d **プロジェクトパッケージの作成**チェックボックスを使用して、プロジェクトをプロジェクトパッケージの一部にするかどうかを指定します。詳細については、"プロジェクトパッケージの操作"を参照してください。

e (オプション)このプロジェクトに関連付けられている大きなファイルの保存に Git LFS を使用する場合は、**Large File Storage(LFS)の使用**チェックボックスを選択します。詳細については、"ファイルベースのバージョン管理と Git Large File Storage"を参照してください。

注: Git LFS は、プロジェクトパッケージが次のいずれかの方法で保存されている場合に使用できます。

- SAS Event Stream Processing の内部にある Git リポジトリ
- Gitea のリモートリポジトリ

---

f (オプション)ファイルベースのバージョン管理が使用されている場合は、**リモートリポジトリの作成**チェックボックスを選択して、プロジェクトをリモート Git リポジトリに保存できます。**リモートサーバー接続**ドロップダウンリストを使用して、リポジトリを作成するリモート Git サーバーを選択します。詳細については、"ファイルベースのバージョン管理"を参照してください。

注: **リモート サーバー接続**ドロップダウンリストに関する追加情報は次のとおりです:

- リモートサーバー接続が設定されていないことを示すメッセージが表示された場合は、システム管理者に問い合わせてください。
- Git サーバーのタイプは、接続名の後の括弧内に表示されます。次に例を示します: **MyGitServer(GitLab)** 。種類が **Generic** である Git サーバーを選択した場合、作成しようとしているプロジェクトの名前と一致する名前を持つ空のリポジトリがその Git サーバー上にすでに存在している必要があります。
- 詳細については、"リモート Git サーバーへの接続を指定する"を参照してください。

---

g **OK** をクリックします。

- Kubernetes 環境を利用している場合、ステップ 5 まで進んでください。Kubernetes 環境では、ESP サーバーがオンデマンドで作成され、手動で作成する必要はありません。

- エッジ環境を使用している場合は、ESP サーバーが利用可能で、構成可能な状態になっている必要があります。エッジ環境にある ESP サーバーを開始する方法の詳細については、*"エッジ環境での ESP サーバーの開始" (SAS Event Stream Processing: Edge 環境での ESP サーバーの使用)*を参照してください。エッジ環境で ESP サーバーが現在構成されていない場合は、ESP サーバーをすぐに構成するかどうかを決定するプロンプトが表示されます。

h **はい**をクリックして ESP サーバーをすぐに構成するか、**いいえ**をクリックして後で ESP サーバーを構成します。

はいを選択した場合は、**ESP サーバープロパティの編集**ウィンドウが表示されます。

3 後で ESP サーバーを構成するために**いいえ**を選択した場合は、このステップをスキップしてください。すぐに ESP サーバーを構成するために**はい**を選択した場合は、次のタスクを実行します。

a **名前**フィールドでは、ESP サーバーを識別する名前を入力または更新します。

b 必要に応じて、**説明**フィールドでは、ESP サーバーの説明を入力または更新します。

c **ホスト**フィールドでは、ESP サーバーのホスト名または IP アドレスを入力または更新します。

d **HTTP ポート**フィールドでは、ESP サーバーの管理ポート番号を入力または更新します。

e 必要に応じて、**編集**をクリックして**認証**フィールドの設定を変更します。

認証ウィンドウが表示されます。

- **なし**: これはデフォルトのオプションです。
- **Kerberos**: このオプションは、ESP サーバーが Kerberos を使用した認証を要求するように構成されている場合にのみ関係します。

f 必要に応じて、**SSL を使用して接続**チェックボックスを選択します。このオプションを選択するのは、ESP サーバーが SSL 暗号化を要求するように構成されている場合のみです。

g (オプション)**接続のテスト**をクリックします。接続をテストすることで、無効な ESP サーバーの接続が発生し、無効な接続が更新されたときにタイムアウトすることを防ぎます。

h 必要に応じて、**タグ**フィールドでは、ESP サーバーを説明するキーワードを入力または更新して、Enter キーを押します。

i 必要に応じて、**サーバーログを有効にする**チェックボックスをオンにして、ESP サーバーでのログ記録を有効にします。

j 必要に応じて、**保持するメッセージ数**フィールドで、ESP サーバーログに保持されるデフォルトのメッセージ数を変更します。デフォルトは 10,000 メッセージです。

k **OK** をクリックします。

4 **OK** をクリックします。

5 SAS Event Stream Processing Studio モデラーが表示されます。ワークスペースにはソースウィンドウが含まれています。プロジェクトは、プライベートのプロジェクトとしてデフォルトプロパティのセットで作成されます。

をクリックします。

注: 別のファイル名でプロジェクトのコピーを作成するには、をクリックします。関連情報を**名前を付けて保存**ウィンドウに入力し、**OK** をクリックします。

モデルの作成を開始する前に、デフォルトのプロジェクトプロパティを確認し、必要に応じて更新してください。詳細については、“プロジェクトプロパティの更新”を参照してください。

モデルの作成の詳細については、“SAS Event Stream Processing Studio Modeler の使用”を参照してください。

プロジェクトのインポート

注: SAS Event Stream Processing Studio にインポートされるプロジェクト XML ファイルは、UTF-8 形式でエンコードされていなければなりません。UTF-8 形式でエンコードされていないプロジェクト XML ファイルをインポートすると、無効な文字が SAS Event Stream Processing Studio に表示されることがあります。プロジェクトパッケージをインポートするか、プロジェクトパッケージの一部ではないプロジェクト XML ファイルをインポートするかに関係なく、プロジェクト XML ファイルは UTF-8 形式でエンコードする必要があります。

1 **プロジェクト**ページで、⋮をクリックします。

2 関連するオプションを選択します。環境で使用できるオプションは、どのように SAS Event Stream Processing 配置されたかによって異なります。

- **XML ファイルのインポート**: ローカルファイルシステムから、プロジェクトパッケージの一部ではないプロジェクト XML ファイルをインポートします。
- **プロジェクトのインポート**: ローカルファイルシステムから、プロジェクトパッケージを含む ZIP ファイルをインポートします。
- **リモートサーバーからプロジェクトをインポート**: リモート Git サーバー上のリポジトリから、プロジェクトパッケージをインポートします。

> 重要　リモート Git サーバーからインポートするプロジェクトパッケージは、最初に SAS Event Stream Processing Studio からリモートリポジトリにパブリッシュ公開されている必要があります。

リモートサーバーからプロジェクトをインポートオプションを使用して、プロジェクトパッケージがすでに含まれているリポジトリではなく、空のリポジトリをインポートすることもできます。空のリポジトリのインポートは、個々のユーザーではなく組織に属するリポジトリにプロジェクトパッケージを保存する場合に便利です。サーバーの種類が GitHub または Gitea の場合は、このアプローチを使用できます。このアプローチを使用するには、自分が所有者ではなく組織のために、Git リモートサーバーに 2 つの空のリポジトリを作成します。リポジトリ名はプロジェクトパッケージの目的の名前と一致する必要があり、リポジトリ名の 1 つは**-config** または**_config** で終わる必要があります。たとえば、プロジェクトパッケージを myproject という名前にしたい場合は、myproject という名前の空のリポジトリと、myproject-config または myproject_config という空のリポジトリを作成する必要があります。

3 プロジェクトパッケージを含む ZIP ファイルをローカルファイルシステムからインポートする場合は、次の手順を実行します。

a **プロジェクトのインポート**ウィンドウでインポートするファイルを選択します。

b (オプション)このプロジェクトに関連付けられている大きなファイルの保存に Git LFS を使用する場合は、**Large File Storage(LFS)の使用**チェックボックスを選択します。詳細については、"ファイルベースのバージョン管理と Git Large File Storage"を参照してください。

注: Git LFS は、プロジェクトパッケージが次のいずれかの方法で保存されている場合に使用できます。

- SAS Event Stream Processing の内部にある Git リポジトリ
- Gitea のリモートリポジトリ

c (オプション)**プロジェクトごとにリモートリポジトリを作成**チェックボックスを選択します。インポート中のリモートリポジトリの作成は、SAS Event Stream Processing Studio にすでに存在するプロジェクトのリモートリポジトリを作成することと似ています。詳細については、"既存のプロジェクトのリモートリポジトリを作成"を参照してください。

4 ローカルファイルシステムから XML ファイルをインポートする場合は、次の手順を実行します。

a **プロジェクトのインポート**ウィンドウでインポートするファイルを選択します。同時に複数のファイルをインポートすることができます。

b (オプション)**プロジェクトごとにプロジェクトパッケージを作成**チェックボックスをオンにします。詳細については、"プロジェクトパッケージ"を参照してください。

c (オプション)このプロジェクトに関連付けられている大きなファイルの保存に Git LFS を使用する場合は、**Large File Storage(LFS)の使用**チェックボックスを選択します。詳細については、"ファイルベースのバージョン管理と Git Large File Storage"を参照してください。

注: Git LFS は、プロジェクトパッケージが次のいずれかの方法で保存されている場合に使用できます。

- SAS Event Stream Processing の内部にある Git リポジトリ
- Gitea のリモートリポジトリ

d (オプション)**プロジェクトごとにリモートリポジトリを作成**チェックボックスを選択します。インポート中のリモートリポジトリの作成は、SAS Event Stream Processing Studio にすでに存在するプロジェクトのリモートリポジトリを作成することと似ています。詳細については、"既存のプロジェクトのリモートリポジトリを作成"を参照してください。

5 リモート Git サーバーからプロジェクトをインポートする場合は、**リモート Git サーバーからプロジェクトをインポート**ウィンドウで、リモートサーバー接続とインポートするプロジェクトを選択します。

注: トピックは、リポジトリを整理するために使用されるラベルです。リモート Git サーバー接続の作成時にトピックが指定されている場合は、そのトピックを持つプロジェクトのみが**リモート Git サーバーからプロジェクトをインポート**ウィンドウに表示されます。汎用 Git サーバーに接続している場合、トピックは使用できないため、リストからプロジェクトを選択するのではなく、プロジェクトの名前を入力する必要があります。

リモートサーバー接続が設定されていないことを示すメッセージが表示された場合は、システム管理者に問い合わせてください。

詳細については、“リモート Git サーバーへの接続を指定する”を参照してください。

6 **インポート**をクリックします。

インポートしたプロジェクトが**プロジェクト**ページに表示され、プロジェクトが開かれます。

複数のプロジェクトをインポートすると、プロジェクトは**プロジェクト**ページに表示されますが、開かれません。

requirements.txt ファイルまたは **modules.txt** ファイルを含むプロジェクトパッケージをインポートする場合、プロジェクトを開くと、プロジェクトを正しく実行するために仮想環境が必要である可能性があることを知らせるメッセージが表示されます。**はい**をクリックして仮想環境を選択します。同様に、プロジェクトパッケージに、特定の Python パッケージまたは Lua モジュールを必要とするカスタムウィンドウが含まれている場合、必要なライブラリと選択した仮想環境の間の不一致を解決するようにメッセージが表示される場合があります。**解決**をクリックして、一致する仮想環境を選択します。仮想環境の選択の詳細については、“プロジェクトプロパティの更新”を参照してください。

空のリポジトリをインポートした場合、インポートした空のリポジトリの名前を反映した名前でプロジェクトが作成されます。インポートしたプロジェクトが**プロジェクト**ページに表示され、プロジェクトが開かれます。

注: 以前にパブリッシュされたプロジェクトバージョンと同じ名前のプロジェクトをインポートすることはできません。これは、後で SAS Event Stream Processing Studio から削除されたプロジェクトバージョンと同じ名前のプロジェクトにも適用されます。

プロジェクトプロパティの更新

特定のウィンドウのプロパティではなく、プロジェクトレベルのプロパティにアクセスするには、ツールバーの をクリックします。

モデルの作成を開始する前に、デフォルトのプロジェクトプロパティを確認し、必要に応じて更新してください。詳細については、を参照してください。仮想環境、コネクタのオーケストレーション、SAS Micro Analytic Service モジュール、スコアリング API、計算要件、エラー処理などの機能のプロパティを更新できます。次のセクションでは、一部の機能について詳しく説明します。

仮想環境の選択

プロジェクトがプロジェクト パッケージの一部である場合は、ツールバーの をクリックし、**仮想環境**セクションを使用して、Python パッケージまたは Lua モジュールのセットを指定する仮想環境を選択できます。Python 仮想環境と Lua 仮想環境をそれぞれ 1 つ選択できます。プロジェクトが環境内にない Python パッケージまたは Lua モジュールに依存している場合、プロジェクトを実行するとエラーが報告されます。

Python 仮想環境の選択

Python 仮想環境の選択に関する情報は次のとおりです。

- Python **ビルトイン環境**オプションを選択すると、プロジェクトではデフォルトの Python パッケージのセットが使用されます。組み込み環境に含まれるパッケージを表示するには、をクリックします。
- **Python の依存関係なし**オプションを選択すると、ビルトイン環境は使用されません。基本的な Python コードを実行することができますが、追加の Python パッケージは含まれていません。SAS Event Stream Manager を使用してプロジェクトのプロジェクトコンテナーを作成する場合、このオプションを選択すると、より迅速なイメージの作成とより小さいコンテナー画像が得られます。詳細については、"Project Containers" (*SAS Event Stream Processing: Using SAS Event Stream Manager*)を参照してください。
- 別の仮想環境を選択すると、プロジェクトはビルトイン環境のコア Python 機能、および選択した仮想環境で定義されている Python パッケージを使用します。環境に含まれる Python パッケージを表示するには、をクリックします。システム管理者が仮想環境を作成する方法の詳細については、"仮想環境の指定"を参照してください。
- Python **ビルトイン環境**オプションを使用するか、Python 仮想環境を選択すると、プロジェクトパッケージの **home** フォルダー、**home/analytics/**フォルダー、および **home/files/**フォルダーが Python の検索パスに含まれます。**Python 依存関係なし**オプションを選択すると、検索パスに**ホーム**フォルダーのみが含まれます。

Lua 仮想環境の選択

Lua 仮想環境の選択に関する情報は次のとおりです。

- Lua **ビルトイン環境**を選択すると、基本的な Lua コードを実行できますが、Lua モジュールは含まれません。
- **Lua の依存関係なし**オプションを選択すると、ビルトイン環境は使用されません。基本 Lua コードを実行できますが、Lua モジュールは含まれていません。このオプションは、Lua **ビルトイン環境**オプションと同じ効果があります。
- 別の仮想環境を選択すると、プロジェクトはビルトイン環境のコア Lua 機能、および選択した仮想環境で定義されている Lua モジュールを使用します。環境に含まれる Lua モジュールを表示するには、をクリックします。システム管理者が仮想環境を作成する方法の詳細については、"仮想環境の指定"を参照してください。
- **ホーム**フォルダーは、選択された仮想環境オプションに関係なく、Lua の検索パスに含まれます。

仮想環境ファイルが検出された場合に一致する環境を選択する

requirements.txt ファイルまたは modules.txt ファイルを含むプロジェクトパッケージを開いても、そのプロジェクトに仮想環境が選択されていない場合は、**仮想環境ファイルが検出されました**ウィンドウが表示されます。

requirements.txt ファイルは、Python 仮想環境の構成を指定します。modules.txt ファイルは、Lua 仮想環境の構成を指定します。

仮想環境ファイルの検出ウィンドウが表示されたら、次のようにします。

1 **はい**をクリックして、仮想環境の選択を示します。

2 **仮想環境**ウィンドウで、**既存の一致する環境**ドロップダウンリストを使用して、適切な仮想環境を選択します。

適切な仮想環境がない場合で、自分がシステム管理者である場合、**仮想環境**ウィンドウを使って仮想環境を作成することができます。仮想環境を作成および更新するための追加機能は、システム管

理者が**設定**ページの**仮想環境**セクションで使用できます。詳細については、“仮想環境の指定”を参照してください。

注: 管理者グループのメンバーのみが、**仮想環境**ウィンドウを使用して仮想環境を作成できます。他のユーザーは、このウィンドウを使用して既存の仮想環境を選択できます。デフォルトでは、管理者グループは SASAdministrators です。

3 **OK** をクリックします。

プロジェクトを MCP サーバーとしてパブリッシュする

モデルコンテキストプロトコル(MCP)は、大きな言語モデル(LLM)と API、データベース、リモートプロシジャなどの外部リソースとの間のインターフェイスを提供します。*MCP ツール*は作業を実行するために LLM によって呼び出されます。これらのツールは、推論の一部として使用でき、応答を作成するために使用できる出力を備えた LLM を提供します。これらのツールは、*MCP サーバー*により利用可能になります。詳細については、MCP ドキュメントを参照してください。

SAS Event Stream Processing の MCP 統合は、MCP が LLM と実行中の ESP プロジェクトの間のインターフェイスとして使用できることを意味します。プロジェクトが実行されると、LLM が対話できる MCP サーバーとしてその機能の一部をパブリッシュできます。つまり、プロジェクトは、LLM が連絡できるリソースになります。たとえば、ウィンドウを使用してデータを分析したり、ウィンドウにデータを挿入したりすることができます。プロジェクトが LLM とどのように連携するかを設定するには、SAS Event Stream Processing Studio を使用して MCP ツールを作成します。ツールは入力を受け取り、ESP サーバーでタスクを実行し、結果を LLM に送り返すことができます。ツールを作成するときに、プロジェクト LLM が相互作用できる機能を決定します。また、LLM を操作しているユーザーに表示できるプロンプトも作成します。SAS Event Stream Processing の MCP 統合の詳細については、“Using the Model Context Protocol” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

注: 組織に、リモート MCP サーバーとやりとりできる LLM が必要です。

MCP サーバーとして使用するプロジェクトを構成するには:

1 ツールバーの をクリックします。

2 **モデルコンテキストプロトコル**セクションを展開します。

3 作成ツール:

   a **ツール**セクションで、 をクリックします。

   b **モデルコンテキストプロトコルプロトコルの追加**ウィンドウで、ツールの名前、タイトル、および説明を入力します。LLM は、タイトルと説明の情報を使用して、ユーザーが実行したいタスクを実行するためにツールを使用できるかどうかを判断します。

   c **種類**フィールドで種類を選択します。

      - スコアツールを使用すると、LLM は入力イベントをスコアリングし、スコアリングされたイベントを返すことができます。結果は、ツール構成で指定した出力ウィンドウからイベントデータとして配信されます。詳細については、“Using the Scoring API” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

- クエリツールを使用すると、LLM をウィンドウのコンテンツにクエリできます。このツールは、指定された出力ウィンドウのコンテンツを直接 LLM に返します。
- サブスクライブツールを使用すると、LLM はモデル内の ESP ウィンドウにサブスクライブし、イベントデータの形式で通知を受信できます。クエリツールとサブスクライブツールの違いは、サブスクライブツールがイベントを要求元の LLM に直接返さないことです。代わりに、サブスクライブツールはウィンドウへのサブスクリプションを設定し、イベントをリッスンします。イベントはリスナーによって受信されると、永続的な SSE 接続を介してサーバー送信イベント(SSE)として送信されます。
- パブリッシュツールにより、LLM はデータを ESP モデルにパブリッシュできます。データは、特定のフィールドにパブリッシュする場合は CSV 形式のイベントまたは文字列形式にすることができます。
- カスタムツールを使用すると、モデルのデザインは Lua または Python コードを記述して、入力イベントを受け入れ、LLM に返される結果を生成できます。

d スコアツールを作成しているときは、次のアクションを実行します。

1. 必要に応じて、**前**フィールドを使用して、モデルにデータをパブリッシュする前に実行する関数の名前を入力します。Python または Lua コードをステップステップ 5 に入力します。
2. 必要に応じて、**ポスト**フィールドを使用して、スコアリングの完了後に実行する関数の名前を入力します。Python または Lua コードをステップステップ 5 に入力します。
3. 必要に応じて、**イベント**フィールドを使用して、入力イベントブロックからイベントが作成されたときに実行する関数の名前を入力します。Python または Lua コードをステップステップ 5 に入力します。
4. **入力**タブを使用して、入力ウィンドウと関連するフィールドを選択します。バイナリフィールドの場合、MIME タイプを設定して、イベントフィールドに含まれるバイナリデータのタイプを示すことができます。

   フィールドを配列として返すチェックボックスを選択し、配列の名前を入力することもできます。
5. **出力**タブを使用して、入力ウィンドウと関連するフィールドを選択します。バイナリフィールドの場合、MIME タイプを設定して、イベントフィールドに含まれるバイナリデータのタイプを示すことができます。

e クエリツールまたはサブスクライブツールを作成しているときは、次のアクションを実行します。

1. **出力**タブを使用して、クエリまたはサブスクライブする出力ウィンドウと関連するフィールドを選択します。バイナリフィールドの場合、MIME タイプを設定して、イベントフィールドに含まれるバイナリデータのタイプを示すことができます。
2. 必要に応じて、**フィルター**タブを使用して、特定の基準に一致するイベントへのレスポンスを制限します。つまり、出力ウィンドウからのデータは返される前にフィルタリングできます。**フィルター**タブで、データをフィルタリングする関数の名前を入力し、関連するフィールドを追加します。フィールドは LLM プロンプトから入力され、オブジェクトのプロパティとして関数に提供されます。たとえば、最大値を超える値をツールで除外する場合は、最大値のフィールドを追加します。このフィールドには、LLM プロンプトから"get all values under 100"などの値が与えられます。

f パブリッシュツールを作成しているときは、次のアクションを実行します。

1. **入力ウィンドウ**フィールドを使用して、イベントをパブリッシュするソースウィンドウを選択します。
2. 必要に応じて、**フィールド**フィールドを使用して、ソースウィンドウでフィールドを選択します。データコンテンツはこのフィールドにパブリッシュされます。
3. 必要に応じて、**データ**フィールドを使用して、パブリッシュするデータのコンテンツを指定します。LLM はこのデータを要求でも指定できます。
4. 必要に応じて、 **URL** フィールドを使用して、パブリッシュするデータのコンテンツを含む URL を指定します。LLM はこのデータを要求でも指定できます。
5. 必要に応じて、**ブロックサイズ**フィールドを使用して、データのパブリッシュ時に各イベントブロックに含めるイベントの数を指定します。LLM はこのデータを要求でも指定できます。

g. カスタムツールを作成しているときは、次のアクションを実行します。

1. **呼び出し**フィールドに、呼び出す Python または Lua 関数の名前を入力します。Python または Lua コードをステップステップ 5 に入力します。
2. 必要に応じて、**入力**タブを使用して入力フィールドを追加します。

   フィールドを配列として返すチェックボックスを選択し、配列の名前を入力することもできます。
3. **出力**タブを使用して、出力フィールドを追加します。

h. 必要に応じて、別のツールを作成します。

4. プロンプトを作成します。

a. **プロンプト**セクションで、をクリックします。

b. **モデルコンテキストプロトコルプロトコルの追加**ウィンドウで、プロンプトの名前、タイトル、および説明を入力します。

c. **テキスト**フィールドに、ダイアログクライアントによってダイアログウィンドウに 表示されるテキストなど、プロンプトのテキストを入力します。テキスト内の先頭に$文字が付いている値はすべて変数として扱われ、ユーザーに値を入力するよう求められます。

d. **テキスト**フィールドに入力した変数は、隣接するテーブルに自動的に追加されます。各変数の説明を入力します。説明はユーザーに表示され、ユーザーが値を入力するようにガイドするテキストが含まれている必要があります。

5. 必要に応じて、**コード**セクションを使用して、追加したツールをサポートする Python コードまたは Lua コードを指定します。エディターを使用してコードを入力したり、プロジェクト内に存在するスニペットを指定したり、コードを含むファイルを選択したりできます。

6. プロジェクトを保存して配置します。たとえば、テストモードでプロジェクトを実行します。プロジェクトが実行されると、オンデマンドで ESP サーバーが作成され、MCP サーバーがパブリッシュされます。

7. LLM を使用して、ESP サーバーの Ingress にある/eventStreamProcessing/v2/mcp エンドポイントの MCP サーバーに接続します。

8. LLM を使用してツールを実行します。

スコアリング API の有効

Scoring API を使用して入力イベントをスコアリングし、オンデマンドでスコアリングされたイベントを返したい場合。

1 ツールバーの  をクリックします。

2 **スコアリング API** セクションを展開し、**スコアリング API を有効にする**を選択します。

3 次のフィールドを使用して、入力イベントと出力イベントに使用されるウィンドウとフィールドを指定します:

- **入力ウィンドウ**フィールドを使用して、スコア要求の本文で渡されるイベントデータを受け取るソースウィンドウを選択します。
- **出力ウィンドウ**フィールドを使用して、スコア要求に応答して作成されるイベントのモニターされるウィンドウを選択します。
- 必要に応じて、**入力フィールド**フィールドを使用して、ソースウィンドウで要求本文を受け取るフィールドを選択します。このフィールドは、プロジェクトがプロジェクトパッケージの一部である場合にのみ使用できます。
- 応答を生データ(たとえば、画像)として返す場合は、**出力フィールド**フィールドを使用して、出力ウィンドウにフィールドを指定します。このフィールドのコンテンツは、スコア要求に応じて返されます。

スコアリング API の一般情報については、"Using the Scoring API" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。SAS Event Stream Processing Studio を使用してスコアリングするイベントを送信したり、スコアリングされたイベントを表示したりする方法の詳細については、"モデルのテスト時にスコアリング API を使用する"を参照してください。

計算要件の指定

ツールバーの  をクリックし、**計算案件**セクションを使用して、プロジェクトに必要なメモリ、CPU リソース、および NVIDIA グラフィックス プロセッシング ユニット(GPU)リソースの推奨最小量を指定します。

これらの詳細を指定することで、プロジェクトの他のユーザーが、必要なリソースを知ることができます。自分または他のユーザーが SAS Event Stream Processing Studio または SAS Event Stream Manager でプロジェクトを実行する場合、計算要件が満たされていない場合は**クラスターのプロジェクトをロードして開始ウィンドウに**警告メッセージが表示されます。

注: システム管理者は、メモリ量と CPU リソースの割り当て量に制限を設定することができます。

詳細については、"Kubernetes クラスター内にある ESP サーバーでのプロジェクトのロードと開始"を参照してください。

エラー処理の戦略の選択

1 ツールバーの  をクリックします。

2 **詳細**セクションを展開します。

3 **致命的なエラーが発生した場合のエラー処理戦略**フィールドを使用して、**エラーをログに記録してプロジェクトの実行を続行**または**プロジェクトを停止して削除**を選択します。

注: プロジェクトレベルで指定したエラー処理設定は、Lua ウィンドウ、Python ウィンドウ、およびカスタムウィンドウのエラー処理設定によってオーバーライドできます。詳細については、次を参照してください。

- “Lua の Windows の操作 SAS Event Stream Processing Studio”
- “Python ウィンドウの操作 SAS Event Stream Processing Studio”
- “プロジェクトにカスタムウィンドウを追加”

プロジェクトの削除

プロジェクトは、プロジェクトの所有者または SASAdministrators グループのメンバーによって 削除できます。

プロジェクトを削除するには、次のようにします:

1 **プロジェクト**ページのテーブルから削除するプロジェクトを選択します。

> ヒント　複数のプロジェクトを同時に削除するには、Ctrl キーを押しながらするプロジェクトを選択します。

2 🗑をクリックします。

3 削除を確認します。

管理者ではないユーザーが SAS Event Stream Processing Studio を使用してプロジェクト XML ファイルを削除した場合、プロジェクトに関連付けられている SAS Micro Analytic Service ストアは自動的に削除されません。ストアは孤立しています。

以下の条件が満たされる場合、確認ウィンドウには SAS Event Stream Manager からプロジェクトを削除するオプションも表示されます。SAS Event Stream Manager からプロジェクトを削除すると、プロジェクトに関連付けられている SAS Micro Analytic Service ストアも削除されます。

- SAS Event Stream Processing は他の SAS 製品とともに配置され、SAS Viya プラットフォームサービスを使用してプロジェクトバージョンがパブリッシュされます。
- パブリッシュ済みのプロジェクトを削除しています。
- SASAdministrator グループのメンバーであり 、SAS Event Stream Processing Studio にサインインした場合、**想定されるグループ**ウィンドウで管理者権限を承認したことになります。

プロジェクトは完全に削除されます。そのプロジェクトがプロジェクトパッケージの一部になっている場合には、そのプロジェクトパッケージ内のすべてのファイルを 含めて、そのプロジェクトパッケージ全体が削除されます。詳細については、“プロジェクトパッケージ”を参照してください。

プロジェクトのエクスポート

1 **プロジェクト**ページで、エクスポートする配置を選択します。

2 ⁝をクリックします。

3 関連するオプションを選択します。プロジェクトのタイプによって、プロジェクトがエクスポートされる形式が決まります。

- 選択したプロジェクトがプロジェクトパッケージの一部である場合、プロジェクトパッケージを ZIP ファイルとしてエクスポートできます。**output** フォルダーと **temp** フォルダーはエクスポートされません。
- 選択したプロジェクトがプロジェクトパッケージの一部ではない場合、プロジェクトを XML ファイルとしてエクスポートできます。

プロジェクトがコンピュータにエクスポートされます。

注: エクスポートしたプロジェクトの場所は、ブラウザの構成によって異なる場合があります。

既存のプロジェクトのリモートリポジトリを作成

ファイルベースのバージョン管理が使用されている場合、既存のプロジェクトのリモート Git リポジトリを作成できます。リモートの詳細については、“ファイルベースのバージョニングと Git”を参照してください。

既存のプロジェクトのリモートリポジトリを作成:

1 **プロジェクト**ページで、リモートリポジトリを作成するプロジェクトをダブルクリックします。

2 ⁝をクリックします。

3 **リモートリポジトリの作成**を選択します。

4 **リモートリポジトリの作成**ウィンドウで、リモートリポジトリ接続を選択します。

注:

- リモートサーバー接続が設定されていないことを示すメッセージが表示された場合は、システム管理者に問い合わせてください。
- Git サーバーのタイプは、接続名の後の括弧内に表示されます。次に例を示します: **MyGitServer(GitLab)**。タイプが **Generic** である Git サーバーを選択した場合、プロジェクトの名前と一致する名前を持つ空のリポジトリがその Git サーバー上にすでに存在している必要があります。
- 詳細については、“リモート Git サーバーへの接続を指定する”を参照してください。

5 **作成**をクリックします。

リモートリポジトリが作成され、プロジェクトパッケージの **home** フォルダーがリモートリポジトリと同期されます。

プロジェクトページの表でプロジェクトを表示すると、リモートリポジトリ列に、プロジェクトがリモートリポジトリにあることが示されます。

プロジェクトを空のリモートリポジトリにリンク

プロジェクトを空のリモートリポジトリにリンクすると、個々のユーザーではなく組織に属するリポジトリにプロジェクトパッケージを保存する場合に便利です。サーバーの種類が GitHub または Gitea の場合は、このアプローチを使用できます。リモートの詳細については、“ファイルベースのバージョニングと Git”を参照してください。

プロジェクトを空のリモートリポジトリにリンクするには、次の操作を実行します。

1 自分が所有者としてではなく、組織に従ってリモートサーバー上に 2 つの空のリポジトリを作成します。リポジトリ名はプロジェクトの名前と一致する必要があり、また、リポジトリ名の 1 つは**-config** または**_config** で終わる必要があります。たとえば、プロジェクトが myproject という名前の場合、myproject という名前の空のリポジトリと、myproject-config または myproject_config という空のリポジトリを作成する必要があります。

2 **プロジェクト**ページで、リモートリポジトリを作成するプロジェクトをダブルクリックします。

3 ⁝をクリックします。

4 **リモートリポジトリの作成**を選択します。

5 **プロジェクトを Git サーバーにリンク**ウィンドウで、リモートリポジトリ接続を選択します。

> **注:** リモートサーバー接続が設定されていないことを示すメッセージが表示された場合は、システム管理者に問い合わせてください。詳細については、“リモート Git サーバーへの接続を指定する”を参照してください。

6 **リモートリポジトリ**フィールドで、リモート Git サーバー上に作成した空のリポジトリを選択します。

7 **リンク**をクリックします。

リモートリポジトリの詳細を表示

ファイルベースのバージョン管理が使用されている場合、プロジェクトはリモート Git サーバー上のリポジトリに保存できます。詳細については、“ファイルベースのバージョニングと Git”を参照してください。

プロジェクトがリモートリポジトリに保存されている場合、リモートリポジトリの詳細を表示できます。

1 **プロジェクト**ページで、関連するプロジェクトをダブルクリックして開きます。

> ヒント　リモートリポジトリ列を使用して、リモートリポジトリに保存されているプロジェクトを識別できます。

2 ⋮をクリックします。

3 **リポジトリの詳細を表示**を選択します。

ウィンドウが表示され、このプロジェクトと関連するリモート Git サーバーとの接続に関する情報が表示されます。たとえば、プロジェクトが GitLab に保存されている場合、**リポジトリの表示**リンクをクリックすると、GitLab のリモートリポジトリを表示できます。

注意
リモートリポジトリを表示することはできますが、操作しないでください。リモート Git リポジトリと対話するには、SAS Event Stream Processing のみを使用してください。 リモート Git サーバー上に SAS Event Stream Processing プロジェクトを保存するリポジトリと直接対話すると、データ損失が発生する可能性があります。

プロジェクトのアクセス制御

SAS Event Stream Processing Studio でプロジェクトを作成またはインポートすると、プロジェクトの所有者になります。プロジェクトが作成またはインポートされると、デフォルトで制限され、プライベートとマークされ、所有者と SASAdministrators グループのユーザーのみがアクセス、更新、削除、またはパブリッシュできます。

パブリックプロジェクトは、すべてのユーザーがアクセス 、更新、またはパブリッシュできます。

プライベートプロジェクトは、所有者か SASAdministrators グループのメンバーのみがパブリックにできます。同様に、パブリックプロジェクトは、所有者か SASAdministrators グループのメンバーによってのみプライベートにできます。

ユーザーが組織を離れても、そのユーザーのプライベートプロジェクトはシステムから消えません。SASAdministrators グループのメンバーは、これらの非パブリッシュプロジェクトをパブリッシュできます。

SASAdministrators グループのメンバーが SAS Event Stream Processing Studio にログオンする際には、**想定されるグループ**ウィンドウで管理者権限を受け入れる必要があります。これを行わないと、プロジェクトへのアクセスの管理者権限が与えられません。

SAS Event Stream Processing が他の SAS 製品と共に展開され、プロジェクトアクセスコントロールが含まれていないリリースからアップグレードされる場合、アップグレード後に既存のプロジェクトは自動的にプライベートになります。スタンドアロンが SAS Event Stream Processing プロジェクトアクセスコントロールを含まないリリースからアップグレードされる場合、既存のプロジェクトはアップグレード後もパブリックのままです。

プロジェクトのアクセスレベルを変更するには 2 つの方法があります。

- **プロジェクト**ページで。この方法では、複数のプロジェクトのアクセスレベルを同時に変更することができます。詳細については、“プロジェクトの操作の概要”を参照してください。

- 開いているプロジェクトの表示中。詳細については、“SAS Event Stream Processing Studio Modeler の使用”を参照してください。

プロジェクトのロック

SAS Event Stream Processing Studio で編集中のプロジェクトは自動的にロックされます。これにより、プロジェクトの変更を他のユーザーが誤って上書きすることがないようにします。プロジェクトを開くと、編集しようとしていると見なされます。プロジェクトが別のユーザーによってロックされていない場合、そのプロジェクトは、開いた直後にロックされます。プロジェクトを編集している場合、アプリケーションでそのプロジェクトを含むタブを閉じると、ロックは自動的に解除されます。ロックは、アプリケーションを閉じた約 2 分後にも解除されます。たとえば、コンピュータの電源を切ったりブラウザを閉じたりすると、他のユーザーは、2 分待たなければ、プロジェクトをロックできません。

別のユーザーによってロックされているプロジェクトを 開こうとすると、別のユーザーがそのプロジェクトを編集中であることが通知されます。続行するかどうかを決定するプロンプトが表示されます。**はい**をクリックすると、SAS Event Stream Processing Studio 読み取り専用モードのモデラーでプロジェクトが開きます。**いいえ**をクリックすると、**プロジェクト**ページに戻ります。

読み取り専用モードでは、モデルを表示し、必要に応じてモデルのウィンドウの位置を再調整できます。ただし、変更を保存することはできません。

注: プロジェクトはユーザー名に対してロックされています。プロジェクトを編集している場合は、別のブラウザタブまたはウィンドウでプロジェクトを開くことはお勧めできません。

プロジェクトの読み取り専用コピーを開くと、バナーが表示されます。バナーは、プロジェクトを編集しているユーザーと、プロジェクトの読み取り専用コピーが開かれていることを示しています。

ウィンドウペインと拡大アイコンは読み取り専用モードでは使用できません。したがって、**ウィンドウ**ペインを使用してモデルにウィンドウを追加したり、モデルの倍率を調整したりすることはできません。

プロジェクトメタデータ

プロジェクトを作成すると、プロジェクトを識別する次の一意の情報が自動的に作成されます。

- プロジェクトを作成したユーザーのユーザー ID
- プロジェクトを最後に変更したユーザーのユーザー ID
- プロジェクトが作成された日付（UNIX のエポック時間)
- プロジェクトが最後に変更された日付（UNIX のエポック時間)

プロジェクト情報は、プロジェクトの XML コードの<metadata>エレメント内に表示されますが、SAS Event Stream Processing Studio データベースに格納されています。

メタデータは、次のいずれかのアクションを実行した場合にも作成されます。

- プロジェクトにタグを適用する

- ワークスペースでモデルを 変更する

  注: ワークスペースでモデルを 変更すると、<meta id="layout">エレメントが追加されます。このエレメントでは、モデルの連続クエリの名前、モデル内のウィンドウの名前、およびワークスペース内の各ウィンドウの X および Y 座標が指定されます。

- SAS Model Manager で作成された SAS Micro Analytic Service モジュールを含むモデルを SAS Event Stream Processing Studio にインポートします。詳細については、“概要”を参照してください。

SAS Viya プラットフォームサービスを使用するプロジェクトのバージョンをパブリッシュすると、そのバージョンを識別する次の一意の情報が自動的に作成されます:

注: 次のエレメントは、作業モデルの XML コードには表示されません。ただし、モデルをパブリッシュし、そのモデルを読み取り専用モードで表示している場合、これらのエレメントはモデルの XML コードに表示されます。

- プロジェクトの一意の ID
- プロジェクトバージョンのメジャーバージョン番号
- プロジェクトバージョンのマイナーバージョン番号

以下は、SAS Viya プラットフォームサービスを使用してパブリッシュされた、パブリッシュされたプロジェクトバージョンのメタデータの例です:

```
<metadata>
  <meta id="studioUploadedBy">espuser</meta>
  <meta id="studioUploaded">1614859578600</meta>
  <meta id="studioModifiedBy">espuser</meta>
  <meta id="studioModified">1614860116540</meta>
  <meta id="layout"><![CDATA[{"contQuery":{"aggregateWindow":{"x":50,"y":175},"sourceWindow":
{"x":50,"y":50}}}]]></meta>
  <meta id="studioVersionMajor">1</meta>
  <meta id="studioVersionMinor">0</meta>
  <meta id="studioId">c835319d-85bf-48e6-aaa5-c27b40bb1eee</meta>
</metadata>
```

この例では、プロジェクトのバージョン番号は 1.0 です。

ファイルベースのバージョン管理を使用するプロジェクトバージョンを公開する場合、プロジェクトの XML コードにはプロジェクトバージョンに関するメタデータが含まれません。

プロジェクトパッケージの操作

プロジェクトパッケージ

プロジェクトの XML ファイルには、ストリーミングデータの処理に必要なウィンドウやエッジなどの要素が含まれています。しかし、プロジェクト XML ファイルは、入力ファイル、分析ストア、カスタムコネクタ、およびプラグインなどの、XML 外にある追加リソースを参照することもできます。プロジェクトが正常に実行されるためには、これらの追加リソースが、プロジェクトが実行される ESP サーバーで使用できるようになっていなければなりません。

プロジェクトパッケージには、プロジェクト XML ファイル、およびプロジェクトが参照するファイルを含む標準の一連のフォルダーが含まれています。プロジェクト XML ファイルと追加ファイルは、永続ボリューム(PV)に格納されています。PV は、Kubernetes クラスター内の 1 つのストレージです。

> 重要　プロジェクトパッケージを使用するには、SAS Event Stream Processing を Kubernetes 環境に配置する場合、永続ボリュームにアクセスするように構成する必要があります。この構成では、オーバーレイを適用します。詳細については、*"永続ボリュームの管理" (SAS Event Stream Processing: Kubernetes 環境での SAS Event Stream Processing の使用)*を参照してください。

プロジェクトパッケージを使用することで、プロジェクトが参照するファイルを容易に管理できます。

- SAS Event Stream Processing Studio プロジェクトパッケージの機能を使用すると、PV 管理ツールを使用せずに、PV 上でのフォルダーの作成や PV へのファイルのアップロードなどのタスクを実行できます。
- ウィンドウからファイルを参照する場合、多くの場合、SAS Event Stream Processing Studio を使用すると、パスを手動で入力するのではなく、プロジェクトパッケージ内のファイルに移動できます。ファイルへのナビゲートは、パスを手入力するよりもミスが起こりにくいです。
- ウィンドウから参照したいファイルがプロジェクトパッケージにまだ存在しない場合は、ファイルを別々にアップロードするのではなく、ファイル参照を設定するプロセスの一部として、そのファイルをプロジェクトパッケージにアップロードして、それらを参照可能にすることができます。さらに、SAS Event Stream Processing Studio は、ファイル参照を追加するプロジェクトの部分に応じて、ファイルをアップロードする適切なフォルダーを提案します。各ファイルの目的に応じた異なるフォルダーにファイルを配置することで、ファイルを整理した状態でプロジェクトパッケージに保持しやすくなります。ここでは、例をいくつか紹介します。
  - ウィンドウのファイルとソケットのパブリッシャーコネクタを構成し、ファイルを参照するプロセスの一部として CSV ファイルをアップロードする場合、SAS Event Stream Processing Studio は、CSV ファイルを **test_files** フォルダーに配置することを推奨します。
  - Model Reader ウィンドウからモデルを参照し、ファイルを参照するプロセスの一部としてモデルファイルをアップロードすると、SAS Event Stream Processing Studio は、モデルファイルを **Analytics** フォルダーに配置することを推奨します。

  - 1 つのウィンドウの file-and-socket サブスクライバーコネクタを構成する場合、まだ存在しない出力ファイルの名前を指定することができます。プロジェクトをテストモードで実行する場合で、出力ファイルが作成される場合、あなたが選択したフォルダーに配置されます。SAS Event Stream Processing Studio は、出力ファイルの場所として**出力**フォルダーにすることを推奨します。
- プロジェクトパッケージ内のファイルを参照、またはプロジェクトパッケージにファイルをアップロードする場合、SAS Event Stream Processing Studio は、表示をプロジェクト パッケージに制限します。PV 上の別の場所(つまり、プロジェクトパッケージの外)にあるファイルに移動したり、PV の別の部分にファイルをアップロードしたりすることはできません。

ファイルベースのバージョン管理を使用すると、プロジェクトパッケージを公開できます。プロジェクトパッケージをパブリッシュすると、プロジェクトパッケージ内のすべてのファイルがバージョン管理されます。詳細については、“ファイルベースのバージョン管理”を参照してください。

プロジェクトパッケージの作成

新しいプロジェクトの作成時にプロジェクトパッケージを作成する

新しいプロジェクトを作成するプロセスの一部としてプロジェクトパッケージを作成できます。詳細については、“プロジェクトの作成”を参照してください。

既存のプロジェクトのプロジェクトパッケージを作成する

既存プロジェクトにプロジェクトパッケージを作成する前に、プロジェクトパッケージが目下のプロジェクトに適しているかどうかを検討してください。プロジェクトパッケージは、プロジェクトでそのプロジェクトパッケージ外(PV 上の他の場所)に保存されているファイルを参照する必要がある場合は、適していません。例えば、プロジェクトが、複数のプロジェクトで共有されるファイルを参照する場合があります。プロジェクトパッケージでは、ファイルに移動するスコープは、同一プロジェクトパッケージ内にあるファイルに制限されています。この制限によって、必要なすべてのファイルをプロジェクトパッケージ内に配置しやすくなります。プロジェクトパッケージを使用するには、それらの参照先ファイルをそのプロジェクトパッケージ内に移動して、それらのファイルの参照を調整することが必要です。

SAS Viya プラットフォームサービスを使用してプロジェクトバージョンをパブリッシュする場合、パブリッシュ済みのプロジェクトのプロジェクトパッケージを作成することはできません。

既存のプロジェクト用のプロジェクトパッケージを作成するには、

1 **プロジェクト**ページで、関連するプロジェクトを見つけて、プロジェクトパッケージが既に存在するかどうかを確認します。テーブルの**タイプ**列の値が **XML** の場合、プロジェクトパッケージは存在しません。

2 プロジェクトをダブルクリックして開きます。

  SAS Event Stream Processing Studio Modeler が表示されます。

3 左側のツールバーの  をクリックします。

プロジェクトパッケージペインが表示されます。

4 **パッケージの作成**をクリックします。

標準の一連のフォルダーを含むプロジェクトパッケージが永続ボリューム(PV)上に作成されます。**プロジェクトパッケージ**ペインにそれらのフォルダーが表示されます。

プロジェクトは、そのプロジェクトパッケージの **model** フォルダーに model.xml として保存されます。つまり、プロジェクトは SAS Event Stream Processing Studio の内部データベースのみではなく、PV 上に存在するようになります。

5 プロジェクト内の既存のファイル参照を調整します。例については、"既存のプロジェクトのプロジェクトパッケージを作成する例"を参照してください。

6 をクリックして、プロジェクトします。

既存のプロジェクトのプロジェクトパッケージを作成する例

次の手順では、https://github.com/sassoftware/esp-studio-examples 内のプロジェクト例のひとつにプロジェクトパッケージを作成する方法が示されています。これらの手順には、パブリッシャーコネクタおよびサブスクライバーコネクタ内のファイル参照の調整と、分析ストアモデルへの参照の調整が含まれています。

1 GitHub から**/Analytics/analytics_modelReader_RPCA** ディレクトリを取得し、それをコンピュータ上に保存します。

2 **model.xml** ファイルを SAS Event Stream Processing Studio にインポートします。詳細については、"プロジェクトのインポート"を参照してください。

3 **プロジェクト**ページで、project_01 プロジェクトを右クリックし、 **プロジェクトを開く**を選択します。

4 左側のツールバーのをクリックします。

5 **プロジェクトパッケージ**ペインで、**パッケージの作成**をクリックします。

6 **model** フォルダー(**home** フォルダーのサブフォルダー)を展開し、 **model.xml** ファイルがそこに存在することを確認します。プロジェクトは SAS Event Stream Processing Studio の内部データベースのみではなく、PV 上に存在するようになります。

注: プロジェクトパッケージを作成すると、そのプロジェクトは常に model.xml という名前で保存されるようになります。この例では、プロジェクトの元のファイル名も model.xml になっています。

7 w_data ウィンドウのパブリッシャーコネクタを更新します。

a ワークスペースの w_data ウィンドウを選択します。このウィンドウはソースウィンドウです。

b 右ペインで、**入力データ(パブリッシャー)コネクタ**を展開します。

c pub コネクタを選択し、クリックします。

d **コネクタ構成**ウィンドウで、 **Fsname** フィールドに既存のファイル参照 **input.csv** が含まれていることを確認します。

e 🗀をクリックします。

f **ファイルの選択**ウィンドウで、**ホーム>test_files** が左上隅に表示されます。

g ⤒をクリックして **input.csv** ファイルをプロジェクトパッケージの **test_files** フォルダーにアップロードします。

h **OK** をクリックします。

i **コネクタ構成**ウィンドウで、**Fsname** の新しいパス: **@ESP_PROJECT_HOME@/test_files/input.csv** を確認します。

j **OK** をクリックします。

8 w_reader ウィンドウ内の分析ストアモデルへの参照を更新します。

a ワークスペースの w_reader ウィンドウを選択します。このウィンドウはモデルリーダーウィンドウです。

b **分析ストアモデルへのパス**フィールドに既存のファイル参照 **RPCA.sasast** が含まれていることを確認します。

c 🗀をクリックします。

d **ファイルの選択**ウィンドウで、左上隅に**ホーム>アナリティクス** が表示されていることを確認します。

e ⤒をクリックし、**RPCA.sasast** ファイルを **analytics** フォルダーにアップロードします。

f **OK** をクリックします。

g **分析ストアモデルのパス**フィールドで、**Fsname** フィールド内の新しいパス: **@ESP_PROJECT_HOME@/analytics/RPCA.sasast** を確認します。

9 w_score1 ウィンドウ内の分析ストアモデルへの参照が自動的に更新されたことを確認します。

a ワークスペースの w_score1 ウィンドウを選択します。このウィンドウはスコアウィンドウです。

b **分析ストア(ASTORE)ファイルからの入出力マップの生成**をクリックします。

c **入力マップと出力マップの生成**ウィンドウで、**ファイルへのパス**フィールドがプロジェクトパッケージの **analytics** フォルダーの **RPCS.sasast** ファイルを参照するように自動的に更新されていることを確認します。このフォルダーは、**@ESP_PROJECT_HOME@/analytics/RPCA.sasast** にあります。

手順 2 で SAS Event Stream Processing Studio にインポートした元の **model.xml** では、このフィールドはパスを指定せずに **RPCA.sasast** ファイルを参照していました。

d **キャンセル**をクリックします。

10 w_score1 ウィンドウのサブスクライバーコネクタを更新します。

a 右ペインで、**サブスクライバーコネクタ**を展開します。

b sub コネクタを選択し、🖉クリックします。

c **コネクタ構成**ウィンドウで、**パス**フィールドに、**result.out** への既存の参照が含まれていることを確認します。w_score1 ウィンドウは、このファイルに出力を書き込みます。

d **Fsname** フィールドで、既存の値を削除し、 **result.out** (同一の値)を入力します。**パス**フィールド内のパスが**@ESP_PROJECT_OUTPUT@/result.out** に変わることを確認します。プロジェクトをテストすると、**result.out** ファイルがプロジェクトパッケージ内のこの場所に書き込まれます。

e **OK** をクリックします。

11 プロジェクトパッケージのルートフォルダー **project01** を選択し、をクリックします。**home** フォルダー内の各フォルダーを拡張し、アップロードした RCPA.sasast および input.csv ファイルがプロジェクトパッケージ内に存在することを確認します。

12 をクリックして、プロジェクトします。

13 プロジェクトをテストします。詳細については、“テストを実行する”を参照してください。

注: プロジェクトをテストする際には、**w_reader** タブにデータは表示されません。これは、予想される動作です。

14 **project_01** ページに戻ります。

15 **プロジェクトパッケージ**ペインで、プロジェクトパッケージのルートフォルダー **project01** を選択し、 をクリックします。**output** フォルダーを展開して、**result.out** ファイルが作成されていることを確認します。

16 **result.out** ファイルを選択し、をクリックします。**result.out** ファイルがコンピュータにダウンロードされます。ファイルの場所は、ブラウザーの構成によって異なる場合があります。

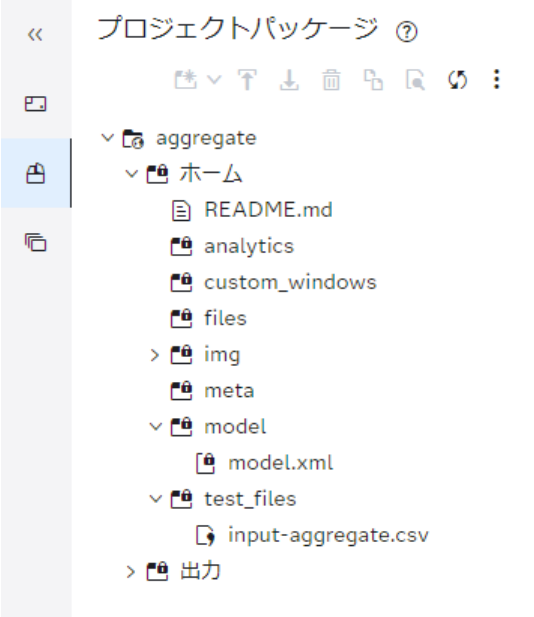
17 テキストエディタを使用して、**result.out** ファイルを開きます。このファイルには、w_score1 ウィンドウの出力イベントが含まれています。

プロジェクトパッケージを管理する

ユーザーインターフェイスの制御

次の図は、SAS Event Stream Processing Studio モデラーの**プロジェクトパッケージ**ペインのプロジェクトパッケージの例を示しています:

図 8　プロジェクトパッケージの例



プロジェクトパッケージのルートフォルダーは、ご使用のプロジェクトと同一の名前になっています。この例では、ルートフォルダーは **pose_estimation_with_onnx** です。

ファイル名の隣にあるテキスト**(LFS)**は、ファイルが Git Large File Storage に保存されていることを示します。詳細については、"ファイルベースのバージョン管理と Git Large File Storage"を参照してください。

プロジェクトパッケージペインのボタンを使用して、次の表に記述されている通りに、プロジェクトパッケージに属するファイルおよびフォルダーを管理します。

ボタン	説明
	選択したフォルダー内に追加のファイルまたはフォルダーを作成します。 **ホーム**フォルダーのサブフォルダー内に、ファイルとフォルダーを作成できます。たとえば、プロジェクトが多くの分析ファイルを参照する場合は、ファイルを整理しやすいように **analytics** フォルダー内に追加のフォルダーを作成した方がよい場合もあります。 ファイルを作成すると、選択したファイル形式がサポートされます。例としては、CSV、LUA、MD、PY、TXT などがあります。 新しいファイルまたはフォルダーが表示されない場合、をクリックします。
	選択したフォルダーにファイルをアップロードします。
	選択したファイルをダウンロードします。ファイルがコンピュータにダウンロードされます。ダウンロードしたファイルの場所は、ブラウザの構成によって異なる場合があります。 **model.xml** ファイルをダウンロードした場合、結果のファイルは、ファイル名を除けば、プロジェクトをエクスポートしたときと同じになります。

ボタン	説明
	プロジェクトパッケージ全体をダウンロードすることはできません。ただし、プロジェクトをローカルファイルシステムにエクスポートして、プロジェクトパッケージ内の各ファイルをローカルファイルシステムにダウンロードすることができます。 プロジェクトのエクスポートの詳細については、“プロジェクトのエクスポート”を参照してください。
	ファイルまたはフォルダーを削除します。複数のプロジェクトを同時に削除するには、Ctrl キーを押しながらするプロジェクトを選択します。 ルートフォルダーやルートフォルダー内の標準のフォルダーは削除できません。model.xml ファイルは削除できません。 ファイルまたはフォルダーを削除する場合は、そのファイルへのすべての参照をプロジェクトから削除する必要があります。
	選択したファイルまたはフォルダーへのパスをコピーします。その後、そのパスを SAS Event Stream Processing Studio の別の場所に貼り付けることができます。
	選択したファイルを開きます。この機能は、選択したファイルタイプで使用できます。画像の表示は、GIF、JPG、PNG などの選択した画像ファイルの種類で使用できます。 ファイルの種類によっては、ファイルの表示に加えて、ファイルを編集および検証できる場合があります。たとえば、PY ファイルを開いたときに、ファイルを編集して、Python コードの構文が有効であることを確認できます。 MD ファイル(たとえば、プロジェクトの **README.md** ファイル)を開いたとき、ファイルを編集して、その HTML レンダリングを表示することができます。詳細については、“README.md ファイルの使用”を参照してください。
	選択したファイルがビデオの場合、  ボタンの代わりに  ボタンが表示されます。をクリックして、ビデオを再生します。MP4 ファイル形式と H.264 コーデックがサポートされています。WEBM ファイル形式もサポートされています。
	すべてのフォルダーとその内容を更新します。このボタンを使用して、**プロジェクトパッケージ**ペインに表示されているフォルダーおよびファイルが、永続ボリュームに存在するフォルダーおよびファイルを反映していることを確認してください。
	このメニューを使用すると、次のタスクを実行できます。 ■ SAS Model Manager からエクスポートした ZIP ファイルをインポートします。詳細については、“SAS Model Manager ZIP ファイルのインポート”を参照してください。 ■ 削除したい一時ファイルが含まれている場合は、 **temp** フォルダーをクリアします。 ■ **README.md** ファイルを作成します。詳細については、“README.md ファイルの使用”を参照してください。

ファイルまたはフォルダーの名前を変更するには、ファイルまたはフォルダーをクリックして選択し、もう一度クリックして名前を編集可能にします。ファイルまたはフォルダーの名前を変更すると、プ

ロジェクトの XML コードでそのファイルまたはフォルダーへのすべての参照が更新されます。model.xml ファイルおよび特定の他のファイルとフォルダーの名前を変更することはできません。

プロジェクトパッケージに含まれるフォルダーの使用

通常、ルートフォルダーには、**home** と **Output** の 2 つのフォルダーが含まれます。状況によっては、**temp** というフォルダーが存在することもあります。

表 1　ホームフォルダーに含まれるフォルダー

フォルダー	説明
analytics	このフォルダーは、分析ストアなど、分析ファイルの格納に使用します。 SAS Model Manager からモデルをインポートする場合、関連するファイルがこのフォルダーに保存されています。たとえば、DS2 モデルをインポートする場合、SAS Micro Analytic Service モジュールは**/home/analytics/mas-modules/** ***moduleName*** フォルダーに保存されます。Python モデルをインポートすると、モデルファイルは**/home/analytics/model_manager** フォルダーに保存されます。詳細については、“SAS Model Manager の統合の概要”を参照してください。
custom_windows	プロジェクトにカスタムウィンドウを追加してそのプロジェクトを保存すると、カスタムウィンドウの構成ファイルがこのフォルダーに保存されます。詳細については、“プロジェクトにカスタムウィンドウを追加”を参照してください。
files	このフォルダーは、プロジェクトで必要とされる他のファイルを保存するのに使用します。
img	このフォルダーは、プロジェクトの **README.md** ファイルによって参照される画像などの画像ファイルを保存するために使用します。詳細については、“README.md ファイルの使用”を参照してください。
meta	このフォルダーは、他の SAS Event Stream Processing 機能で使用できます。 プロジェクトが仮想環境を使用することを指定してプロジェクトを保存すると、その仮想環境の構成を指定する **requirements.txt** ファイルがこのフォルダーに保存されます。詳細については、“プロジェクトの作成”を参照してください。
model	プロジェクトパッケージを保存すると、SAS Event Stream Processing Studio は、プロジェクト XML ファイルをこのフォルダーに保存します。つまり、SAS Event Stream Processing Studio はプロジェクトを内部データベースだけでなく PV にも保存します。PV 上のプロジェクトのファイル名は、常に model.xml です。
test_files	このフォルダーは、プロジェクトのテストに必要とされる入力ファイルの保存に使用します。このフォルダーは、プロダクションのために必要なファイルではなく、一時ファイルを格納することを目的としています。 このフォルダーは、スコアリング構成ファイルの保存にも使用されます。詳細については、“モデルのテスト時にスコアリング API を使用する”を参照してください。

これらのフォルダーに加えて、**ホーム**フォルダーには **README.md** ファイルが含まれています。詳細については、“README.md ファイルの使用”を参照してください。

プロジェクトの実行中は、**ホーム**フォルダーまたはそのサブフォルダーにファイルを書き込むことはできません。**出力**フォルダーにファイルを書き込むことができます。

出力フォルダーは、プロジェクトをテストモードで実行するときに生成されるファイルを保存するために使用されます。このフォルダーは、プロダクションのために必要なファイルではなく、一時ファイルを格納することを目的としています。

プロジェクト内のウィンドウに出力をファイルに書き込むサブスクライバーコネクタが含まれている場合、それらのファイルは **output** フォルダーに保存されます。出力ファイルを選択し、をクリックしてダウンロードします。その後、ファイルを開いて表示することができます。ダウンロードしたファイルの場所は、ブラウザの構成によって異なる場合があります。

表 2　出力フォルダーに含まれるフォルダー

フォルダー	説明
luaroot	このフォルダーは、永続プロパティを使用する場合に関係します。詳細については、“Persistent Properties” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。
scoreing_output	**スコアリング API** タブをテストモードで使用し、生データを含むファイルが返されると、生データファイルは **scoring_output** フォルダーに保存されます。詳細については、“モデルのテスト時にスコアリング API を使用する”を参照してください。

ファイルのアップロード時にプロジェクトパッケージにエラーが発生すると、**temp** という名前のフォルダーが**プロジェクトパッケージ**ペインのファイルリストの最下部に表示される場合があります。**temp** フォルダーが存在すると、システム管理者にファイルの削除を依頼することなく、一時ファイルを確認し、必要に応じて削除できます。**temp** フォルダ内のすべてのファイルを削除するには、ツールバーのをクリックします。それから、**temp フォルダーをクリア**を選択します。

README.md ファイルの使用

プロジェクトパッケージを作成すると、パッケージの**ホーム**フォルダーに **README.md** ファイルが含まれます。**README.md** ファイルを含まない既存のプロジェクトパッケージを開くと、**README.md** ファイルが追加されます。**README.md** ファイルを使用して、プロジェクトに関するメモ、自身の参照用、他のユーザーの利益のために使用できます。

README.md ファイルを開いて表示すると、その Markdown コードが HTML としてレンダリングされます。**編集**をクリックして、ファイルのマークダウンコードを編集します。

README.md ファイルには、独自のテキストに置き換えることができるサンプルテキストが含まれています。このファイルは、**ホーム**フォルダー内の **img** フォルダーにある **example.png** という名前の画像を参照します。独自のイメージを **img** フォルダーに追加し、 **README.md** ファイルからそれらのイメージを参照できます。

README.md ファイルを削除すると、サンプルテキストを含む **README.md** ファイルを作成できます。**Project Package** ペインのをクリックし、 **"README.md"ファイルの作成**を選択します。

サンプルプロジェクトをインストールすると、プロジェクトパッケージ内の **README.md** ファイルに、プロジェクトの使用方法に関する情報が含まれます。詳細については、“サンプルプロジェクトのインストール”を参照してください。

プロジェクトパッケージに含まれるフォルダーの参照ファイル

同じプロジェクトパッケージ内のファイルを参照する最も簡単な方法は、ファイルパスを手動で入力するのではなく、ファイルに移動することです。ユーザーインターフェイスで利用可能な場合は常に、この機能を使用してください。ファイルへのナビゲートは、パスを手入力するよりもミスが起こりにくいです。パスを手動で入力する場合、誤って、プロジェクトパッケージ外にあるファイルを参照してしまう場合もあります。

パスを手動で入力して同じプロジェクトパッケージ内のファイルを参照する必要がある場合は、プロジェクトの XML コードで ESP_PROJECT_HOME 環境変数を使用して、プロジェクトパッケージの **home** フォルダーを参照できます。たとえば、**analytics** フォルダーにある **model.astore** というファイルを**@ESP_PROJECT_HOME@/analytics/model.astore** のように参照できます。同様に、ESP_PROJECT_OUTPUT 環境変数を使用して、プロジェクトパッケージの **output** フォルダーを参照することができます。パスの中で環境変数を使用することは、PV 上のファイルのパスを知る必要がないことを意味します。

別の例を次に示します。**/mnt/data/mydata.csv** などのファイルパスを持つファイルおよびソケットコネクタは、参照される CSV ファイルがプロジェクトパッケージの外部にあるため機能しません。ただし、CSV ファイルをプロジェクトパッケージの **test_files** フォルダーに配置し、**@ESP_PROJECT_HOME@/test_files/mydata.csv** として参照することは機能します。

重要

- ESP_PROJECT_HOME および ESP_PROJECT_OUTPUT 環境変数を使用して、プロジェクトの XML コードの外部からプロジェクトパッケージ内の場所を参照することはできません。
- プロジェクトがプロジェクトパッケージの一部である場合、プロジェクトはプロジェクトパッケージの外部にあるファイルを参照できません。そうしないと、プロジェクトが機能しない可能性があります。
- SAS Event Stream Processing Studio または SAS Event Stream マネージャーを使用してプロジェクトを実行すると、そのプロジェクトの Kubernetes ポッドは、プロジェクトパッケージの外部にあるファイルにアクセスできないため、プロジェクトが実行されない可能性があります。
- プロジェクトから作成したプロジェクトコンテナーが機能しない可能性があります。プロジェクトコンテナーにビルドされるファイルには、プロジェクトパッケージ内のファイルと、プロジェクト用に選択された仮想環境に関連するファイルが含まれます。SAS Event Stream Processing Studio のテストモードでプロジェクトを実行する場合、これらのファイルのみが使用されます。これらのファイルのみを使用すると、テストモードで使用されるコンテンツがプロジェクトコンテナーに組み込まれているコンテンツと同じになるようになります。

プロジェクトパッケージのインポートまたはエクスポート

プロジェクトパッケージのインポートの詳細については、“プロジェクトのインポート”を参照してください。プロジェクトパッケージのエクスポートの詳細については、“プロジェクトのエクスポート”を参照してください。

プロジェクトパッケージの解凍

プロジェクトパッケージの一部になっているプロジェクトを解凍することはできません。ただし、次の手順で完了することができます。

1 プロジェクト XML ファイルをローカルファイルシステムにエクスポートします。詳細については、“プロジェクトのエクスポート”を参照してください。

2 プロジェクトパッケージに含まれているファイルを保持しておきたい場合には、それらのファイルをローカルファイルシステムにダウンロードします。**プロジェクトパッケージ**ペインの  ボタンを使用します。

3 SAS Event Stream Processing Studio からプロジェクトを削除します。プロジェクトパッケージ全体が削除されます。詳細については、“プロジェクトの削除”を参照してください。

4 SAS Event Stream Processing Studio にプロジェクト XML ファイルを再度インポートします。詳細については、“プロジェクトのインポート”を参照してください。

5 プロジェクトの XML コードの中のファイルパスに、ESP_PROJECT_HOME および ESP_PROJECT_OUTPUT 環境変数の参照が含まれている場合があります。そのような参照がある場合は、必要に応じて調整してください。

プロジェクトパッケージの一部になっているプロジェクトの XML コードには、それを、プロジェクトパッケージの一部となっているプロジェクトとして指定する要素または属性が含まれていません。

プロジェクトパッケージの名前の変更

プロジェクトパッケージの一部になっているプロジェクトの名前を変更することはできません。ただし、プロジェクトパッケージを別の名前で保存できます。

1 そのプロジェクトのコピーを異なる名前で保存する場合は、  をクリックします。コピーされたプロジェクトには、プロジェクトパッケージが含まれていますつまり、コピーされたプロジェクトには、元のプロジェクトパッケージのすべてのファイルとフォルダ—が含まれています。

2 元のプロジェクトを削除します。

SAS Event Stream Processing Studio Modeler の使用

SAS Event Stream Processing Studio Modeler の使用

SAS Event Stream Processing Studio Modeler を使用すると、SAS Event Stream Processing モデルを構築、変更およびテストすることができます。 *モデル*は、SAS Event Stream Processing が入力イベントストリームを分析して意味のある結果に変換する方法を指定します。SAS Event Stream Processing Modeler から、指定されたサブジェクトのコンテキスト依存のヘルプがあれば、それにアクセスできます。これを行うには、アイコンがユーザーインターフェイスに 表示されている場合は ⓘ をクリックします。

SAS Event Stream Processing Studio Modeler は、新しいプロジェクトを作成したり既存のプロジェクトを開いたときに表示されます。

図 9　SAS Event Stream Processing Modeler

ツールバーの**プロジェクトタイプ**フィールドは、プロジェクトのタイプを示します。

- **プロジェクトパッケージ**は、プロジェクト XML ファイルがプロジェクトパッケージの一部であることを意味します。詳細については、“プロジェクトパッケージの操作”を参照してください。
- **LFS を有効にしたプロジェクトパッケージ**は、プロジェクトパッケージ内の大きなファイルを Git Large File Storage に保存できることを意味します。詳細については、“ファイルベースのバージョン管理と Git Large File Storage”を参照してください。

- **XML ファイル**は、プロジェクト XML ファイルがプロジェクトパッケージの一部ではないことを意味します。

ツールバーには、プロジェクトのアクセスレベルに関する情報が表示されます。

- ツールバーに次のボタンが表示されている場合、そのプロジェクトは現在プライベートです。.
- 次のボタンが表示されている場合、そのプロジェクトは現在パブリックです。.

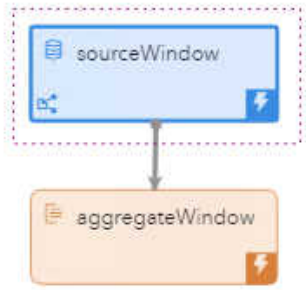
プロジェクトの所有者または SASAdministrators グループのメンバーであれば、ボタンをクリックしてプロジェクトのアクセスレベルを変更することができます。詳細については、“プロジェクトのアクセス制御”を参照してください。

次のボタンを使用して、モデルの倍率とレイアウトを調整します。

- をクリックすると、拡大率を上げます。
- をクリックすると、拡大率を下げます。
- モデル全体がワークスペースに 表示されるようにモデルの倍率を調整するにはをクリックします。
- をクリックし、ワークスペース 内のウィンドウのレイアウトを自動的に調整します。

ウィンドウを選択するには、ワークスペースでウィンドウをクリックします。選択したウィンドウが破線の境界ボックスでワークスペースに 表示されます。

図 10　ワークスペースで 選択されたウィンドウの例



複数のウィンドウを選択するには、ワークスペースをクリックし、選択するウィンドウの周りに破線の境界ボックスをドラッグします。

モデルをパンするには、Shift キーを押しながらワークスペースの 任意の場所をクリックして、カーソルを適切な方向にドラッグします。

ESP Studio Moder は は、一度に 1 つの連続クエリを表示します。新しいモデルを作成すると、デフォルトで cq1 という名前の連続クエリが作成されます。モデルを構築するには、少なくとも 1 つの連続クエリを構成する必要があります。連続クエリの構成の詳細については、“SAS Event Stream Processing Studio での連続クエリの構成”を参照してください。

モデルの全部または一部をコピーするには、ワークスペースで 1 つ以上のウィンドウを選択して Ctrl + C キーを押します。コピーしたモデルエレメントを貼り付けるには、ワークスペースにカーソルを置いて Ctrl + V キーを押します。コピーされたウィンドウがプロパティフィールドとスキーマフィールドを保持します。エッジの接続ウィンドウもコピーすることを条件に、エッジをコピーできます。1 つ以上のプロジェクト間の連続クエリ間で、モデルの全部または一部をコピーアンドペーストすることもできます。モデルの全部または一部を別々のプロジェクト内の連続クエリ間でコピーする場合、コネクタオーケストレーションなどのプロジェクトレベルの構成はコピーされません。ソースプ

ロジェクトの構成を複製するには、ターゲットプロジェクトのプロジェクトレベルの構成を手動で更新する必要があります。

モデルの ESP サーバーを構成

ESP サーバーを作成するには、Kubernetes クラスター内で作成する方法と、エッジサーバー上で作成する方法の 2 つがあります。

Kubernetes クラスターで実行される ESP サーバー

SAS Event Stream Processing Studio が Kubernetes で実行されている場合、Kubernetes クラスターでプロジェクトをロードして実行できます。プロジェクトが Kubernetes クラスターで実行されると、ESP サーバーがオンデマンドで作成されます。プロジェクトが停止すると、ESP サーバーは Kubernetes クラスターから削除されます。作成された ESP サーバーのプロパティは編集できません。それ以外の場合、Kubernetes クラスター内の ESP サーバーは、エッジサーバーにある ESP サーバーと同じように動作します。詳細については、“SAS Event Stream Processing Studio での Kubernetes クラスターの操作”を参照してください。

プロジェクトを開くと、**ESP サーバー**ドロップダウンリストには、プロジェクトに関連付けられている ESP サーバーが表示されます。Kubernetes クラスター内の ESP サーバーは、ドロップダウンリストで次のアイコンで識別されます。次に例を示します。

ESP サーバーの管理の詳細については、“SAS Event Stream Processing Studio での ESP サーバーの管理”を参照してください。

エッジサーバーで実行される ESP サーバー

エッジサーバー上で実行する ESP サーバーの場合、エッジサーバーは Kubernetes クラスターの一部を形成しません。そのため、ESP サーバーの詳細を手動で追加する必要があります。詳細については、“ESP サーバーを追加または編集してエッジサーバー上で実行する”を参照してください。使用できる ESP サーバーが **ESP サーバー**ページに表示されます。

初めてプロジェクトを開くと、**ESP サーバー**ページの表の最初に表示されている ESP サーバーがプロジェクトに割り当てられます。**ESP サーバー**ドロップダウンリストにその ESP サーバーの名前が表示されます。次に例を示します。

ドロップダウンリストを使用して、別の ESP サーバーを割り当てることができます。

ESP サーバーがプロジェクトに割り当てられると、関連付けはブラウザーのローカルストレージに保存されます。つまり、プロジェクトを閉じて再度開くと、同じ ESP サーバーがプロジェクトに関連付けられたままになります。

SAS Event Stream Processing Studio に ESP サーバーが追加されていない場合、プロジェクトで使用できる機能は制限されます。たとえば、ESP サーバーが割り当てられていないプロジェクトでは、コネクタプロパティは使用できません。

ウィンドウの追加

現在表示されている連続クエリにウィンドウを追加する場合、ワークスペースの 左の**ウィンドウ**ペインからウィンドウをドラッグします。

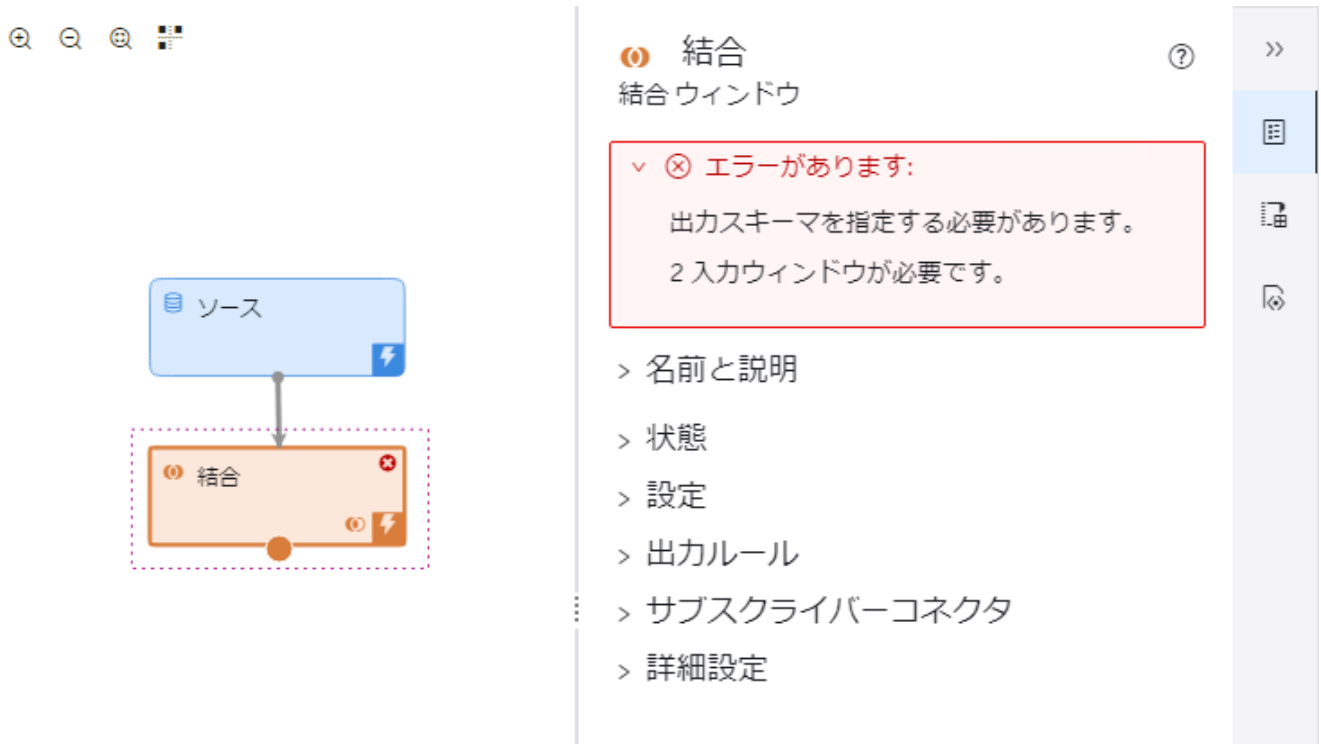
注: また、**ウィンドウ**ペインで関連するウィンドウをダブルクリックして、連続クエリにウィンドウを追加することもできます。

ウィンドウは次のカテゴリに分類されます。

- 入力ストリーム
- 変換
- ユーティリティ
- 分析
- カスタム

ウィンドウごとに有効なプロパティを入力する必要があります。無効なウィンドウプロパティを入力すると、ウィンドウに下記のエラーアイコンが表示されます。結合ウィンドウに無効なプロパティが含まれているモデルの例を次に示します。

図 11　ウィンドウ検証エラーアイコンと 対応するメッセージ



モデルを正常に実行するためにウィンドウをさらに変更する必要がある場合、ウィンドウには警告アイコンが表示されます。 .ウィンドウのプロパティを表示する右ペインに、対応する警告メッセージが表示されます。

ウィンドウの接続

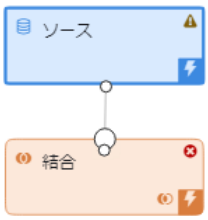
ウィンドウを別のウィンドウにエッジを使用して接続するには、次の操作を実行します。

1 カーソルをウィンドウの下部にあるアンカーポイントの上に置くと、アンカーポイントの色が白に変わります。

図 12　アンカーポイント上のカーソル

2 白いアンカーポイントをクリックし、マウスの左ボタンを押したまま、別のウィンドウのアンカーポイントに線を引きます。

図 13　2 つのウィンドウを接続するエッジ



エッジが自動的にウィンドウに接続します。

注: あるウィンドウから別のウィンドウにエッジを移動することによって接続を変更することはできません。代わりに、新しいエッジを作成して新しい接続を確立する必要があります。エラーのある 2 つのウィンドウ間の接続を作成した場合は、エッジを削除する必要があります。これを行うには、削除するエッジを選択し、**Delete** キーを押します。

エッジ表示タイプ

接続エッジは、ウィンドウ間の接続のタイプに応じて異なる表示にすることができます。

エッジ表示タイプの詳細については、次の表を参照してください。

エッジ表示タイプ	接続エッジ	詳細
イベントストリーム		イベントストリームエッジは、イベントストリームデータを含む入力ウィンドウを接続します。
サポートデータ		サポートデータエッジでは、ジオメトリデータを含む入力ウィンドウが接続されます(つまり、エッジの役割が Geometry に設定されます)。セカンダリ入力ウィンドウを結合ウィンドウに接続することもできます。
非データエッジ		非データエッジでは、イベントストリームデータを含まないウィンドウが接続されます(つまり、エッジの役割が Model または Request に設定されます)。
イベント転送		イベント転送エッジは、Lua ウィンドウまたは Python ウィンドウをソースウィンドウに接続できます。イベント転送エッジが非アクティブな場合、アイコンがエッジに表示されます。詳細については、“ソースウィンドウにイベントを転送”を参照してください。

注: SAS Event Stream Processing Studio Modeler では、無効なエッジが赤い破線で表示されます。

エッジの役割

SAS Event Stream Processing Studio Modeler を使用すると、モデル内の接続エッジの役割を構成できます。ストリーミング分析ウィンドウを接続するエッジ、およびジオフェンスウィンドウと結合ウィンドウを接続するエッジに、エッジの役割を指定する必要があります。各エッジにはデフォルトで役割が割り当てられます。

エッジのデフォルト役割を変更するには、次の操作を実行します。

1 ワークスペースで、役割を変更するエッジを選択します。

2 右ペインの**役割**フィールドで、デフォルトの選択を変更します。

エッジの役割の詳細については、次の表を参照してください。

ウィンドウ名	デフォルトのエッジの役割	使用可能なエッジの役割	依存関係
結合	左テーブル	左テーブルまたは右テーブル	結合ウィンドウの接続エッジにエッジの役割を割り当てると、残りの接続エッジに自動的に代替エッジの役割が割り当てられます。
ジオフェンス	位置	位置またはジオメトリ	ジオフェンスウィンドウの接続エッジにエッジの役割を割り当てると、残りの接続エッジに

ウィンドウ名	デフォルトのエッジの役割	使用可能なエッジの役割	依存関係
			自動的に代替エッジの役割が割り当てられます。
モデルリーダー	リクエスト	リクエスト	適用できません
モデルスーパーバイザ	リクエスト	リクエストまたはモデル	適用できません
計算	データ	データまたはリクエスト	適用できません
学習	データ	データまたはリクエスト	適用できません
スコア	データ	データまたはモデル	接続エッジにエッジの役割を割り当てると、残りの接続エッジに自動的に代替エッジの役割が割り当てられます。

ウィンドウまたはエッジの削除

選択したウィンドウまたはエッジをモデルから削除するには、**Delete** キーを押します。

注: モデルからウィンドウを削除すると、自動的に接続エッジがすべて削除されます。

ウィンドウアイコン

モデルの各ウィンドウには、現在の状態を表すアイコンが表示されます。たとえば、サブスクライバーコネクタを含むソースウィンドウには、  が表示されます。各アイコンの詳細については、次の表を参照してください。

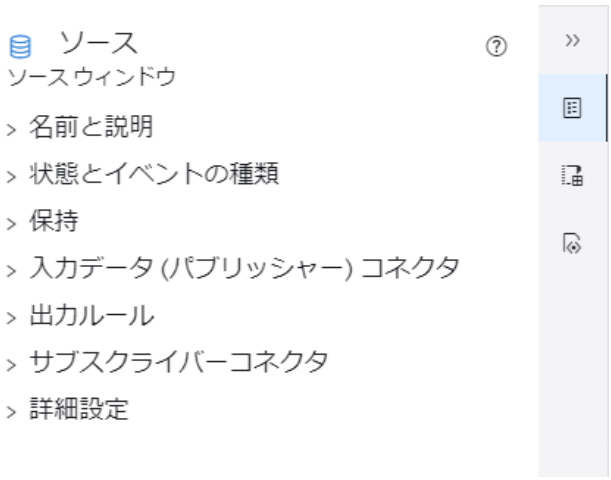
アイコン	説明
	モデルの実行に失敗する原因となるエラーメッセージがウィンドウに表示されることを示します。
	ウィンドウが有効な状態になるためには、ウィンドウをさらに変更する必要があることを示します。

アイコン	説明
	ウィンドウに、メモリに格納されている完全にステートフルなインデックスが含まれていることを示します。 **注:** ウィンドウに非ステートフルインデックスが含まれている場合、このアイコンは表示されません。 このアイコンの色は、それを含むウィンドウによって変わります。この例では、ソースウィンドウにアイコンを表示しています。
	ウィンドウに、ディスクに格納されている完全にステートフルなインデックスが含まれていることを示します。 **注:** ウィンドウに非ステートフルインデックスが含まれている場合、このアイコンは表示されません。 このアイコンの色は、それを含むウィンドウによって変わります。この例では、ソースウィンドウにアイコンを表示しています。
	ソースウィンドウにパブリッシャーコネクタが含まれていることを示します。
	ソースウィンドウにサブスクライバーコネクタが含まれていることを示します。
	結合ウィンドウに内部結合タイプが含まれていることを示します。
	結合ウィンドウに完全外部結合タイプが含まれていることを示します。
	結合ウィンドウに右外部結合タイプが含まれていることを示します。
	結合ウィンドウに左外部結合タイプが含まれていることを示します。
	ウィンドウがスコアリング API によって入力ウィンドウとして使用されることを示します。
	ウィンドウがスコアリング API によって出力ウィンドウとして使用されることを示します。
	このウィンドウがイベントを別の連続クエリ内のウィンドウに転送することを示します。
	このウィンドウが別の連続クエリ内のウィンドウから転送されたイベントを受信することを示します。

ウィンドウのプロパティの構成

ウィンドウのプロパティを設定するには、ワークスペースでウィンドウをクリックします。右ペインにはそのウィンドウのプロパティが表示されます。必要に応じてプロパティを編集します。

図 14　ソースウィンドウのプロパティ



注: 右ペインに XML コードまたは出力スキーマが表示されている場合は、右ペインの をクリックしてウィンドウのプロパティを表示できます。

ウィンドウの出力スキーマの構成

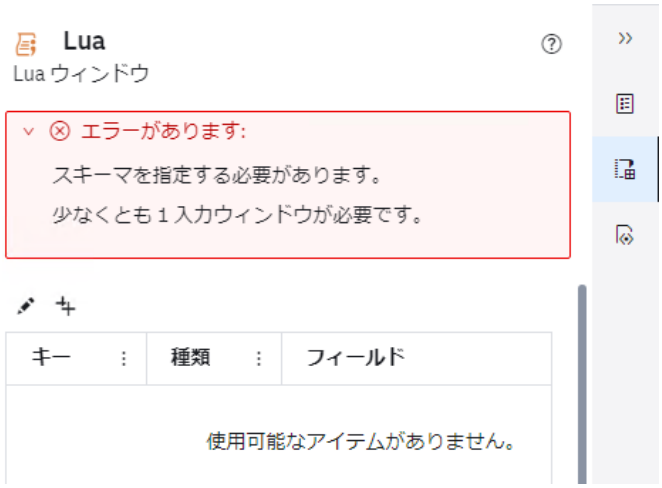
ウィンドウの出力スキーマは、そのウィンドウから次のウィンドウに渡されるデータの構造を定義します。

ウィンドウの出力スキーマは次のように構成します:

1　ワークスペースのウィンドウを 選択します。

2　 をクリックします。これは、右側のツールバーにあります。

図 15　空の出力スキーマの例



3 をクリックします。

4 **出力スキーマ**ウィンドウで、必要に応じて出力スキーマを構成します。

- 新しいスキーマフィールドを追加するか、親ウィンドウから既存のスキーマフィールドをインポートします。
- どのフィールドがキーフィールドであるかを指定します。
- フィールド順序を指定して、出力イベントに表示されるデータの順序を決定します。

5 **OK** をクリックします。

ヒント　親ウィンドウからすべてのスキーマフィールドをインポートする 簡単な方法については、手順 3 の をクリックする代わりに、 をクリックします。 ボタンは、選択したウィンドウの種類で使用できます。

コネクタの構成

コネクタを使用すると、SAS Event Stream Processing との間でイベントをストリーミングできます。パブリッシャーコネクタを使用すると、ソースウィンドウの中へイベントをストリーミングできます。サブスクライバーコネクタを使用すると、ウィンドウから、プロジェクト外の宛先へイベントをストリーミングすることができます。詳細については、*“コネクタを使用してイベントストリームをパブリッシュおよびサブスクライブする” (SAS Event Stream Processing: 概要)*を参照してください。

プロジェクトにイベントを渡す別の方法は、スコアリング API を使用することです。詳細については、*“Using the Scoring API” (SAS Event Stream Processing: Using Source and Derived Windows)*を参照してください。

コネクタの構成

1 プロジェクトを開きます。

2 ワークスペースにおいて 、コネクタを構成するウィンドウを選択します。

3 右ペインで、**入力データ(パブリッシャー)コネクタ** または **サブスクライバーコネクタ**を展開します。

4 をクリックします。

5 **コネクタ構成**ウィンドウを使用して、コネクタのプロパティを設定します。

利用可能なプロパティは、**コネクタタイプ**フィールドで選択するコネクタによって異なります。コネクタの種類の詳細については、*"コネクタとアダプターについて" (SAS Event Stream Processing: コネクタとアダプター)*を参照してください。

6 **OK** をクリックします。

ファイルとソケットコネクタを構成する

CSV ファイルからソースウィンドウにデータをパブリッシュするためのファイルとソケットコネクタの構成方法の例を示します。この手順では、Kubernetes 環境で SAS Event Stream Processing Studio を使用しており、ファイルストレージを有効にするために永続ボリュームが設定されていることを前提としています。これらのステップには、コネクタの日付形式の設定も含まれます。

1 プロジェクトを開きます。

2 ワークスペースで 、パブリッシャーコネクタを構成するソースウィンドウを選択します。

3 右ペインで、**入力データ(パブリッシャー)コネクタ**を展開します。

4 をクリックします。

5 必要に応じて、**コネクタ構成**ウィンドウの**名前**フィールドに、コネクタの名前を調整します。

6 **Fsname** フィールドで、をクリックし、**ファイル選択**ウィンドウで、ソースウィンドウにパブリッシュするイベントを含む CSV ファイルを選択します。ファイルを選択する際に検討すべきことをいくつか示します。

- ファイルへのナビゲートは、パスを手入力するよりもミスが起こりにくいです。
- Kubernetes 環境では、ファイルのベースロケーションは、常に**/mnt/data** です。ただし、入力ファイルが存在する場所は、ファイルストレージの設定によって異なります。サポートが必要な場合は、システム管理者に連絡してください。
- そのプロジェクトがプロジェクトパッケージの一部である場合は、既存のファイルを選択する場合も、またコンピュータのファイルをプロジェクトパッケージにアップロード (つまり、ファイルを PV にアップロード) する場合も**ファイル選択**ウィンドウを使用できます。詳細については、*"プロジェクトパッケージの操作"*を参照してください。

7 CSV ファイルを選択したため、 **Fstype** フィールドが自動的に **csv** に設定されることを確認します。

8 **すべてのプロパティ** をクリックします。

9 **すべてのプロパティ** ウィンドウで、dateformat 行を見つけて、値の列に日付形式を入力します。次に、日付形式の例を示します。%d/%b/%Y:%H:%M:%S.適切な日付形式は、CSV ファイル内のデータの形式によって異なります。

注: CSV ファイル内のイベントに演算コードとフラグが含まれていない場合、addcsvopcode プロパティは自動的に true に設定され、addcsvflags プロパティは自動的に normal に設定されます。CSV ファイルの最初の行にヘッダーが含まれている場合、header プロパティは自動的に 1 に設定されます。

10 **OK** をクリックして、**コネクタ構成**ウィンドウに戻ります。

11 **OK** をクリックします。

ヒント　または、ファイルとソケットパブリッシャーコネクタを含むソースウィンドウをすばやく作成するには、をクリックし、**プロジェクトパッケージ**ペインからワークスペースの空の部分にファイルをドラッグします。代わりに、ファイルを既存のソースウィンドウにドラッグすると、そのウィンドウにコネクタが追加されます。ソースウィンドウをさらに調整できます。

- コネクタのデフォルトのプロパティを調整する場合は、右ペインで**入力データ (パブリッシャー)コネクタ**を展開し、コネクタをダブルクリックして編集します。
- 使用したファイルが CSV ファイルの場合、そのファイルからのフィールドがウィンドウの出力スキーマに追加されます。出力スキーマを編集してキーフィールドを選択する必要があり、出力スキーマにその他の調整を行うことができます。詳細については、"ウィンドウの出力スキーマの構成"を参照してください。

ファイルとソケットコネクタの詳細については、"ファイルとソケットコネクタとアダプターの使用" (*SAS Event Stream Processing: コネクタとアダプター*)を参照してください。

ファイルおよびソケットコネクタを使用してパブリッシャーコネクタを構成する場合、SAS Event Stream Processing Studio プロジェクトを実行するユーザーには入力ファイルへの読み込みアクセス権が必要です。ファイルおよびソケットコネクタを使用してサブスクライバーコネクタを構成する場合、SAS Event Stream Processing Studio ユーザーはファイルを書き込むディレクトリへの書き込みアクセス権を持っている必要があります。詳細については、"ファイルアクセスについて" (*SAS Event Stream Processing: トラブルシューティング*)を参照してください。プロジェクトパッケージを使用することで、入力ファイルおよび出力ファイルの場所にアクセスしやすくなります。詳細については、"プロジェクトパッケージ"を参照してください。

ヒント　CSV ファイルからソースウィンドウにイベントをストリームするのに、ファイルとソケットコネクタではなく、WebSocket を使用することができます。こちらの方がプロジェクトのテストに適している場合もあります。詳細については、"CSV ファイルからイベントをパブリッシュする"を参照してください。

コネクタのパスワードを暗号化する

一部のコネクタやアダプターでは、接続の構成詳細にパスワードを入力する必要があります。SAS Event Stream Processing Studio は、コネクタとアダプター内のパスワードを暗号化できるようにします。

- コネクタ内のパスワードを暗号化するには、**コネクタ構成**ウィンドウを使用します。詳細については、次を参照してください。

- アダプター内のパスワードを暗号化するには、**設定**ページでパスワード暗号化ツールを使用します。詳細については、"アダプター内のパスワードを暗号化する"を参照してください。

次に、 **コネクタ構成**ウィンドウを使用して MQTT コネクタを追加し、MQTT パスワードを暗号化する例を示します。

1 プロジェクトを開きます。

2 ワークスペースで、MQTT コネクタを構成するソースウィンドウを選択します。

3 右ペインで、**入力データ(パブリッシャー)コネクタ**を展開します。

4 をクリックします。

5 必要に応じて、**コネクタ構成**ウィンドウの**名前**フィールドに、コネクタの名前を調整します。

6 **コネクタータイプ**フィールドで、 **MQTT** を選択します。

7 **パスワードを暗号化**をクリックします。

8 **パスワードの暗号化**ウィンドウで、ユーザー ID を入力します。

9 暗号化するパスワードを入力し、**パスワードを暗号化**をクリックします。

10 **コピー**をクリックして、暗号化されたパスワードをクリップボードにコピーします。

11 **閉じる**をクリックします。

12 **コネクタ構成**ウィンドウで、 **すべてのプロパティ** をクリックします。

13 **すべてのプロパティ** ウィンドウで、暗号化されたパスワードを **mqttpassword** フィールドに貼り付けます。

14 **mqttpasswordencrypted** フィールドを **true** に設定し、 **OK** をクリックします。

15 ニーズに合わせて**コネクタ構成**ウィンドウの残りのフィールドに入力し、 **OK** をクリックします。詳細については、"MQTT コネクタの使用" (*SAS Event Stream Processing: コネクタとアダプター*)を参照してください。

別の例を次に示します。データベースコネクタの場合、暗号化されたパスワードを使用してシステムデータソース名(DSN)を作成し、**コネクタの構成**ウィンドウの**接続文字列**フィールドに入力します。

DSN=Oracle Wire Protocol;uid=exampleuserid;pwd=pkrrtO8hAlzcgkUUrO5zRhgZjePLR0HgqQYWtyLr5W70yUjT3Wf1Eo72VGd4i6s3;hostname=example_DB_server_name.com;servicename=example.process_name"

詳細については、"コネクタとアダプターについて" (*SAS Event Stream Processing: コネクタとアダプター*)を参照してください。

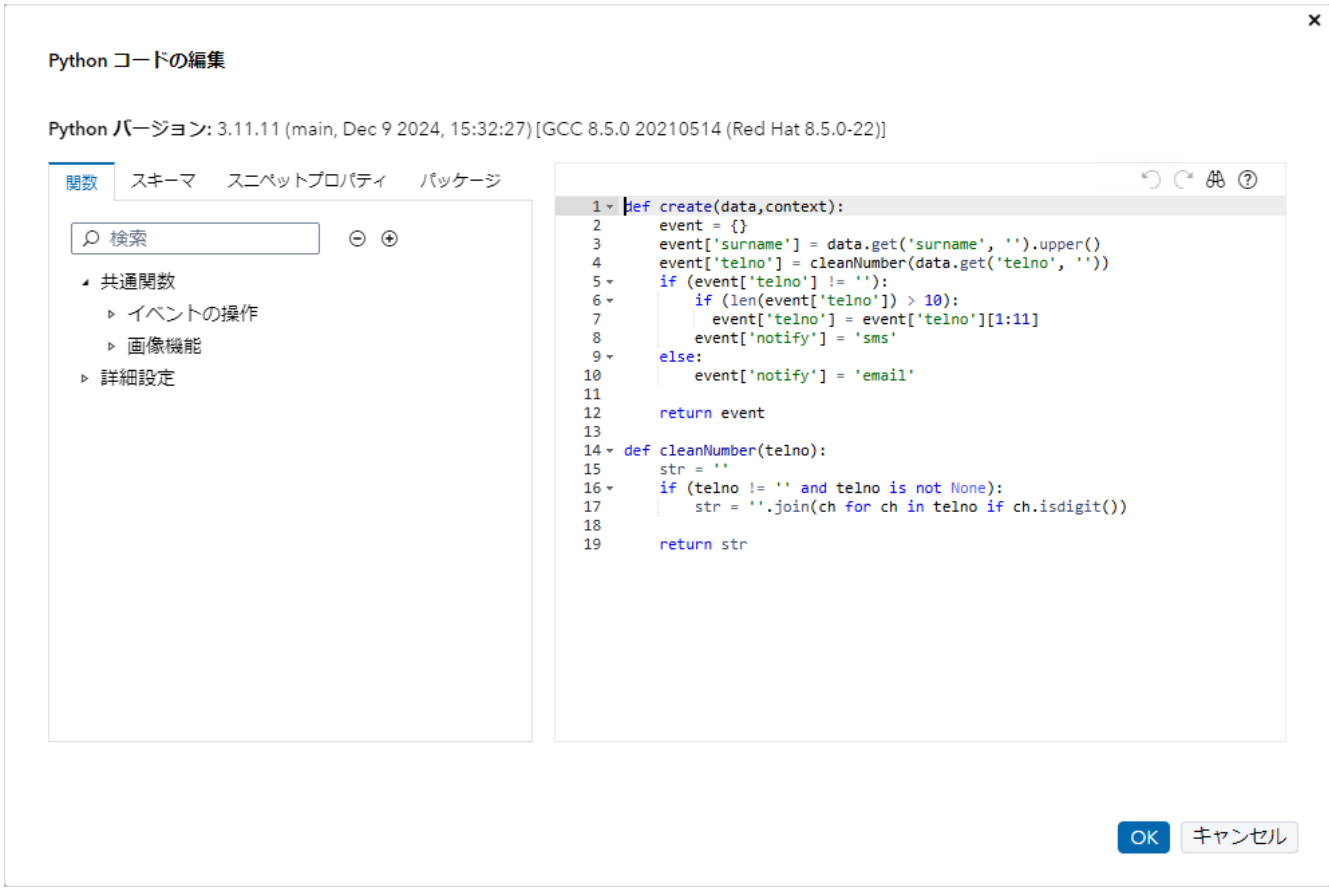
コードと式を編集する

コードと式の編集の概要

大きなコードブロックを入力する場合は、コードと式を編集するために SAS Event Stream Processing Studio で利用可能なエディターを使用することをお勧めします。

場合によっては、該当するフィールドの隣りの $x=\frac{y}{z}$ アイコンをクリックしてエディターにアクセスします。このような場合、エディターウィンドウには、**Lua コードの編集**、Python コードの編集、**式エディター**などの名前が付いていることがよくあります。

図 16　Python コードの編集例



また、エディターが他のウィンドウに埋め込まれている場合もあります。たとえば、Lua ベースのパターンウィンドウで Lua コードを編集する場合、エディターは、**新しいパターン**または**パターンの編集**の中に埋め込まれています。

エディターの機能は、編集しているコードの言語に依存します。エディターには 2 つのペインを含めることができます。

左ペインを使用してコードに項目を追加します

左ペインでは、アイテムを検索してコード追加できます。編集しているコードの言語に応じて、関数、スキーマフィールド、スニペット、テンプレート、イベント、変数などの項目を検索して追加できます。

Python コードを編集すると、左ペインの上に Python バージョンが表示されます。

アイテムを追加するには、右側のペインのそのコードの該当場所にカーソルを移動して、左ペインのそのアイテムをダブルクリックします。

次の情報は、Lua および Python コードの編集に固有です。

- **関数**タブには、Lua または Python コード内で呼び出すことができる SAS Event Stream Processing 関数がリストされます。Lua コードで使用できる関数についての詳細は "Using SAS Event Stream Processing Functions with Lua" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。Python コードから呼び出すことができる関数についての詳細は "Using SAS Event Stream Processing Functions with Python" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。
- **スキーマ**タブには、Lua または Python コードで使用できるフィールドがリストされます。
  - Lua ウィンドウまたは Lua ベースのフィルターウィンドウで Lua コードを編集する場合、または Python ウィンドウまたは Python ベースのフィルターウィンドウで Python コードを編集する場合、**スキーマ**タブには入力ウィンドウのフィールドのサブセットがリストされます。リストには、**Lua コードで使用するフィールド**フィールドまたは **Python コードで使用するフィールド**フィールドの設定が反映されます。
    - **Lua コードで使用するフィールド**フィールドまたは **Python コードで使用するフィールド**フィールドを使用して特定のフィールドを含める場合、それらのフィールドは**スキーマ**タブにリストされます。
    - **Lua コードで使用するフィールド**フィールドまたは **Python コードで使用するフィールド**フィールドを使用して特定のフィールドを除外すると、他のすべてのフィールドが **スキーマ**タブにリストされます。
    - **Lua コードで使用するフィールド**フィールドまたは **Python コードで使用するフィールド**フィールドが空の場合、すべてのフィールドが **スキーマ**タブにリストされます。
  - パターンウィンドウで Lua コードを編集すると、**スキーマ**タブに入力ウィンドウのすべてのフィールドがリストされます。
- **スニペットプロパティ**タブが使用可能な場合、Lua スニペットまたは Python スニペット、およびそれらのスニペットに関連するプロパティがリストされます。左側のペインでプロパティをダブルクリックすると、スニペットコードをコピーするのではなく、そのプロパティに関連するコードが右側のペインに追加されます。

  たとえば、スニペットが 5 分ごとにプロパティを設定すると、右側のペインに追加されたコードは、関連する変数を読み取るためにそのプロパティを取得します。同様に、スニペットが定期的に実行されてトークンを更新し、そのトークンを設定するように設定されている場合、右側のペインに追加されたコードはそのトークンを取得します。Lua コードを使用する場合、Lua ウィンドウまたは Lua スニペットのみがプロパティ値を設定できますが、Lua エンティティはすべてこれらの値を取得できます。Python コードを操作する場合、プロパティ値を設定できるのは Python ウィンドウまたは Python スニペットのみですが、これらの値は任意の Python エンティティで取得できます。

  スニペットのプロパティタブでスニペットの上にマウスを置くと、そのスニペットの説明が表示されます。スニペットを作成するときに説明を入力すると、適切なスニペットを選択するのに役立ちます。Lua スニペットの作成の詳細については、"SAS Event Stream Processing Studio での Lua スニペットの使用"を参照してください。Python スニペットの作成の詳細については、"SAS Event Stream Processing Studio での Python スニペットの操作"を参照してください。
- **パッケージ**タブは、Python コードを編集するときに使用できます。このタブでは、SAS Event Stream Processing 環境で使用できる Python パッケージを簡単に確認できます。パッケージのリストは情報提供のみを目的として提供されています。左側のペインでパッケージを選択して右側のペインに追加することはありません。

右ペインを使用してコードに手動で入力します

右ペインには、コードを手動で入力できるコード領域が含まれています。エディターは、配色を使用して構文を強調表示します。エディターの次のボタンを使用して、コードに対して特定の操作を実行します。

アイコン	説明
	以前の変更を元に戻します。
	取り消し操作の効果を元に戻します。
	エディターでコードを 検証します。 検証の完了後に、コードの有効性を通知するメッセージが表示されます。
	ヘルプへのアクセスを提供します。 例えば、Lua ウィンドウで Lua コードを編集している場合にこのボタンをクリックすると、Lua ウィンドウ内の Lua コードに関する情報のあるヘルプトピックが表示されます。
	コードにエラーが含まれていることを示します。 エディターに、エラーとその場所を示すメッセージが表示されます。エラーの詳細説明については、アイコンの上にカーソルを置いてください。

エディターでは、既存のコード要素を選択して挿入し、部分的に入力したコードを完成させることができます。エディターにコードを部分的に入力すると、自動補完の候補のリストが表示されます。リストから最初の提案を受け入れるには、Tab キーを押します。別の提案の 1 つを受け入れるには、関連する提案が選択されるまで下矢印キーを押してから、Tab を押します。

EEL 式を編集する場合、エディターを使用して、数値、文字列、関数、および他の関数を使用する関数と組み合わせて演算子を含めることができます。エディターには、これらの演算子にアクセスできるツールバーが含まれています。次の例では、ツールバーが赤いボックスで強調表示されています。

図 17　エディターでの EEL 式の例

演算子の説明については、ツールバーの関連する演算子の上にカーソルを置きます。演算子を挿入するには、そのコードの該当場所にカーソルを移動して、左側のペインのその演算子をダブルクリックします。EEL コード編集時に利用可能な演算子の詳細は、*"演算子" (式言語: リファレンスガイド)*を参照してください。

ウィンドウを無効または有効にする

ウィンドウを無効にすると、プロジェクトをデバッグするときに役立ちます。たとえば、メモリの問題や esp.publish()関数を含む複雑なロジックをデバッグする場合、一部のウィンドウを無効にすると、問題の原因となっているウィンドウを特定するのに役立ちます。ウィンドウを削除するのではなく無効にすることは、問題を特定して対処した後、関連するウィンドウをすばやく有効にできることを意味します。

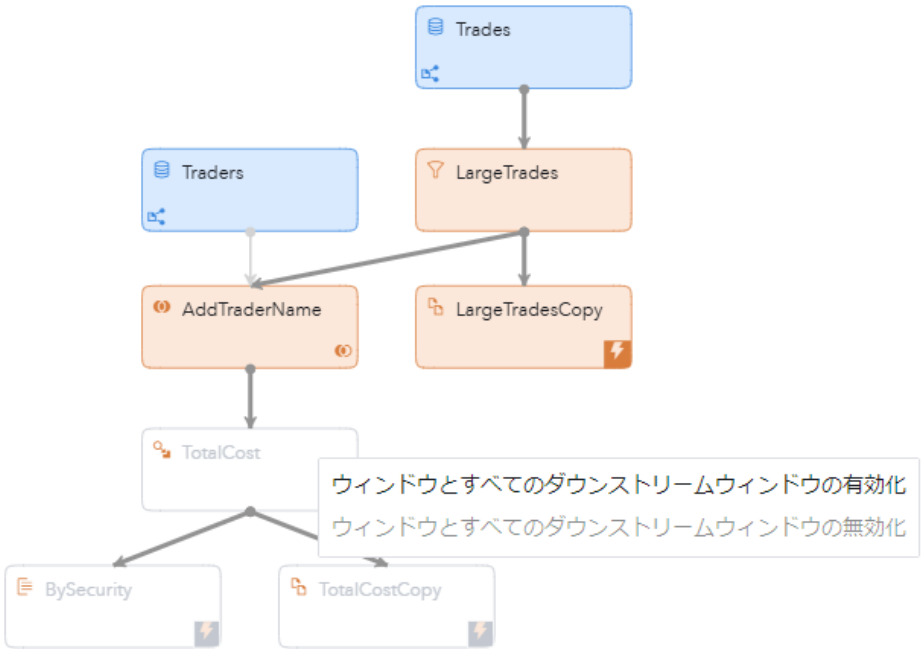
ウィンドウを無効にすると、下位のウィンドウも無効になります。同様に、ウィンドウを有効にすると、下位のウィンドウも有効になります。

プロジェクトを実行すると、無効化されたウィンドウはプロジェクトの一部ではないかのように扱われ、そのコネクタは実行されません。無効化されたウィンドウがコネクタのオーケストレーションに含まれている場合、そのウィンドウのコネクタはターゲットの状態(実行、停止、または終了)に達したとして扱われます。

ウィンドウを無効または有効にするには、SAS Event Stream Processing Studio モデラーワークスペースでウィンドウを右クリックし、**有効ウィンドウとすべての 下位ウィンドウ**または**無効ウィンドウ**

とすべての 下位ウィンドウを選択します。ウィンドウを無効にすると、その色が変わります。たとえば、Light アプリケーションテーマを使用する場合、無効化されたウィンドウは白に変わります。

図 18　ウィンドウの無効化と有効化の例



XML エディタの使い方

XML エディタの使い方

SAS Event Stream Processing Studio Modeler には XML エディタが含まれています。SAS Event Stream Processing StudioModeler の視覚的なモデリング機能と比較して、モデルを作成する代替方法として使用できます。ワークスペースには、モデルの XML コードのスナップショットが表示されます。XML エディタを使用してウィンドウの名前を変更できます。これを行うには、ワークスペースで名前を変更するウィンドウを選択し、XML エディタでウィンドウの名前を変更します。

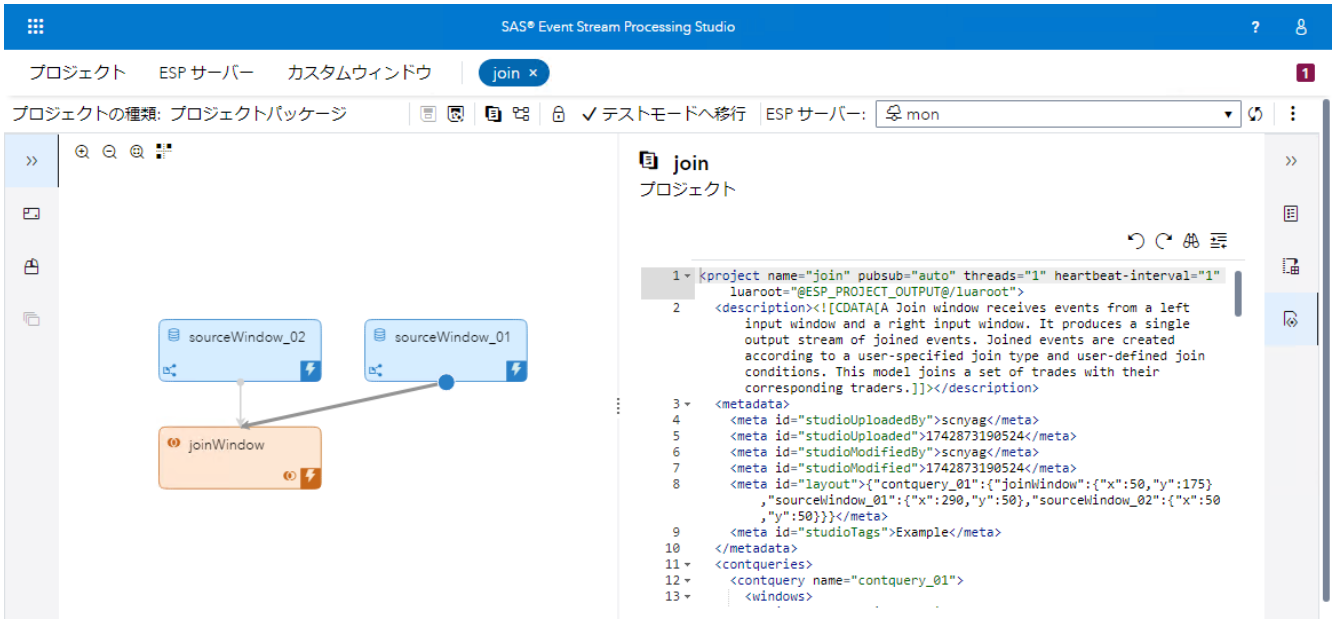
注意
XML エディタを使用してモデルの XML コードを手動で編集すると、モデルが無効になる可能性があります。 SAS Event Stream Processing StudioModeler を使用してモデルを構築すると、モデルに無効な XML コードが含まれる可能性が制限されます。プロパティペインに戻る前に、XML エディタで無効な XML を修正する必要があります。 XML エディタで手動で行った変更は、必ずしもワークスペースに 反映されません。 最初に選択せずに XML エディタを使用してウィンドウの名前を変更すると、ウィンドウがワークスペースのデフォルトの位置にある新しいウィンドウに置き換えられます。これにより、モデルが無効にな

ります。冗長ウィンドウとの接続は、削除してからワークスペースで 再作成する必要があります。また、XML エディタでエッジを手動で編集することもできます。

XML エディタを開くには、プロジェクトを開いて、右側のツールバーで をクリックします。

右ペインに XML エディタが表示されます。

図 19　XML エディタ



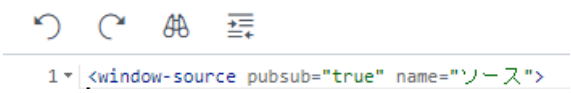
ワークスペース内の特定のエレメントを選択すると、XML エディタがリロードされ、XML コードの対応するセクションのみが表示されます。プロジェクトの XML 全体を再度表示するには、 をクリックします 。

特定の ESP ウィンドウを選択した場合、ワークスペース内の空白領域をクリックすると、XML エディタがリロードされ、モデルの連続クエリに関連する XML が表示されます。

関連する XML 要素に囲まれていない XML コードにコメントを追加した場合、そのコメントは自動的に XML 要素内に移動します。これは、XML コードが並べ替えられたときに発生します。たとえば、モデルを保存した場合、または XML エディタから移動した場合は、XML コードが並べ替えられます。

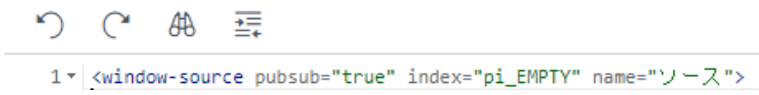
モデルが作成されると、オプションの属性は XML コードに含まれません。モデルには、モデルの XML コードで直接指定されていない設定も含まれていますが、それらはユーザーインターフェイスに 表示されます。たとえば、ソースウィンドウのデフォルトのウィンドウ状態が**(クエリから継承) pi_HASH** の場合、暗黙の設定は XML コードには表示されません。次の例を参照してください。

図 20　ウィンドウのデフォルト状態なしでソースウィンドウの XML コードを表示する XML エディタ



ウィンドウの状態をデフォルト値から変更するには、次のようにモデルの XML コードの属性が含まれます。

図 21　ウィンドウのデフォルト状態でソースウィンドウの XML コードを表示する XML エディタ



一意識別プロジェクト情報は、プロジェクトの XML コードの<metadata>エレメント内に表示されます。メタデータはプロジェクトの XML コードに表示されますが、SAS Event Stream Processing Studio データベースにあります。詳細については、“プロジェクトメタデータ”を参照してください。

編集ツールとキーボードショートカットの使用

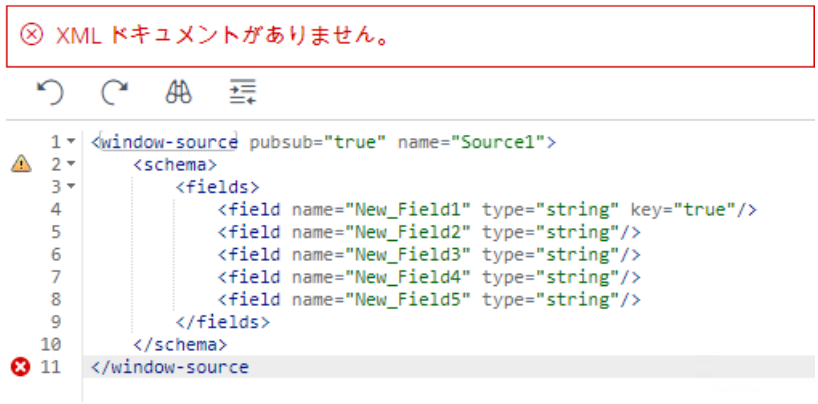
XML エディタには、編集ツールを含むツールバーが含まれています。これらのツールは、キーボードショートカットを使用してアクセスすることもできます。

アイコン	アクション	キーボードショートカット
	以前の変更を元に戻します。	Ctrl + Z
	取り消し操作の効果を元に戻します。	Ctrl + Y
	特定のテキストの検索を促すプロンプトが表示されます。もう一度 Ctrl+F を押すと、検索したテキストの置換を促すプロンプトが表示されます。	Ctrl+F
	手動で入力した XML コードをフォーマットします。	使用不可
アイコンなし	選択したコードを元の位置から削除します。	Ctrl+X
アイコンなし	選択したコードをクリップボードにコピーします。	Ctrl+C
アイコンなし	クリップボードのコードをカーソルの位置に貼り付けます。	Ctrl+V
アイコンなし	XML エディタですべてのコードを 選択します。	Ctrl+A

検証

XML エディタは、入力したコードの構文を自動的に検証します。無効なコードを入力すると、XML エディタに次のエラーメッセージが表示されます。

図 22　無効な XML の警告



XML エラーメッセージは、ブレッドクラムトレイルでエラーの場所も示します。ブレッドクラムトレイルの各セクションは、XML コード内の XML タグを表します。XML コードの最上位 XML タグは、ブレッドクラムトレイルには含まれていません。

XML コードのエラーの一般的な説明を表示するには、⚠警告アイコンの上にカーソルを置きます。⊗アイコンは、XML コード内のエラーの場所も表示します。

エラーの詳細説明については、⊗アイコンの上にカーソルを置いてください。

効率化のヒント

ウィンドウの種類によっては、モデル内に同じフィールドを使用するウィンドウがある場合は、ウィンドウ間でスキーマフィールドをコピーできます。**出力スキーマ**ウィンドウで↘をクリックし、**入力スキーマからフィールドをコピー**ウィンドウを開きます。コピーするスキーマフィールドを選択し、**OK** をクリックします。また、XML エディタを使用して、ウィンドウ間にフィールドをコピーして貼り付けることもできます。

注: この機能は、別のウィンドウからスキーマフィールドをコピーすることが適切でないウィンドウタイプでは使用できません。たとえば、暗黙的または内部的に生成されたスキーマを含むウィンドウとの間でスキーマフィールドのコピーはできません。

連続クエリ

連続クエリの追加

モデルに新しい連続クエリを追加するには、ツールバーの⋮をクリックし、**連続クエリの追加**を選択します。**連続クエリの削除**を選択して、モデルから連続クエリを削除することもできます。

モデル内の各連続クエリを切り替えるには、ツールバーの**連続クエリ**ドロップダウンリストから表示する連続クエリを選択します。

SAS Event Stream Processing Studio での連続クエリの構成

連続クエリにより、SAS Event Stream Processing はデータを分析して操作できます。*連続クエリ*は、リアルタイムでデータに対して自動的かつ定期的に実行されるクエリです。

モデルは少なくとも 1 つの連続クエリを含む必要があります。SAS Event Stream Processing Studio Modeler は、デフォルトで連続クエリ cq1 を作成します。この連続クエリ内にウィンドウを追加して構成することができます。モデルには多くの連続クエリを含めることができます。

連続クエリには、少なくとも 1 つのソースウィンドウが含まれている必要があります。ソースウィンドウは、1 つまたは複数の派生ウィンドウ(たとえば、パターンウィンドウまたは結合ウィンドウ)に接続します。ソースウィンドウを作成したら、モデルに派生ウィンドウを追加できます。

連続クエリのプロパティの構成

1 **プロジェクト**ページで、構成する連続クエリを含むプロジェクトを右クリックし、**プロジェクトを開く**を選択します。

   SAS Event Stream Processing Studio Modeler が表示されます。右ペインにプロジェクトのプロパティが表示されます。

2 ツールバーの  をクリックします。

   右ペインには、プロジェクトが最後に保存されたときに選択した連続クエリのプロパティが表示されます。

3 必要に応じてフィールドを構成します。

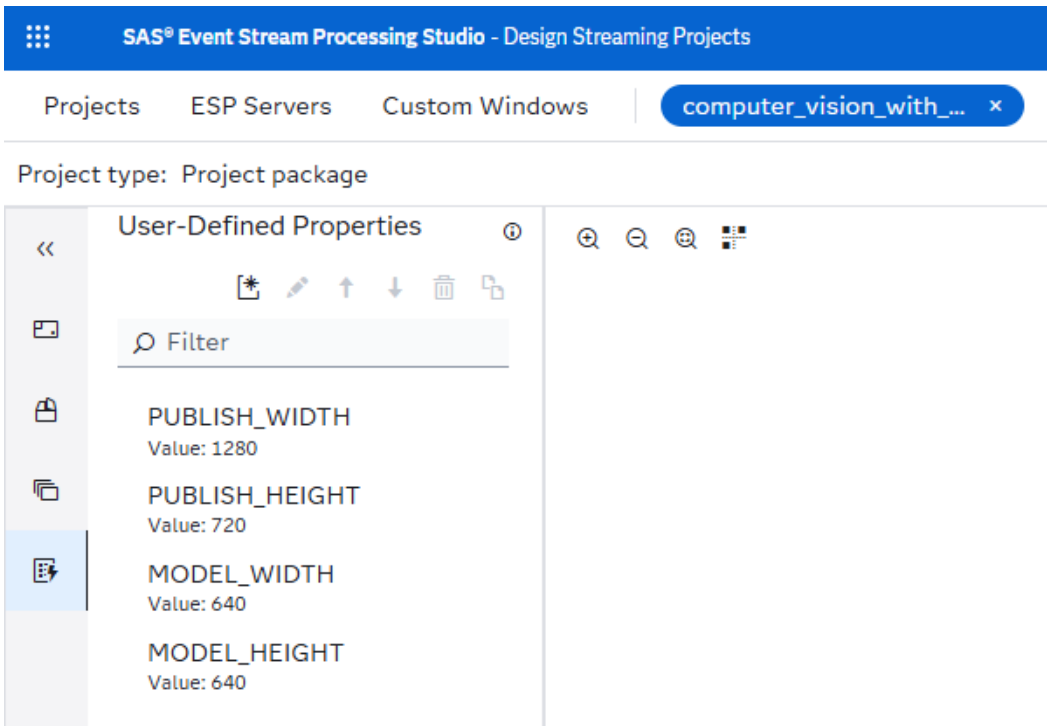
ユーザー定義プロパティの操作

ユーザー定義プロパティの概要

ユーザー定義プロパティを使用すると、同じ値を繰り返し入力するかわりに、特定の値を一度に定義してプロジェクト全体で再利用できます。ユーザー定義プロパティは、複雑なプロジェクト内の一貫性を維持するのに役立ちます。たとえば、ウィンドウ状態、環境変数、またはファイルパスのプロパティを定義します。その後、プロジェクト内の値が必要な場所で、そのプロパティを参照できます。

ユーザー定義プロパティは、SAS Event Stream Processing Studio モデラーの**ユーザー定義プロパティ**ペインで管理されます。次の図は、**ONNX を使用したコンピューター Vision の例のユーザー定義プロパティ**ペインがどのように見えるかを示したものです。

図 23　ユーザー定義プロパティ



この例では、ユーザー定義プロパティを使用して、画像のサイズを変更して、パブリッシュされた画像フレームの解像度を定義します。完全な例にアクセスする方法の詳細については、“サンプルプロジェクトのインストール”を参照してください。

ユーザー定義プロパティの追加

1 プロジェクトを開きます。

2 をクリックします。

3 **ユーザー定義プロパティ**ペインで、をクリックします。

4 **プロパティ**ウィンドウを使用してユーザー定義プロパティを指定し、**OK** をクリックします。

プロパティが**ユーザー定義プロパティ**ペインに表示されます。および を使用して、プロパティのリスト順序を変更できます。この機能は、プロジェクトに多数のユーザー定義プロパティが含まれている場合に便利です。ただし、この順序は、プロジェクトを実行する際のユーザー定義プロパティの動作には影響しません。

ユーザー定義プロパティの参照

ユーザー定義プロパティを追加した後、**ユーザー定義プロパティ**ペインからプロパティをコピーして目的の場所に貼り付けることで、同じプロジェクト内の別の場所からそのユーザー定義プロパティを参照できます。

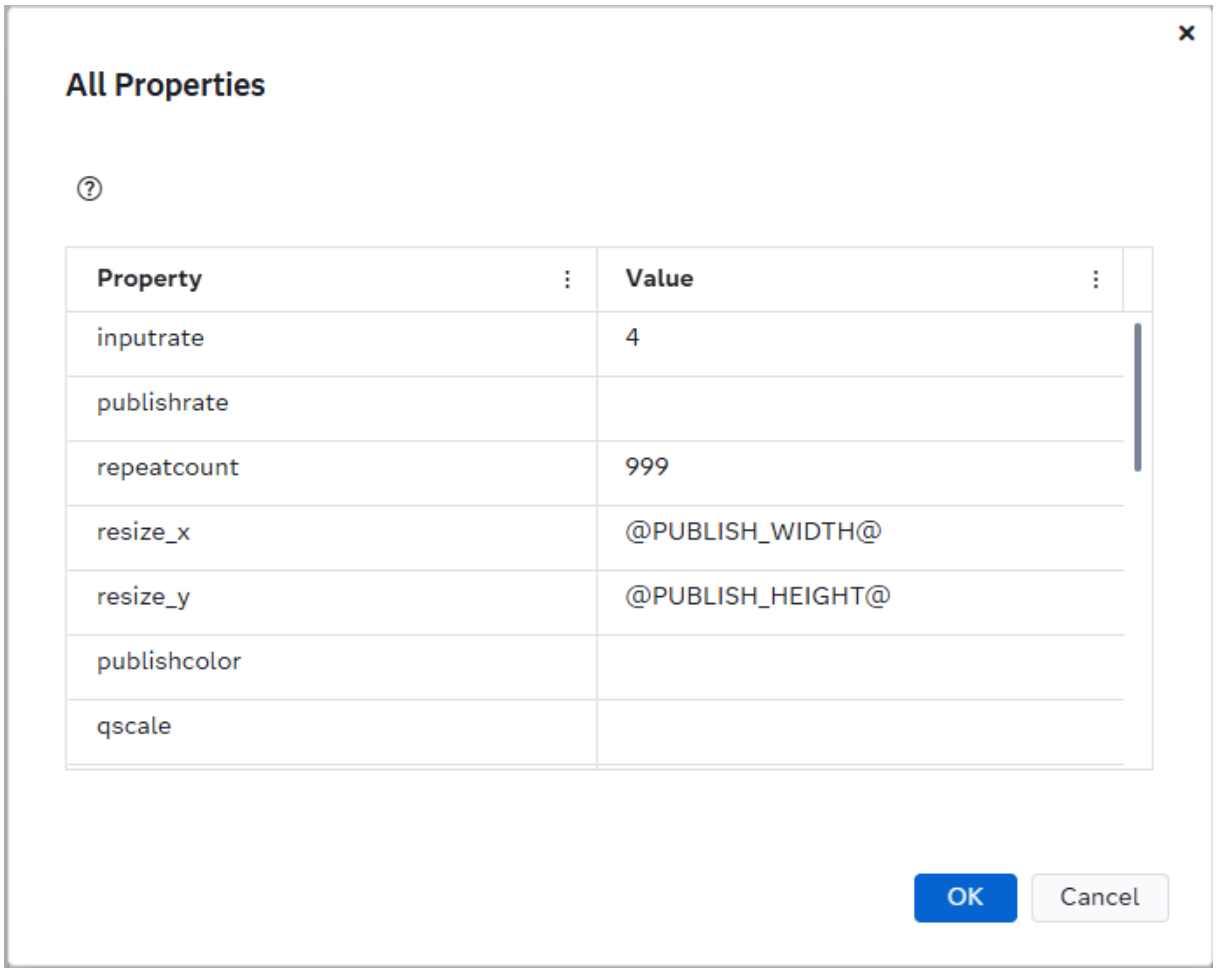
ユーザー定義プロパティをコピーして貼り付けるには、次の操作を実行します。

1 **ユーザー定義プロパティ**ペインでプロパティを選択します。

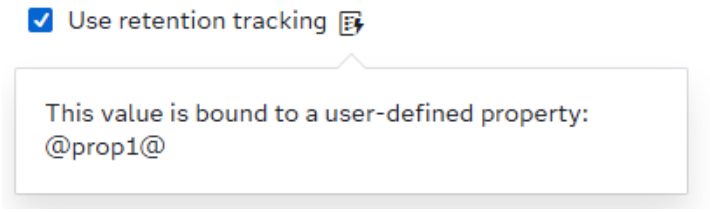
2 をクリックしてユーザー定義プロパティをコピーします。プロパティは@記号で囲まれています。つまり、コピーされたテキストの形式は@*property_name*@です。

3 コピーした参照を SAS Event Stream Processing Studio モデラー内の別の場所に貼り付けます。

- コピーした参照を、テキスト入力を受け入れるフィールドに貼り付けることができます。次の例は、コネクタ構成で使用されているユーザー定義プロパティを示しています。



- XML エディターで、コピーした参照を XML コードに貼り付けることができます。このアプローチにより、チェックボックスの値など、ユーザーインターフェイスでテキストとして 編集できない値をユーザー定義プロパティで制御する高度なユースケースが有効になります。このアプローチを使用すると、 がユーザーインターフェイスコントロールの隣に表示され、その値がユーザー定義プロパティにバインドされていることを示します。次の例は、XML コードに retention-tracking=@prop1@が含まれている場合にユーザーインターフェイスがどのように見えるかを示しています。



SAS Event Stream Processing Studio でのモデルのテスト

テストを実行する

SAS Event Stream Processing Studio を使用して、ご使用のモデルが意図したとおりに動作することを確認できます。受信データを変換してサブスクライバーによって 消費される有意義なイベントストリームにする方法を分析できます。

1 **プロジェクト**ページで、テストするモデルを含むプロジェクトを右クリックします。

2 メニューから**プロジェクトを開く**選択します。

プロジェクトは新しいページに表示されます。

注: プロジェクトに加えた変更をすべて保存したことを確認します。未保存の変更を含むプロジェクトはテストモードで開くことができません。

3 ✓ テストモードへ移行 をクリックします。

ページが表示され、モデルをテストできます。

4 **ESP サーバー**ドロップダウンリストで、テストを実行する ESP サーバーが選択されていることを確認します。

注: テストが Kubernetes クラスターで実行されると、ESP サーバーが Kubernetes クラスターにオンデマンドで作成されます。プロジェクトが停止すると、ESP サーバーは Kubernetes クラスターから削除されます。詳細については、"SAS Event Stream Processing Studio での Kubernetes クラスターの操作"を参照してください。エッジサーバーを使用している場合は、モデルのテストを続ける前に ESP サーバーを登録する必要があります。**ESP サーバー**ページで新しい ESP サーバーを登録できます。

5 プロジェクトに複数の連続クエリが含まれている場合、左側のペインで**連続クエリ**のドロップダウンメニューが使用可能です。関連する連続クエリを選択するには、このクエリを使用します。

6 左側のペインを使用し、結果を表示するウィンドウとフィールドを選択します。デフォルトでは、各ウィンドウの最初の 15 フィールドが選択されます。BLOB フィールドは例外で、デフォルトでは選択されません。

> ヒント　各ウィンドウ名とフィールド名の横にアイコンが表示されます。アイコンの上にカーソルを置くと、ウィンドウのタイプまたはフィールドのタイプが識別されます。

注: 左側のペインには、テストモードに入った時点のウィンドウの出力スキーマが反映されます。その後、ウィンドウの出力スキーマを編集してプロジェクトを保存すると、↻ スキーマの更新 をクリックして左側のペインを更新できます。

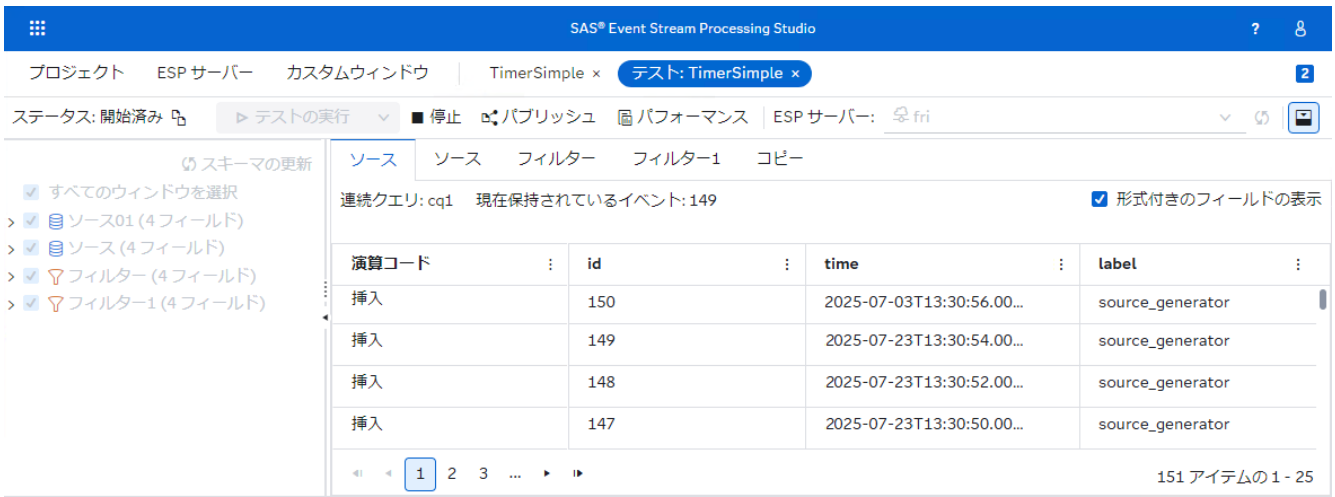
7 テストを実行するには、▷ テストの実行 をクリックします。

テストが Kubernetes クラスターで実行される場合、Kubernetes クラスターでの ESP サーバーの作成に使用される設定を変更できます。

- ▷ テストの実行 をクリックすると、前回テストを実行したときと同じ設定でテストが実行されます。以前に設定を調整していない場合は、デフォルト設定が使用されます。
- 設定を変更するには、▷ テストの実行 横にある▾をクリックし、 テストの構成と実行 をクリックします。**プロジェクトをクラスターにロードして開始**ウィンドウが表示されます。必要に応じて設定を調整し、**OK** をクリックします。このウィンドウで使用可能な設定の詳細については、“Kubernetes クラスター内にある ESP サーバーでのプロジェクトのロードと開始”を参照してください。

8 テスト結果が表示されます。各ウィンドウの結果は、対応するタブに表示されます。

図 24 テストモード



ツールバーのステータスインジケータは、テストの現在のステータスを知らせます。次の表は、考えられるいくつかのテストステータスを示しています。

表 3 テストステータス

		ステータスが表示される環境	
ステータス	説明	Kubernetes	エッジ
初期	テストは初期状態にあり、開始されていません。	✓	✓
開始中	テストが始まっています。	✓	✓
開始済み	テストは開始され、実行中です。		✓
コンテナは初期化されません	Kubernetes クラスターがテスト開始の要求を処理しています。	✓	
作成の保留中です	Kubernetes クラスターがテスト開始の要求を処理しています。	✓	

ステータス	説明	ステータスが表示される環境	
		Kubernetes	エッジ
コンテナは実行中です	テストは Kubernetes クラスターで実行されます。	✓	
ロード中の SAS Micro Analytic Store モジュールの**合計数**	**total** の値はロードされる SAS Micro Analytic Store モジュールの合計数を示し、**count は**これまでにロードされたモジュールの数を示します。	✓	✓
プロジェクトのロードの待機	大量の SAS Micro Analytic Store コードを含むプロジェクトがロードされています。	✓	
停止中	テストは停止しています。	✓	✓
停止済み	テストは中止されました。	✓	✓
テストに失敗しました	テストに失敗しました。ウィンドウが表示され、テストが失敗した理由の詳細が示されます。	✓	✓
不明	ポッドの状態が不明です。詳細については、**ログ**ペインで**ポッドイベント**を選択してください。	✓	

モデル内のウィンドウに 1,000 を超える結果が含まれている場合、テストモードでは最後の 1,000 の結果のみが表示されます。

列ごとに情報をグループ化できます。結果テーブルには、テーブルの上部に水平バーがあり、そこに**列ヘッダーをここにドラッグして列でグループ化します**というテキストが表示されます。列ごとに情報をグループ化するには、列ヘッダーをバーにドラッグします。必要に応じて、追加の列をバーにドラッグすることができます。

または、サブスクライバーコネクタのプロパティで 指定した出力ファイルを開いて、テスト結果を表示することもできます。

フォーマットされたフィールドの表示チェックボックスを使用して、データを ESP サーバーから受信したとおりに表示するか、追加のフォーマットを使用して表示するかを選択できます。チェックボックスがオンの場合に適用される追加の書式設定の例をいくつか示します。

- 日付とタイムスタンプは ISO 8601 形式の協定世界時(UTC)として表示されます (例:2018-11-30T13:33:47.000Z)。このチェックボックスをオフにすると、日付とタイムスタンプは UNIX エポック時間で表示されます。これは、ESP サーバーからデータを受信する場合の形式です。

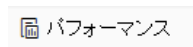
- ドットは、コンマなどの別の区切り文字ではなく、特定の種類の数値データの区切り文字として使用されます。チェックボックスをオフにして、ロケールが別の区切り文字を使用するロケールに設定されている場合は、ドットの代わりにその区切り文字が表示されます。
- ロケールが英語のロケールに設定されていない場合、演算コードはローカライズされた名前を使用して表示されます。チェックボックスをオフにすると、演算コードは常に英語で表示されます。これは、ESP サーバーからデータを受信する方法です。

テストモードのセルに JSON または XML コードが含まれ、セルにカーソルを合わせると、  ボタンがセルに表示されます。ボタンをクリックすると、コードがフォーマットされ、構文の強調表示によってコードが読みやすくなるウィンドウが開きます。

9 テストモードで動作しているモデルに CSV ファイルから WebSocket 接続でイベントをパブリッシュできます。これにより、パブリッシャーコネクタを構成せずに、モデル内のソースウィンドウにイベントを送信できます。これにより、CSV ファイルをサーバーにアップロードせずに、コンピュータ上の便利な場所に配置できます。CSV ファイルからモデルにイベントをパブリッシュしない場合は、このステップをスキップしてください。

CSV ファイルからイベントをパブリッシュするには、  をクリックします。詳細については、“CSV ファイルからイベントをパブリッシュする”を参照してください。

10 下部のペインの**スコアリング API** タブを使用して、スコアリング用のイベントを送信したり、スコアリングされたイベントを表示したりできます。詳細については、“モデルのテスト時にスコアリング API を使用する”を参照してください。

11 モデル内のウィンドウのパフォーマンス情報を表示するには、  をクリックします。

詳細については、“モデルのパフォーマンス情報の表示”を参照してください。

注: エッジサーバー上で実行される ESP サーバー上でテストを実行する場合、パフォーマンス情報は利用できません。

12 環境に Grafana が含まれている場合、SAS Event Stream Processing Data Source Plug-in for Grafana を使用すると、イベントを視覚化できます。詳細については、https://github.com/sassoftware/grafana-esp-plugin を参照してください。

13 テストを停止するには、  をクリックします。

14 モデルのテストが終了したら、ページを閉じます。

注: テストが失敗し、その結果のエラーメッセージで失敗の原因についての説明が得られない場合は、ログペインでテストログを確認して問題をトラブルシューティングできます。詳細については、“モデルのテストログの表示”を参照してください。

プロジェクトがプロジェクトパッケージの一部であり、プロジェクトがサブスクライバーコネクタを使用して出力をファイルに書き込む場合、プロジェクトパッケージの**出力**フォルダー内の出力ファイルにアクセスできます。詳細については、“プロジェクトパッケージを管理する”を参照してください。

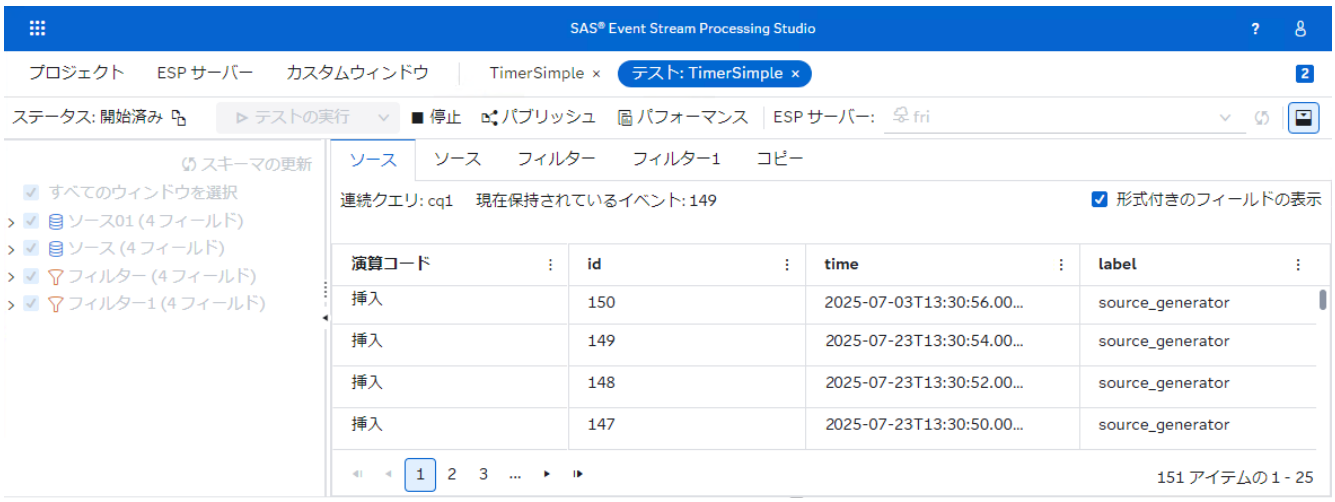
モデルのテストログの表示

テストログ表示の概要

テストモードでモデルのログを表示することで、モデルをリアルタイムで監視できます。ログメッセージは、テストモードの下部にある水平枠であるログ枠に表示されます。

履歴ログメッセージは表示されません。SAS Event Stream Processing Studio は、モデルをテストモードで実行した後にログに記録されたメッセージを表示します。特定のモデルに対して別の ESP サーバーを選択するか、テストモードを閉じると SAS Event Stream Processing Studio に表示されたログメッセージは保持されません。

図 25　ログ記録を有効にしたテストモード



ここでは、ログペインのコントロールの使い方について説明します。

ログメッセージのソースの選択

ログドロップダウンリストを使用して、ログメッセージのソースを選択します。

- **ESP サーバー** – ESP サーバーのログからのメッセージ。これらのメッセージは、モデルを実行している ESP サーバーに関するものです。TRACE レベルメッセージは表示されません。一部のメッセージの詳細はユーザーインターフェイスに表示されません。TRACE レベルのメッセージを含む完全なメッセージテキストを表示するには、Kubernetes ポッドのログをエクスポートして、その内容を表示します。詳細については、“ログのエクスポート”を参照してください。

- **ESP 演算子**– ESP 演算子ログのメッセージ。ESP 演算子は Kubernetes 演算子で、カスタムリソースを使用してアプリケーションを管理するソフトウェア拡張機能です。Kubernetes 演算子の詳細については、Kubernetes 関連ドキュメントを参照してください。このオプションは、エッジサーバーで実行される ESP サーバーでは使用できません。このオプションが有効な場合、システム管理者がアクセスする必要があるような Kubernetes メッセージを、kubectl コマンドを使用して表示することができます。ポッドが起動せず、ログペインのログメッセージにアクセスできない場合は、代わりに**設定**ページの ESP オペレーターログにアクセスできます。詳細については、"ESP オペレータ"を参照してください。
- **コンソール** – ESP サーバーによって作成された標準出力(stdout)メッセージですが、ESP サーバーのログには含まれません。このオプションは、エッジサーバーで実行される ESP サーバーでは使用できません。
- **ポッドイベント** – 特定のポッドに対して Kubernetes クラスターが作成するメッセージ。このオプションを使用すると、システム管理者がアクセスする必要があるような Kubernetes メッセージを、kubectl コマンドを使用して表示することができます。

ログメッセージのフィルタリング

ログメッセージをメッセージタイプでフィルタリングするには、ログペインで次のオプションを 1 つ以上選択します。

- すべて – ESP サーバーログからの TRACE レベルのメッセージを除く、すべてのログメッセージタイプを表示します。
- 情報ー警告またはエラーではない一般的なログメッセージを表示します。
- 警告 - 警告メッセージのみを表示します。
- 致命的なエラーー致命的なエラーおよび標準のエラーを含むエラーメッセージのみを表示します。

ログドロップダウンリストから**コンソール**または**ポッドイベント**を選択した場合、ログメッセージをフィルタリングする機能は利用できません。

ログペインのクリア

ログペインの内容を消去するには、 Clear Logをクリックします。

ログのエクスポート

すべてのログをテキストファイルにエクスポートするには、 すべてのログのエクスポートをクリックします。ログファイルを含む ZIP ファイルがコンピューターにエクスポートされます。ZIP ファイルがエクスポートされる場所は、ブラウザーの設定によって異なる場合があります。

次のログファイルが利用可能です。

- **pod.log** – Kubernetes ポッドの完全なログ。ログには、TRACE レベルのメッセージ(TRACE レベルのログが有効な場合)およびすべてのコンソールメッセージを 含む、すべての ESP サーバーメッセージの完全なテキストが含まれます。

> ヒント　エクスポートされた Kubernetes ポッドログに、表示したいレベルのログ詳細が含まれていない場合は、 **設定**ページの**ユーザーセッション**セクションでログプロパティを変更できます。たとえば、特定のロガーのログレベルを INFO から TRACE に変更できます。詳細については、"ユーザーセッションの設定"を参照してください。

- **previous pod.log** –このファイルは、Kubernetes ポッドが再起動された場合にのみ存在します。このファイルには、元のポッドのログエントリが含まれています。
- **operator.log** – ESP 演算子のログ。

ログペインの表示/非表示

ログペインを非表示にするには、ツールバーで  をクリックします。ペインを表示するには、  をクリックします。

モデルのテスト時にスコアリング API を使用する

SAS Event Stream Processing Studio でテストを実行するとき、下部のペインの**スコアリング API** タブを使用して、スコアリング用のイベントを送信したり、スコアリングされたイベントを表示したりできます。

スコアリング API を使用するプロジェクトの例については、スコアリング API を使用した画像処理のサンプルをインストールします。詳細については、"サンプルプロジェクトのインストール"を参照してください。

スコアリング入力ウィンドウとスコアリング出力ウィンドウがプロジェクトレベルで指定されていることを確認します。詳細については、"スコアリング API の有効"を参照してください。

スコアリングするイベントを送信し、スコアリングされたイベントを表示するには、次のようにします。

1. **入力**テーブルの**スコアリング API** タブで、  をクリックします。
2. 新しく追加されたテーブル行を使用して、スコアリングに送信するイベントの詳細を指定します。
3. プロジェクトがプロジェクトパッケージの一部であり、**入力フィールド**フィールドがプロジェクトレベルで指定されている場合、  をクリックして、スコアリングする生データを含むファイルを選択することもできます。たとえば、画像ファイルを選択できます。ファイルは、プロジェクトパッケージの **test_files** フォルダーに配置する必要があります。
4. プロジェクトレベルで**入力フィールド**フィールドが指定されていない場合は、次のタスクを実行できます。
   -  をクリックして、別のイベント追加します。
   -  をクリックして、テーブルからイベントを削除します。
   -  をクリックして、後で使用できるようにこのスコアリング構成を保存します。ファイルは、プロジェクトパッケージの **test_files** フォルダーの **scoring_configurations** サブフォルダーに保

存されます。スコアリング構成の保存は、プロジェクトがプロジェクトパッケージの一部である場合のみ可能です。

- をクリックし、以前に保存したスコアリング構成をインポートします。
- をクリックし、イベントを JSON コードとして表示または編集します。
- プロジェクトがプロジェクトパッケージの一部である場合は、をクリックして、スコアリングする生データを含むファイルを選択します。たとえば、画像ファイルを選択できます。ファイルは、プロジェクトパッケージの **test_files** フォルダーに配置する必要があります。

  次に、テーブルの 3 番目の列の値を編集して、選択したファイルを参照します。次の形式を使用します。_FILE_:/home/test_files/*filename* 例: _FILE_:/home/test_files/image.jpg

5 **リクエストの送信**をクリックします。

6 プロジェクトレベルで**出力フィールド**フィールドが指定されていない場合、応答イベントは**出力**テーブルに表示されます。をクリックし、をクリックし、イベントを JSON コード表すとして表示します。

7 **出力フィールド**フィールドがプロジェクトレベルで指定されている場合、応答には画像などの生データが含まれます。**出力ファイルの保存**ウィンドウが表示されます。

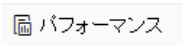
- 応答が単一のイベントである場合、**出力ファイルの保存**ウィンドウは、ファイルの種類を検出してファイルのコンテンツのプレビューを表示しようとします。GIF、JPG、MP4、PNG など、選択したファイルの種類でプレビューが利用可能です。ファイルの種類が検出されない場合は、ファイル名とファイル拡張子を入力します。**保存**をクリックして、出力ファイルします。ファイルはプロジェクトパッケージの**/output/scoring_outputs** フォルダーに保存されます。プロジェクトがプロジェクトパッケージの一部ではない場合、代わりにファイルがコンピューターにダウンロードされます。

図 26　スコアリングされたイメージファイルのプレビューの例

- 応答に複数のイベントが含まれる場合、返されるすべてのファイルに関する情報を含む 1 つの JSON 応答が返されます。**出力ファイルの保存**ウィンドウに JSON コードが表示されます。**保存**をクリックして、出力ファイルします。ファイルはプロジェクトパッケージの**/output/scoring_outputs** フォルダーに保存されます。プロジェクトがプロジェクトパッケージの一部ではない場合、代わりにファイルがコンピューターにダウンロードされます。

モデルのパフォーマンス情報の表示

モデルにパフォーマンスの問題があると思われる場合、パフォーマンス情報を表示すると、問題の原因となっているウィンドウを特定するのに役立ちます。

SAS Event Stream Processing Studio でテストを実行するときに、ツールバーの 

をクリックしてパフォーマンス情報を表示します。

ESP サーバーページからパフォーマンス情報にアクセスすることもできます。詳細については、“SAS Event Stream Processing Studio での ESP サーバーの管理”を参照してください。

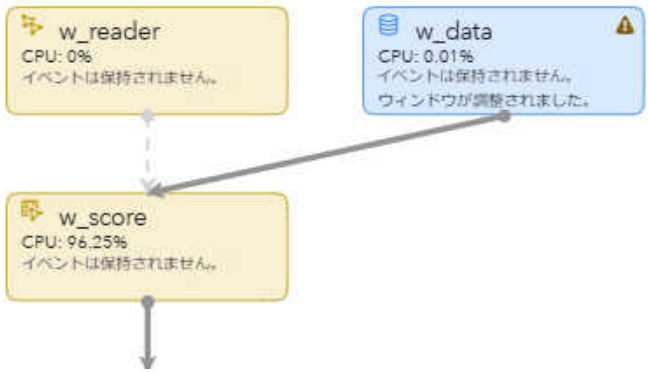
注: エッジサーバー上で実行される ESP サーバー上でプロジェクトが実行されている場合、パフォーマンス情報は利用できません。

次のフィールドがページの上部に表示されます。

- **システムメモリ**フィールドには、Kubernetes ノードで使用可能なメモリの量が表示されます。
- **仮想メモリ**フィールドには、プロジェクト(つまり、Kubernetes ポッド)に割り当てられているメモリの量が表示されます。これは厳しい制限ではありません。常駐サイズが大きくなるにつれて、ESP サーバーはより多くの仮想メモリを必要とします。
- **常駐メモリ**フィールドには、プロジェクト(Kubernetes ポッド)が実際に消費するメモリの量が表示されます。
- **メモリの制限**フィールドには、指定したプロジェクト(Kubernetes ポッド)で使用できるメモリの量が表示されます。**クラスターにプロジェクトをロードして開始**ウィンドウからアクセスできる**配置設定**ウィンドウでメモリの制限を指定できます。

このページの読み取り専用プロジェクトダイアグラムには、CPU 使用率やイベント数など、モデル内の各ウィンドウの概要情報が表示されます。ウィンドウがステートレスである場合、イベントは保持されません。問題の重大度に応じて、  および  アイコンを表示できます。たとえば、  アイコンは、ウィンドウが調整されている場合(ESP サーバーの処理よりも早くデータが到着するためコネクタがブロックされている場合)、またはコネクタに障害が発生した場合に表示されます。

図 27　ダイアグラムのパフォーマンス情報の例



ダイアグラムのウィンドウを選択すると、ダイアグラムの下のタブにそのウィンドウの詳細情報が表示されます。

- **ヘルス**タブには、ウィンドウが調整されている時間やウィンドウのコネクタの現在の状態など、ウィンドウのさまざまなメトリックが表示されます。これらのメトリックの詳細については、*“プロジェクトの正常性メトリックを理解する” (SAS Event Stream Processing: トラブルシューティング)* を参照してください。
- **イベント**タブには、モデルを通過した最新のイベントブロック内のイベントが表示されます。たとえば、各イベントブロックに 1 つのイベントが含まれている場合、このタブには一度に 1 つのイベントが表示されます。
- **ESP サーバーログ**タブには、ESP サーバーログからのメッセージが表示されます。このタブのコントロールは、ESP サーバーログのみが表示されることを除いて、テストモードの**ログ**ペインに似ています。ESP Studio Moder の詳細については、“モデルのテストログの表示” (71 ページ)を参照してください。

各ウィンドウの CPU 使用率情報は、期間を検査し、その時間のうち CPU がビジーである割合をチェックすることによって作成されます。たとえば、1 秒の期間が検査され、CPU が 1 秒の期間全体でビジー状態である場合、100%の CPU 使用率が報告されます。1 秒間の検査が行われ、その時間の半分で CPU がビジー状態である場合、50%の CPU 使用率が報告されます。

ウィンドウがモデル内の他のウィンドウよりも多くの CPU リソースを使用しても、これは必ずしもモデルの設計に問題があることを示すわけではありません。モデル内で最も多くのイベントを処理するように設計されたウィンドウは、処理するイベントが少ないウィンドウよりも多くの CPU リソースを使用する可能性が高いウィンドウでもあります。同様に、ウィンドウが調整されている場合、プロジェクトはイベントの取り込みを管理しています。ソースウィンドウで調整が発生すると、後続のウィンドウで調整が発生するのを防ぐことができます。

クラスターでプロジェクトをロードして開始ウィンドウを使用してテストの配置設定を指定する場合は、適切な量の CPU リソースを指定するように注意してください。リソースを大量に使用するモデルに不十分な量の CPU リソースを指定すると、CPU 使用率と調整に関する警告が表示される場合があります。この警告は、モデルの設計の問題ではなく、CPU リソースの割り当てが不十分であることを示している可能性があります。

CSV ファイルからイベントをパブリッシュする

テストモードで動作しているモデルに CSV ファイルから WebSocket 接続でイベントをパブリッシュできます。これにより、パブリッシャーコネクタを構成せずに、モデル内のソースウィンドウにイベントを送信できます。これにより、CSV ファイルを Kubernetes 永続ボリュームやサーバーにアップロードせずに、コンピュータ上の便利な場所に配置できます。

注: パブリッシュしたい入力データにヘッダー行が含まれていないことを確認してください。ヘッダー行を含む入力データは、モデルへパブリッシュできません。

これを行うには、次の操作を実行します。

1 テストモードでは、イベントをパブリッシュしたいモデルを起動します。詳細については、“テストを実行する”を参照してください。

2 パブリッシュをクリックします。

ファイルのデータをパブリッシュウィンドウが表示されます。

3 **ソースウィンドウ**ドロップダウンリストで、CSV ファイルからイベントを受信するソースウィンドウを選択します。

注: モデルに複数のソースウィンドウが含まれている場合は、モデル内の各ソースウィンドウに対してこの手順を繰り返す必要があります。この機能を使用して、複数のソースウィンドウに同時にイベントをパブリッシュすることはできません。

4 **CSV ファイル**フィールドで**参照**をクリックし、CSV ファイルの場所に移動します。

5 イベントをパブリッシュする CSV ファイルに移動し、**開く**をクリックします。

6 **日付形式**ドロップダウンリストで、形式化された日付を含むイベントが送信されるときに使用される日付形式を選択します。

7 **デフォルトの演算コード**ドロップダウンリストで、入力イベントに演算コードが含まれていない場合に使用するデフォルトの演算コードを選択します。有効な値は、insert、upsert、upsert、および delete です。デフォルト値は insert です。

入力データに演算コードが指定されている場合、選択したデフォルトの演算コードは無視されます。

8 **デフォルトのフラグ**ドロップダウンリストで、入力イベントにフラグが含まれていない場合に使用するデフォルトのフラグを選択します。有効な値は、標準および部分的な更新です。デフォルト値は normal です。

入力データにフラグが指定されている場合、選択したデフォルトのフラグは無視されます。

9 **ブロックサイズ**フィールドに、パブリッシュ時にイベントブロックに入れるイベントの数を指定します。デフォルトは 1 イベントです。したがって、パブリッシャーはデフォルトで各イベントを受信時に挿入します。

10 **維持される最大のイベントレート(1 秒あたりのイベント数)**フィールドに、維持する最大イベントレートをイベント/秒単位で入力します。1 秒が経過する前にこのレートに達すると、パブリッシャーはその秒の残りの間はスリープしてからパブリッシュを再開します。レートを設定しないと、パブリッシャーは最大の速度でイベントをパブリッシュします。デフォルト値は 0 です。

11 **イベントとイベントの挿入間の一時停止(ミリ秒)**フィールドに、イベントの挿入の間に一時停止するミリ秒数を入力します。受信したイベントごとに、そのイベントによってイベントブロックが ESP サーバーにインジェクトされると、指定されたミリ秒数の間、パブリッシャーが一時停止します。デフォルト値は 0 です。

12 **OK** をクリックします。

テストモードに戻ります。イベントは、WebSocket 接続を介してパブリッシュされます。

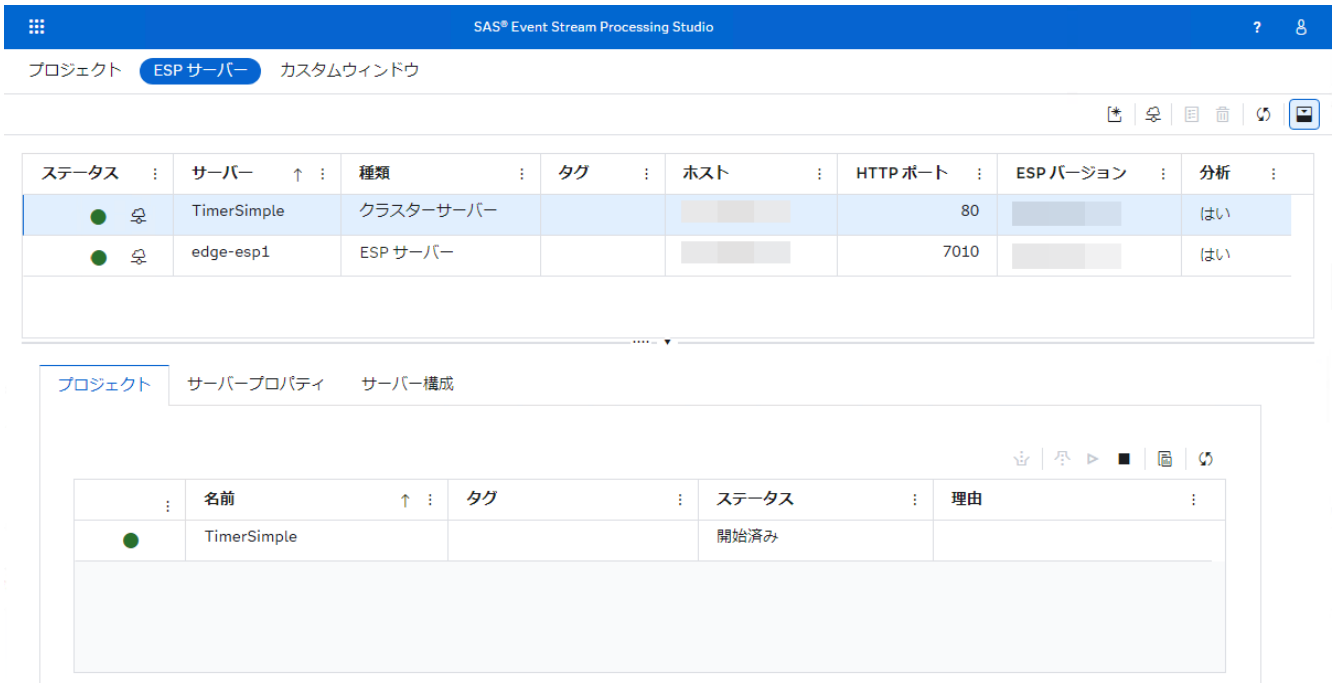
SAS Event Stream Processing Studio での ESP サーバーの管理

SAS Event Stream Processing Studio での ESP サーバーの管理

ESP サーバーページを使用して、配置内の既存の ESP サーバーの詳細を表示できます。このページのコントロールを使用して、ESP サーバーを追加、編集および削除することもできます。

次の図にその例を示します。

図 28　ESP サーバーページ



ESP サーバーページには、ESP サーバーごとに次の情報が表示されます。

- ESP サーバーのステータス。
- ESP サーバーの名前。
- ESP サーバーの種類:
  - ESP サーバー: ESP サーバーはエッジサーバーにあります
  - クラスターサーバー: ESP サーバーは Kubernetes クラスターにあります
- ESP サーバーに割り当てられているタグの識別
- ESP サーバーが実行されているホスト。

  注: プロジェクトが Kubernetes クラスター内の ESP サーバー上で開始される場合、ホスト名は接頭辞 esps の後にクラスター内で開始されたプロジェクトの名前とポッドの一意の識別子が続きます。この 3 つの部分はハイフンで区切られています。プロジェクト名に Kubernetes の命名規則に準拠しない文字が含まれている場合、結果のポッド名には変更された文字が含まれます。プロジェクト名に小文字(発音区別符号を除く)、数字、ハイフン、ピリオドのみが含まれており、名前が小文字で始まる場合は、文字は変更されません。

- HTTP 管理リクエストに使用されるポート。
- ESP サーバーが実行中であるホストにインストールされている SAS Event Stream Processing のバージョン。
- ESP サーバーでストリーミング分析が利用可能かどうか。

ステータス列に、ESP サーバーの状態をまとめたアイコンが表示されます。ESP サーバーの状態は、ESP サーバーの接続性と可用性によって決まります。この情報は、問題のある ESP サーバーに集中するのに役立ちます。ステータスの列には、次のアイコンが表示されます。

アイコン	説明
○	未定義ー ESP サーバーのステータスはまだ確立されていません。
●	使用可能ー ESP サーバーはエッジサーバーにあり、使用可能で、正常に動作しています。
●	クラスターで使用可能ー ESP サーバーは Kubernetes クラスター内にあり、使用可能で、正常に動作しています。
●	使用不可ー ESP サーバーは使用できません。

ESP サーバーに関する追加情報を表示するには、**ESP サーバー**ページで関連する ESP サーバーを選択します。

この追加情報を非表示にするには、  をクリックします。

図 29　プロジェクトタブ

プロジェクトタブがデフォルトで下部ペインに表示され、選択した ESP サーバーにアップロードされたプロジェクトを指定します。

プロジェクトタブを使用して、ESP サーバー上のプロジェクトをロード、アンロード、開始、および停止できます。

Kubernetes クラスター内にない ESP サーバーにプロジェクトをロードするには、をクリックします。**プロジェクトのロード**ウィンドウで、テーブルからロードするプロジェクトを選択し、**OK** をクリックします。このプロセスにより、プロジェクトの作業用コピーが ESP サーバーにアップロードされます。Kubernetes クラスター内の ESP サーバーでのプロジェクトのロードと開始については、“Kubernetes クラスター内にある ESP サーバーでのプロジェクトのロードと開始”を参照してください。Kubernetes クラスター内の ESP サーバーでプロジェクトを停止する方法については、“Kubernetes クラスター内にある ESP サーバーの削除”を参照してください。

注: すでにパブリッシュされているプロジェクトをアップロードすることは許可されていません。

プロジェクトを ESP サーバーにロードしたら、▷をクリックしてプロジェクトを実行できます。**ESP サーバー**ページのテーブル内の対応する行にある **実行中のプロジェクト**フィールドが、実行中のプロジェクトの名前で更新されます 。実行中のプロジェクトを停止するには、■をクリックして、プロジェクトを停止することを確認します。ESP サーバーからプロジェクトをアンロードするには、をクリックします。

モデル内のウィンドウのパフォーマンス情報を表示するには、 をクリックします。詳細については、"モデルのパフォーマンス情報の表示"を参照してください。

注: エッジサーバー上で実行される ESP サーバー上でプロジェクトが実行されている場合、パフォーマンス情報は利用できません。

下部ペインの**サーバーのプロパティ**タブには、ESP サーバーで使用されている認証方法など、ESP サーバーのプロパティに関する情報が表示されます。このタブには、ESP サーバーのホスト名、HTTP ポート、および ESP サーバーでトランスポート層セキュリティ(TLS)が有効かどうかなどの情報も表示されます。詳細については、"ESP サーバーを追加または編集してエッジサーバー上で実行する"を参照してください。ESP サーバーがエッジサーバーで実行されている場合は、**プロパティの編集**をクリックして、ESP サーバーのプロパティを変更できます。これを可能にする **ESP サーバープロパティの編集**ウィンドウが表示されます。

サーバーの構成タブには、コネクタの種類、ストリーミング分析、SAS Event Stream Processing が ESP サーバー上で処理されるイベントの数を計測するために有効になっているかどうかという情報が含まれています。

SAS Event Stream Processing Studio での Kubernetes クラスターの操作

SAS Event Stream Processing Studio には、Kubernetes クラスターでプロジェクトをロードして実行する機能が含まれています。プロジェクトがロードされてクラスター内で開始されると、ESP サーバーがオンデマンドで Kubernetes クラスター内に作成されます。プロジェクトが停止すると、ESP サーバーは Kubernetes クラスターから削除されます。それ以外の場合、Kubernetes クラスター内の ESP サーバーは、エッジサーバー上の ESP サーバーと同じように動作します。

ESP サーバーページから、Kubernetes クラスター内の ESP サーバーでプロジェクトをロードして開始できます。Kubernetes クラスター内の ESP サーバーでプロジェクトを開始する方法の詳細については、"Kubernetes クラスター内にある ESP サーバーでのプロジェクトのロードと開始"を参照してください。Kubernetes クラスター内の ESP サーバー上のプロジェクトを停止することもできます。ESP サーバーが Kubernetes クラスターから自動的に削除されます。詳細については、"Kubernetes クラスター内にある ESP サーバーの削除"を参照してください。

ESP サーバーを追加または編集してエッジサーバー上で実行する

このトピックは、エッジサーバーで実行される ESP サーバーに適用されます。ESP サーバーを追加すると、ESP サーバーの詳細が ESP サーバーページの表に追加されます。これにより、SAS Event Stream Processing Studio は、その ESP サーバーを認識するようになります。ESP サーバーの詳細を表に正常に追加するためには、ESP サーバー自体が存在している必要があります。エッジ環境にある ESP サーバーを開始する方法の詳細については、"エッジ環境での ESP サーバーの開始" (*SAS Event Stream Processing: Edge 環境での ESP サーバーの使用*)を参照してください。Kubernetes クラスタ

ーの ESP サーバーのについては、“SAS Event Stream Processing Studio での Kubernetes クラスターの操作”を参照してください。

1 新規 ESP サーバーを **ESP サーバー**ページで追加する場合は、をクリックします。

ESP サーバープロパティ ウィンドウが表示されます。

または

既存の ESP サーバーの詳細を **ESP サーバー**ページで編集する場合は、対象の ESP サーバーを選択し、をクリックします。

ESP サーバープロパティの編集ウィンドウが表示されます。

2 ESP サーバーを追加または編集するには、次の手順を実行します。

a **名前**フィールドでは、ESP サーバーを識別する名前を入力または更新します。

b 必要に応じて、**説明**フィールドでは、ESP サーバーの説明を入力または更新します。

c **ホスト**フィールドでは、ESP サーバーのホスト名または IP アドレスを入力または更新します。

d **HTTP ポート**フィールドでは、ESP サーバーの管理ポート番号を入力または更新します。

e 必要に応じて、**編集**をクリックして**認証**フィールドの設定を変更します。

認証ウィンドウが表示されます。

- **なし**: これはデフォルトのオプションです。
- **Kerberos**: このオプションは、ESP サーバーが Kerberos を使用した認証を要求するように構成されている場合にのみ関係します。

f 必要に応じて、**SSL を使用して接続**チェックボックスを選択します。このオプションを選択するのは、ESP サーバーが SSL 暗号化を要求するように構成されている場合のみです。

g (オプション)**接続のテスト**をクリックします。接続をテストすることで、無効な ESP サーバーの接続が発生し、無効な接続が更新されたときにタイムアウトすることを防ぎます。

h 必要に応じて、**タグ**フィールドでは、ESP サーバーを説明するキーワードを入力または更新して、Enter キーを押します。

i 必要に応じて、**サーバーログを有効にする**チェックボックスをオンにして、ESP サーバーでのログ記録を有効にします。

j 必要に応じて、**保持するメッセージ数**フィールドで、ESP サーバーログに保持されるデフォルトのメッセージ数を変更します。デフォルトは 10,000 メッセージです。

k **OK** をクリックします。

Kubernetes クラスター内にある ESP サーバーでのプロジェクトのロードと開始

SAS Event Stream Processing Studio で Kubernetes クラスターを使用するための概要については、“SAS Event Stream Processing Studio での Kubernetes クラスターの操作”を参照してください。

1 **ESP サーバー**ページで、をクリックし、**プロジェクトの開始**を選択します。

プロジェクトをクラスターにロードして開始ウィンドウが表示されます。

2 **プロジェクト**フィールドでプロジェクトを選択します。

3 プロジェクトがプロジェクトパッケージの一部ではない場合は、入力ファイルが目的の場所にあることを確認してください。プロジェクトで Kubernetes 永続ボリュームを必要とするコネクタを使用する場合は、Kubernetes 永続ボリューム用に構成された場所に入力ファイルを配置する必要があります。SAS Event Stream Processing が想定する、入力ファイルが配置される基本的な場所は**/mnt/data** ですが、入力ファイルを配置する必要がある類似の場所は、ファイルストレージの設定方法によって異なります。

サポートが必要な場合は、システム管理者に連絡してください。

4 オプション)**配置設定**セクションの値を調整します。**配置設定**セクションでは、クラスター内に ESP サーバーを作成する際に使用する設定が表示されます。選択されたプロジェクトで以前使用された設定にアプリケーションがアクセスできる場合は、その設定がデフォルト値として表示されます。それ以外の場合は、プロジェクトの計算要件がデフォルト値として使用されます。詳細については、"計算要件の指定"を参照してください。システム管理者に連絡してください。

a **配置設定の編集**をクリックします。

配置設定ウィンドウが表示されます。

b 次のリソースごとに、ESP サーバーによる要求量(初期値)と最大使用量を指定します。

- メモリの容量。
- CPU リソースの量。
- NVIDIA グラフィックスプロセッシングユニット(GPU)リソースの量。SAS Event Stream Processing Studio のユーザーインターフェイスには、要求量と最大量の両方を設定するためのフィールドが 1 つあり、これらの量は同じでなければなりません。

注:

- メモリと CPU リソースに設定する必要がある要求量と最大量は、状況によって異なります。多くの場合、要求額と上限額は同じ値に設定できます。ただし、リソースの使用を最適化するために、別の値を設定できます。異なる値を設定すると、プロジェクトを実行する Kubernetes ポッドを、ピーク使用時にポッドが必要とする最大量よりも少ない量のリソースで開始できることを意味します。ただし、プロジェクトを長時間実行したままにしておくことが予想される場合、要求された量と最大量の間に大きな差があると、Kubernetes クラスターが最大量に達する前にポッドを起動してしまう可能性があります。つまり、Kubernetes クラスターは、クラスターが要求された最小量を満たすことができる場合にポッドを開始する可能性がありますが、指定した最大量のリソースを取得できない可能性があります。十分なリソースが不足すると、プロジェクトが予期せず停止する可能性があります。
- システム管理者は、メモリ量と CPU リソースの割り当て量に制限を設定することができます。指定した値がそのような制限値を超える場合は、代わりにシステム管理者が指定した値が使用されます。システム管理者によって設定された値を超える値を設定しようとすると、ユーザーインターフェイスにエラーメッセージが 表示されます。

c **OK** をクリックします。

5 **OK** をクリックします。

新しい ESP サーバーが作成されます。ロードしたプロジェクトが ESP サーバーで開始されます。

環境に Grafana が含まれている場合、SAS Event Stream Processing Data Source Plug-in for Grafana を使用すると、イベントを視覚化できます。詳細については、https://github.com/sassoftware/grafana-esp-plugin を参照してください。

エッジサーバー上で動作する ESP サーバーを SAS Event Stream Processing Studio から削除する

ESP サーバーページのテーブルから ESP サーバーの詳細を削除できます。これは、SAS Event Stream Processing Studio がその ESP サーバーを認識しなくなったことを意味します。しかし、ESP サーバー自体は存在し続けています。ESP サーバーの詳細の削除は、使用されなくなった ESP サーバーがテーブルに含まれている場合に便利です。実行中の ESP サーバーを削除することもできます。テーブルから特定の ESP サーバーを削除するには、次の操作を実行します。

1 **ESP サーバー**ページで、テーブルから削除する ESP サーバーを選択し、🗑をクリックします。

ESP サーバーの削除ウィンドウが表示されます。

2 ESP サーバーのテーブルからの削除を確認します。

注: 複数の ESP サーバーを同時に削除するには、Ctrl キーを押しながら、テーブルから削除する ESP サーバーを選択して、🗑をクリックします。その後、削除を確認してください。

Kubernetes クラスター内にある ESP サーバーの削除

1 **ESP サーバー**ページで、表から削除するプロジェクトを選択します。

2 ☁をクリックし、および**プロジェクトの停止**を選択します。

> ヒント　または下部ペインの**プロジェクト**タブで、削除するプロジェクトを選択し、■をクリックします。

プロジェクトを停止し、クラスターから ESP サーバーを削除ウィンドウが表示されます。

3 **削除**をクリックします。

プロジェクトが停止し、ESP サーバーが Kubernetes クラスターから削除されます。

バージョニングの概要

SAS Event Stream Processing Studio および SAS Event Stream Manager を使用して、プロジェクトの複数のバージョンを管理できます。

SAS Event Stream Processing Studio および SAS Event Stream Manager で利用できるバージョン管理機能は、SAS Event Stream Processing の配置方法によって異なります。

スタンドアロン版 SAS Event Stream Processing が配置されている場合、ファイルベースのバージョン管理を使用してプロジェクトバージョンをパブリッシュします。プロジェクトのバージョンは、SAS Event Stream Processing 内部の Git リポジトリに保存され、外部 Git リポジトリと同期できます。詳細については、“ファイルベースのバージョン管理”を参照してください。

SAS Event Stream Processing が他の SAS 製品とともに配置された場合、プロジェクトバージョンは、各プロジェクトの種類に応じて、ファイルベースのバージョン管理または SAS Viya プラットフォームサービスを使用してパブリッシュできます。

- プロジェクトがプロジェクトパッケージの一部である場合、プロジェクトのバージョンはファイルベースのバージョン管理を使用してパブリッシュされます。
- プロジェクトがプロジェクト XML ファイルである場合、プロジェクトバージョンは SAS Viya プラットフォームサービスを使用してパブリッシュされます。その後、SAS Event Stream Processing Studio を使用して、プロジェクト XML ファイルをファイルベースのバージョン管理を使用するプロジェクトパッケージに変換します。詳細については、“ファイルベースのバージョン管理を使用するようにプロジェクトを変換する”を参照してください。

  SAS Viya プラットフォームサービスがプロジェクトバージョンをパブリッシュするために使用される際、ファイルサービスとファイルストアサービスがプロジェクトバージョンをデータベースに保存するために使用されます。詳細については、“SAS Viya Platform Services を使用したプロジェクトバージョンのパブリッシュの概要”を参照してください。

スタンドアロン SAS Event Stream Processing が配置されている場合、SAS Viya プラットフォームサービスを使用してプロジェクトを公開することはできません。

ファイルベースのバージョン管理を使用してプロジェクトバージョンをパブリッシュする

ファイルベースのバージョン管理

ファイルベースのバージョン管理を使用するには、プロジェクトがプロジェクトパッケージの一部である必要があります。詳細については、“プロジェクトパッケージ”を参照してください。

ファイルベースのバージョン管理の利点

ファイルベースのバージョン管理が利用可能で、プロジェクトがプロジェクトパッケージの一部である場合、プロジェクトの XML コードとプロジェクトパッケージ内の他のすべてのファイルがバージョン管理されます。これは、ファイルベースのバージョン管理の利点の 1 つです。

ファイルベースのバージョン管理のもう 1 つの利点は、プロモーションレベルを使用してプロジェクトのバージョンを管理できることです。プロモーションレベルは 3 つあります。それは開発、テスト、本番の 3 つです。プロモーションレベルの目的は、非プロダクションプロジェクトのバージョンがプロダクション環境に配置されるのを防ぐことです。

- 開発– このプロモーションレベルは、SAS Event Stream Processing Studio Modeler でプロジェクトの設計に取り組むことを目的としています。プロジェクトの複数のバージョンを保存できます。
- テスト– このプロモーションレベルは、プロジェクトバージョンのテストを目的としています。SAS Event Stream Processing 環境に SAS Event Stream Processing Studio および SAS Event Stream Manager の単一インスタンスが含まれている場合、SAS Event Stream Processing Studio のプロジェクトバージョンを開発からテストにプロモートすると、プロジェクトバージョンが SAS Event Stream Manager で使用できるようになります。SAS Event Stream Processing 環境に SAS Event Stream Manager のインスタンスが複数含まれている場合は、SAS Event Stream Processing Studio のプロジェクト バージョンを開発からテストにプロモートし SAS Event Stream Manager テストインスタンスにインポートします。どちらの場合でも、SAS Event Stream Manager を使用して、テスト目的で使用している配置にプロジェクトバージョンを配置します。
- プロダクション– プロジェクトバージョンがプロダクション環境で使用できる状態になっていることが確認できたら、SAS Event Stream Manager を使用して、プロジェクトバージョンをテストからプロダクションにプロモートできます。SAS Event Stream Processing 環境に SAS Event Stream Manager のインスタンスが複数含まれている場合は、プロジェクトを SAS Event Stream Manager プロダクションインスタンスにもインポートする必要があります。次に、SAS Event Stream Manager を使用して、プロジェクトのバージョンを、プロダクション目的で使用している配置に配置します。

SAS Event Stream Manager を使用してプロジェクトをプロモートおよびインポートする方法の詳細については、“Managing Project Versions That Are Published Using File-Based Versioning” (*SAS Event Stream Processing: Using SAS Event Stream Manager*)を参照してください。

ファイルベースのバージョニングと Git

ファイルベースのバージョン管理を使用すると、デフォルトでプロジェクトバージョンは SAS Event Stream Processing 内部の Git リポジトリに保存されます。

プロジェクトのバージョンは、リモート Git サーバー上のリポジトリと同期することもできます。各プロジェクトは独自のリポジトリに保存されます。

注意
内部 Git リポジトリおよびリモート Git リポジトリと対話する場合は、SAS Event Stream Processing のみを使用します。 サードパーティツールを使用して Git リポジトリと対話すると、データが失われる可能性があります。

注意
バックアッププロセスが実行されていることを確認します。 内部 Git リポジトリを使用する場合、SAS Event Stream Processing Studio が使用される永続ボリューム上のプロジェクトデータに対してバックアッププロセスを実行する必要があります。 プロジェクトデータがリモート Git サーバーに保存されている場合は、リモート Git サーバーのバックアッププロセスを設定する必要があります。 バックアップを作成していない場合、データが失われる可能性があります。

ユーザーがプロジェクトをリモートリポジトリに保存できるようにするには、システム管理者は最初にリモート Git サーバーへの接続を指定する必要があります。詳細については、“リモート Git サーバーへの接続を指定する”を参照してください。

次のようにリモートリポジトリを作成できます。

- プロジェクトを作成するときに、リモートリポジトリを作成するオプションを選択します。詳細については、“プロジェクトの作成”を参照してください。
- 既存のプロジェクトのリモートリポジトリを作成します。詳細については、“既存のプロジェクトのリモートリポジトリを作成”を参照してください。

リモート Git サーバーに保存されている既存のプロジェクトをインポートできます。詳細については、 “プロジェクトのインポート”を参照してください。

サーバーの種類が GitHub または Gitea の場合、個々のユーザーではなく組織が所有するリモートリポジトリにプロジェクトを保存できます。詳細については、次の資料を参照してください。

- “プロジェクトのインポート”
- “プロジェクトを空のリモートリポジトリにリンク”

ヒント　**プロジェクト**ページのリモートリポジトリ列を表示すると、既存のプロジェクトが内部 Git リポジトリに保存されているかリモート Git リポジトリに保存されているかを確認できます。

プロジェクトが保存されているリモート Git サーバーの詳細を確認することもできます。詳細については、“リモートリポジトリの詳細を表示”を参照してください。

ファイルベースのバージョン管理と Git Large File Storage

Git Large File Storage(LFS)は、大きなファイルを効率的に管理するために設計された Git の拡張機能です。大きなファイルを Git リポジトリに直接保存する代わりに、Git LFS はそれらのファイルを Git 内の参照に置き換えます。実際のファイルのコンテンツは個別に保存され、必要な場合にのみ取得されます。Git LFS は一般的に、圧縮されない大規模ファイルや頻繁に変更されるファイルに使用されます。

Git LFS は Git と同じプラットフォームによって提供されることがよくありますが、Git 自体の一部ではありません。通常の Git ファイル(テキストファイルなど)は Git リポジトリに保存されますが、大きなファイルは Git LFS サービスによって個別に保存されます。同じ Git プロバイダーを使用している場合でも、Git LFS は標準の Git ストレージとは異なります。たとえば、Gitea LFS ストレージは標準の Gitea ストレージとは異なります。

Git LFS を使用しないと、大きなファイルを Git リポジトリに直接保存すると、いくつかの問題が発生する可能性があります。

- Git ホスト製品の多くは、最大ファイルサイズを強制します。この制限を超えるファイルを追加しようとすると、ファイルは拒否されます。
- ファイルごとの制限に加えて、Git ホストサービスはフェアユースポリシーや全体的なリポジトリサイズ制限を適用することがよくあります。
- 大きなファイルに変更を加えるたびに、リポジトリサイズが大幅に増加するため、リポジトリサイズがすぐに増加します。
- Git はテキストファイル用に最適化されているため、大きなバイナリファイルではパフォーマンスが低下する可能性があります。大きなバイナリファイルは変更を効率的に比較しないため、保存などの一般的な操作が大幅に遅くなる可能性があります。

SAS Event Stream Processing Studio では、Git LFS は大規模ファイルを管理するためのファイルベースのバージョン管理の一部として使用できます。

- プロジェクトパッケージを作成するときに、そのプロジェクトに Git LFS を使用することを選択できます。その後、バイナリファイルをプロジェクトパッケージにアップロードすると、ファイルは Git LFS に自動的に保存されます。テキストファイルをプロジェクトパッケージにアップロードすると、SAS Event Stream Processing Studio は、ファイルを Git LFS に保存するかどうかを選択するよう求められます。
- プロジェクトパッケージをインポートするとき、そのプロジェクトに Git LFS を使用することを選択できます。プロジェクトパッケージ内のバイナリファイルは、Git LFS に自動的に保存されます。

  その後、バイナリファイルをプロジェクトパッケージにアップロードすると、ファイルは Git LFS に自動的に保存されます。テキストファイルをプロジェクトパッケージにアップロードすると、SAS Event Stream Processing Studio は、ファイルを Git LFS に保存するかどうかを選択するよう求められます。

以下は、SAS Event Stream Processing Studio がデフォルトで Git LFS に格納するファイルの種類の例です: JPG、MP4、PDF、PNG、ONNX。

Git LFS が SAS Event Stream Processing Studio で使用される場合、次の利点が提供されます。

- 大規模ファイルの効率的なバージョン管理。プロジェクトパッケージの一部である大きなファイルは、Git リポジトリを開くことなくバージョン管理されます。
- Git の完全なファイルコンテンツの代わりにファイル参照を転送することによる、パフォーマンスの向上。
- プロジェクトのバージョンの一貫。大きなファイルは、論理的にプロジェクトバージョンの一部のままです。プロジェクトバージョンを取得すると、大きなファイルの正しいバージョンが自動的に取得されます。

注: Git LFS は、プロジェクトパッケージが次のいずれかの方法で保存されている場合に使用できます。

- SAS Event Stream Processing の内部にある Git リポジトリ

- Gitea のリモートリポジトリ

重要　SAS Event Stream Processing には、Git LFS サービスは含まれていません。SAS Event Stream Processing Studio 内で Git LFS 機能をで使用する。

- 組織に Git LFS サービスが必要であり、Git LFS サービスが組織の Git プロバイダーで有効になっている必要があります。
- リモートリポジトリ接続を作成するときは、システム管理者が Git LFS サービスへのアクセスを構成する必要があります。詳細については、“リモート Git サーバーへの接続を指定する”を参照してください。

次のタスクを完了すると、プロジェクトに Git LFS を使用するオプションが使用可能になります。

- “プロジェクトの作成”
- “プロジェクトのインポート”
- “サンプルプロジェクトのインストール”

プロジェクトパッケージペインのファイル名の隣にあるテキスト**(LFS)**は、ファイルが Git Large File Storage に保存されていることを示します。例については、“ユーザーインターフェイスの制御” を参照してください。

プロジェクトで Git LFS を有効にした場合、そのプロジェクトで Git LFS を後で無効にすることはできません。ただし、をクリックし、**名前を付けて保存**ウィンドウで **Git Large File Storage(LFS)の有効化**チェックボックスをオフにしてから、プロジェクトを別の名前で保存することもできます。

プロジェクトのバージョン管理

このトピックは、ファイルベースのバージョン管理を使用してプロジェクトバージョンをパブリッシュする場合に関係します。プロジェクトのバージョンの概要については、“ファイルベースのバージョン管理”を参照してください。

ユーザーインターフェイスの制御

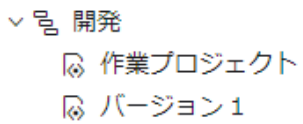
プロジェクトバージョンを表示するには、左側のツールバーでプロジェクトを開いて、をクリックします。プロジェクトのバージョン情報の例を次の図に示します。

図 30　プロジェクトのバージョンの表示例

プロジェクトバージョンペインを使用して、関心のあるプロジェクトのバージョンを選択します。

- ペインの上部に*作業プロジェクト*が表示されます。

  図 31　作業中のプロジェクト

  これは、左側のツールバーの  をクリックするとデザインを変更できるプロジェクトバージョンです。

- 他のプロジェクトバージョンもペインに表示されます。作成後に変更することはできません。ただし、作業プロジェクトを他のバージョンのいずれかに戻して、指定したバージョンを新しい作業プロジェクトにすることができます。

プロジェクトバージョンは、このトピックで説明するバージョン管理機能を使用する場合にのみ作成されます。プロジェクトの設計に取り組むとき(ウィンドウの追加やコネクタの構成など)、それらの変更は作業プロジェクトの XML コードに保存されます。これらの変更に対してプロジェクトバージョンは作成されません。

右側のペインには、**プロジェクトバージョン**ペインで選択したプロジェクトバージョンの詳細が表示されます。

- **管理**タブを使用して、プロジェクトバージョンを作成したり、プロジェクトバージョンを別のプロモーションレベルにプロモートしたり、作業プロジェクトを特定のプロジェクトバージョンに戻したりすることができます。このタブで実行されるアクションによって、**プロジェクトバージョン**ペインに新しいバージョンが表示されます。

> ヒント　**管理**タブには、選択したプロジェクトバージョンのステータスに関するメッセージも表示されます。次に例を示します。 バージョン 1 は、<ユーザー>によって<時間>にテストにプロモートされました。 この情報を使用して、プロジェクトバージョンがプロモートされたプロモーションレベルを知ることができます。

- プロジェクトのバージョン履歴を表示するには、**履歴**タブを使用します。

特定のプロジェクトのバージョンをテストするには、次の例で示すように、プロジェクトのバージョンをテストにプロモートし、SAS Event Stream マネージャーを使用して、そのプロジェクトのバージョンをテスト目的で使用している配置に配置します。

プロジェクトバージョンを開発からテストにプロモートさせる例

次の手順では、セーリングサンプルを使用して、プロジェクトバージョンを作成およびプロモートするための基本的なシナリオを示します。次のタスクを学習します:

- プロジェクトのバージョンを保存する。
- 作業プロジェクトに変更を加える。
- 変更を元に戻す。
- プロジェクトバージョンを開発からテストにプロモートする。
- SAS Event Stream Manager: プロジェクトのバージョンをテストし、テストからプロダクションにプロモートし、プロダクションに配置します。

> **注:** この例では、SAS Event Stream Processing 環境の単一の SAS Event Stream Manager インスタンスを想定しています。SAS Event Stream Manager インスタンスが複数の場合、プロジェクトを各プロモーションレベルにプロモートした後、プロジェクトを関連する SAS Event Stream Manager インスタンスにインポートする必要があります。関連するプロモーションレベルにアクセスできる SAS Event Stream Manager インスタンスのみがプロジェクトをインポートできます。詳細については、“Deploying a Project in a Separated System” (*SAS Event Stream Processing: Using SAS Event Stream Manager*)を参照してください。

次の手順を実行します:

1 セーリングの例をインストールします。

   a SAS Event Stream Processing Studio において、**プロジェクト**ページで**サンプルプロジェクトの表示**をクリックします。

   b **SAS Event Stream Processing の例**サンプルウィンドウでセーリングサンプルを探し、**サンプルのインストール**をクリックします。

   SAS Event Stream Processing Studio モデラーが表示され、プロジェクトダイアグラムが表示されます。

> ヒント　セーリングのサンプルのプロジェクトダイアグラムと、ウィンドウを介したイベントの流れの詳細については、https://github.com/sassoftware/esp-studio-examples/tree/main/EndtoEndExamples/sailing の README を参照してください。

2 左側のツールバーの  をクリックすると、**プロジェクトバージョン**ペインが表示されます。

3 作業プロジェクトの現在の状態を新しいプロジェクトバージョンとして保存するとします。

   a **作業プロジェクト**を選択します。

   b **管理**タブで、**新しいバージョン**をクリックします。

   c **新しいバージョン**ウィンドウで、タグの**優先順位**およびコメント**プロジェクトの最初のバージョン**を入力し、**OK** をクリックします。

   新しいバージョンは、**プロジェクトバージョン**ペインに**バージョン 1** として表示されます。ペイン内のこの新しいバージョンの上にマウスを移動すると、入力したタグとコメントが表示されることを確認します。

4 ここで、作業プロジェクトに変更を加えたいとします。

   a 左側のツールバーの  をクリックします。

   b 新しいウィンドウをワークスペースに 追加し、  をクリックします。

5 気が変わって、追加のウィンドウを含まない以前のプロジェクトバージョンの方が目的に適していると判断したとします。以前のバージョンに戻す手順は次のとおりです。

   a 左側のツールバーの  をクリックします。

   b **作業プロジェクト**を選択します。

   c **管理**タブで、**最後のバージョンに戻す**をクリックします。

   d **元に戻す**ウィンドウで、**元に戻す**をクリックします。

   e 左側のツールバーの  をクリックします。追加したウィンドウがそこに存在しないことを確認します。元に戻すは成功しています。

   f  をクリックします。

   g  をクリックし、プロジェクトのバージョン管理に戻ります。

6 バージョン 1 をテストにプロモートさせることにしたとします。

   a **バージョン 1** を選択します。

   b **管理**タブで、**バージョン 1 をテストにプロモート**をクリックします。

   c **バージョンのプロモート**ウィンドウで、**OK** をクリックします。

   管理タブには次のメッセージが表示されます。バージョン 1 は、 <ユーザー>によって<時間>にテストにプロモートされました。

7 SAS Event Stream Manager を使用して、テスト目的で使用する配置にバージョン 1 を配置してテストし、後でバージョン 1 をテストからプロダクションにプロモートします。“Example of

Promoting a Project Version from Test to Production" (*SAS Event Stream Processing: Using SAS Event Stream Manager*)の手順に従います。

ファイルベースのバージョン管理を使用するようにプロジェクトを変換する

ファイルベースのバージョン管理は、リリース 2024.08 から利用可能になりました。詳細については、"ファイルベースのバージョン管理"を参照してください。

このトピックでは、SAS Viya プラットフォームサービスを使用して以前にパブリッシュされたプロジェクトを一度に 1 つのプロジェクトずつ手動で変換する方法について説明します。ただし、管理者はプロジェクトのバルク変換を実行できます。詳細については、"プロジェクトをバルクでファイルベースのバージョン管理に変換"を参照してください。

プロジェクトに以前にプロジェクトパッケージがなかった場合は、変換の一部としてプロジェクトパッケージが作成されます。

> 重要　変換を元に戻すことはできません。

SAS Viya プラットフォームサービスを使用してパブリッシュされたプロジェクトをファイルベースのバージョン管理を使用するように変換するには、次の手順を実行します:

1 SAS Event Stream Processing Studio でプロジェクトを開きます。

   プロジェクトが SAS Viya プラットフォームサービスを使用してパブリッシュされたこと、および代わりにファイルベースのバージョン管理を使用できることを通知するメッセージが表示されます。

   注: メッセージが表示されない場合、プロジェクトは以前にパブリッシュされていません。

2 **プロジェクトの変換** をクリックします。

3 **ファイルベースのバージョン管理の使用**ウィンドウで**はい**をクリックし、プロジェクトを変換します。

4 左側のツールバーのをクリックし、プロジェクトパッケージが存在することを確認します。プロジェクトにファイルへの参照が含まれている場合(たとえば、テストデータを含むファイル)、それらのファイルをプロジェクトパッケージに追加し、ファイルの新しい場所を指すように参照を調整することを検討してください。詳細については、"プロジェクトパッケージを管理する"を参照してください。

5 左側のツールバーのをクリックし、**プロジェクトバージョン**ペインにプロジェクトのバージョンが表示されることを確認します。詳細については、"プロジェクトのバージョン管理"を参照してください。

SAS Viya Platform Services を使用したプロジェクトバージョンのパブリッシュ

SAS Viya Platform Services を使用したプロジェクトバージョンのパブリッシュの概要

SAS Viya プラットフォームサービスを使用してプロジェクトのバージョンをパブリッシュできる状況については、“バージョニングの概要”を参照してください。

SAS Viya プラットフォームサービスがプロジェクトバージョンをパブリッシュするために使用される際、ファイルサービスとファイルストアサービスがプロジェクトバージョンをデータベースに保存するために使用されます。

プロジェクトバージョンは、パブリッシュされた後で編集することはできません。編集可能なプロジェクトは、プロジェクトの作業コピーとして指定されます。これは、パブリッシュするまでプロジェクトバージョンにはなりません。プロジェクトの作業コピーを使用すると、以前にパブリッシュしたプロジェクトバージョンに影響を与えずにプロジェクトを変更できます。

プロジェクトバージョンをパブリッシュすると、そのバージョンの XML コードが更新され、一意の ID 番号が表示されます。初めてパブリッシュされたプロジェクトバージョンにはバージョン番号 1.1 が割り当てられます。小数点の左側の数字はプロジェクトのメジャーバージョン番号です。小数点の右側の数字はプロジェクトのマイナーバージョン番号です。プロジェクトの後続バージョンをパブリッシュすると、メジャーバージョン番号が増えます。SAS Event Stream Processing Studio では、メジャープロジェクトバージョンはパブリッシュできますが、マイナープロジェクトバージョンはパブリッシュできません。マイナーバージョンは、SAS Event Stream Manager のプロジェクトバージョンに変更を加えたときに作成されます(たとえば、SAS Event Stream Manager を使用してプロジェクトが参照する SAS Micro Analytic Service モジュールへの更新を受け入れたとき)。

バージョン管理ページのバージョン階層でプロジェクトのメジャーバージョンとマイナーバージョンを表示できます。

プロジェクトのバージョンのパブリッシュ

1 **プロジェクト**ペインで、関連するプロジェクトを右クリックし、 **プロジェクトを開く**を選択します。

SAS Event Stream Processing Studio Modeler が表示されます。

注: プロジェクトの新しいバージョンを作成するには、そのバージョンが有効な状態で存在する必要があります。

2 をクリックします。

バージョン管理ページが表示されます。このページには、作業中のプロジェクトのバージョンを表示するバージョン階層が含まれています。

3 をクリックします。

パブリッシュ - バージョンウィンドウが表示されます。

a **バージョンのメモ**フィールドに、パブリッシュするプロジェクトのバージョンに関連するメモを入力します。これにより、プロジェクトのバージョン履歴の記録を維持することができます。

注: すでにパブリッシュされているプロジェクトバージョンのバージョンメモは変更できません。

b **OK** をクリックします。

バージョン管理ページには、パブリッシュされたプロジェクトバージョンがバージョン階層に表示されます。

4 パブリッシュしたプロジェクトバージョンに関する情報を表示するには、左側のバージョン階層で関連するバージョンを選択します。

プロジェクトバージョンをパブリッシュすると、そのバージョンの XML コードが更新され、一意の ID 番号が表示されます。プロジェクトの ID 番号、メジャーバージョン番号、およびマイナーバージョン番号は、バージョンの XML コードでメタデータとして指定されています。詳細については、“プロジェクトの操作の概要”を参照してください。

プロジェクトのパブリッシュ済みバージョンの表示

1 **プロジェクト**ペインで、関連するプロジェクトを右クリックし、 **プロジェクトを開く**を選択します。

SAS Event Stream Processing Studio Modeler が表示されます。

2 をクリックします。

バージョン管理ページが表示されます。このページには、プロジェクトの現在および以前のバージョンを含むバージョン階層が表示されます。

3 左側のバージョン階層で、SAS Event Stream Processing Studio Modeler で表示するプロジェクトのバージョンを選択します。

4 をクリックします。

読み取り専用モードで表示されたパブリッシュ済みバージョン

注: すでにパブリッシュされているプロジェクトのバージョンを変更することはできません。プロジェクトをさらに変更する場合は、SAS Event Stream Processing Studio で編集可能な新しい作業コピーが用意されています。

以前のバージョンに戻す

1 **プロジェクト**ペインで、関連するプロジェクトを右クリックし、 **プロジェクトを開く**を選択します。

SAS Event Stream Processing Studio Modeler が表示されます。

2 をクリックします。

バージョン管理ページが表示されます。このページには、プロジェクトの現在および以前のバージョンを含むバージョン階層が表示されます。

3 左側のバージョン階層で、元に戻すプロジェクトのバージョンを選択します。

4 をクリックします。

バージョンに戻すウィンドウが表示されます。

5 **はい**をクリックして、元に戻すことを確認します。

プロジェクトの作業バージョンが、バージョン階層で選択したパブリッシュ済みバージョンに戻ります。この操作は元に戻すことはできません。

以前にパブリッシュ済みのバージョンのエクスポート

1 **プロジェクト**ペインで、関連するプロジェクトを右クリックし、 **プロジェクトを開く**を選択します。

SAS Event Stream Processing Studio Modeler が表示されます。

2 をクリックします。

バージョン管理ページが表示されます。このページには、プロジェクトの現在および以前のバージョンを含むバージョン階層が表示されます。

3 左側のバージョン階層で、エクスポートするプロジェクトのバージョンを選択します。

4 をクリックします。

プロジェクトバージョンがコンピュータにエクスポートされます。エクスポートしたプロジェクトバージョンの場所は、ブラウザの構成によって異なる場合があります。

検疫されたプロジェクトのバージョン

SAS Viya プラットフォームのファイル™サービスは、ポリシーで禁止されている悪意のあるコンテンツや絶対 URL のファイルをスキャンするように構成できます。このような構成が設定されている場合、SAS Event Stream Processing Studio でプロジェクトバージョンをパブリッシュすると、悪意のあるコンテンツと許可されていない URL がないかスキャンされます。不審なコンテンツが検出されると、SAS Event Stream Processing Studio の**バージョン管理**ページにプロジェクトのバージョンが検疫されていることを通知するメッセージが表示されます。SAS Event Stream Manager に検疫されたプロジェクトのバージョンは利用できません。

SAS Event Stream Processing Studio に検疫されたプロジェクトバージョンを元に戻したり、エクスポートしたりできるのは管理者のみです。管理者がファイルサービスを使用して検疫されたファイルを管理する方法と、スキャンを有効にするために使用されるプロパティの詳細については、"Files Service" (*SAS Viya Platform: Configuration Properties*)を参照してください。

例

サンプルプロジェクトのインストール

SAS Event Stream Processing のサンプルウィンドウでは、SAS Event Stream Processing Examples GitHub のサンプルの GitHub から、サンプルをインストールできます。

サンプルをインストールするには、次のようにします。

1 **プロジェクト**ページで、サンプルプロジェクトの表　をクリックします。

2 左ペインで例を選択します。

選択した例の動作の詳細については、右側のペインで **GitHub の README を表示**をクリックします。サンプルをインストールすると、**プロジェクトパッケージ**ペインからその README ファイルにアクセスすることもできます。

3 (オプション)このプロジェクトに関連付けられている大きなファイルの保存に Git LFS を使用する場合は、**Large File Storage(LFS)の使用**チェックボックスを選択します。詳細については、"ファイルベースのバージョン管理と Git Large File Storage"を参照してください。

注: Git LFS は、プロジェクトパッケージが次のいずれかの方法で保存されている場合に使用できます。

- SAS Event Stream Processing の内部にある Git リポジトリ
- Gitea のリモートリポジトリ

4 (オプション)**リモートリポジトリの作成**チェックボックスを選択して、サンプルプロジェクトをリモート Git リポジトリに保存できます。**リモートサーバー接続**ドロップダウンリストを使用して、リポジトリを作成するリモート Git サーバーを選択します。詳細については、"ファイルベースのバージョン管理"を参照してください。

詳細については、GitHub のメインの README ファイルの使用例を参照してください。

トレードの処理

概要

この例では株式市場のトレードを処理します。このモデルは、大規模な証券取引およびそのトレードに関与したトレーダーを特定します。

次の手順では、SAS Event Stream Processing Studio を使用してプロジェクトを最初から作成し、そのプロジェクトをテストする方法を説明します。

> ヒント　あるいは、プロジェクトの既製バージョンをインストールしてテストすることもできます。Trades サンプルをインストールするには、"サンプルプロジェクトのインストール"の手順に従います。次に、"モデルをテストモードで実行する"の手順に従います。

この例では、https://github.com/sassoftware/esp-studio-examples/tree/main/EndtoEndExamples/trades で入手可能なファイルを使用します。GitHub から次のファイルを取得し、コンピューターに保存します。

- **trades.csv** は入力ファイルです。このファイルには証券取引に関するイベントが含まれています。
- **traders.csv** は入力ファイルです。このファイルには証券取引に関わるトレーダーに関するイベントが含まれています。

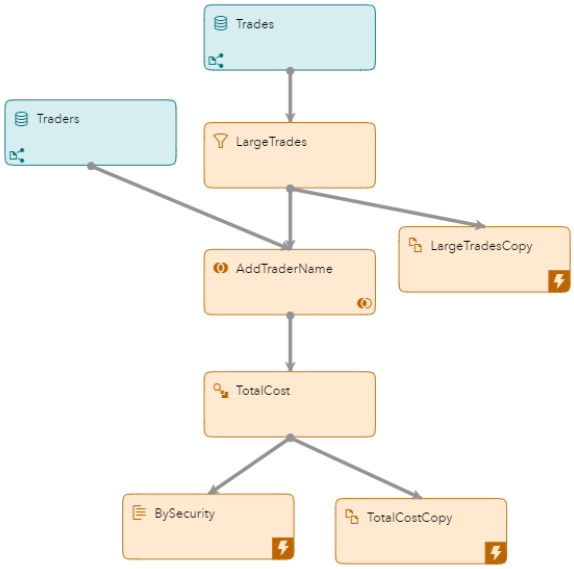
入力データをプロジェクトにフィードするには、いくつかの方法があります。この手順では、入力ファイルを指すようにファイルコネクタとソケットコネクタを構成する方法、およびそれらの入力ファイルをコンピュータからプロジェクトパッケージにアップロードする方法について説明します。

この手順では、Kubernetes 環境を使用していることを前提としています。つまり、エッジサーバー上で実行される ESP サーバー上でプロジェクトをテストしているわけではありません。

プロジェクト詳細

次の図は、プロジェクトのダイアグラムを示しています。

図 32　処理トレードプロジェクトのダイアグラム



このプロジェクトには 8 つのウィンドウが含まれています。

- トレードはソースウィンドウです。ここで、**trades.csv** ファイルの証券がモデルに入力されます。
- トレーダーはソースウィンドウです。ここで、**traders.csv** ファイルの取引名がモデルに入力されます。
- LargeTrades はフィルターウィンドウです。このウィンドウは、指定された範囲内にないすべてのトレードを除外します。
- LargeTradesCopy はコピーウィンドウです。このウィンドウには、すべての大きな取引のリストが保持されます。
- AddTraderName は結合ウィンドウです。このウィンドウは、大規模トレードとそれに対応するトレーダー名を組み合わせたものです。
- TotalCost は計算ウィンドウです。このウィンドウには、各取引の合計コストが表示されます。この情報を使用して、価値の高い取引を特定することができます。
- TotalCostCopy はコピーウィンドウです。このウィンドウには、保持ポリシーに従って各トランザクションの合計コストのリストが保持されます。
- BySecurity は集計ウィンドウです。このウィンドウには、大規模トレードのすべての挿入、削除、および更新ブロックが表示されます。

ステップの例

プロジェクトの作成

1　**プロジェクト**ページで、をクリックします。

　新しいプロジェクトウィンドウが表示されます。

2　**新しいプロジェクト**ウィンドウで、次の操作を行います。

a **名前**フィールドに、**trades_lua** と入力します。

b **説明**フィールドに説明を入力します。**このモデルは、大規模な証券取引とそれらのトレードに関与したトレーダーを特定するために使用できます。**

c **プロジェクトパッケージの作成**チェックボックスがまだ選択されていない場合は選択します。この例の後半では、入力ファイルをプロジェクトパッケージにアップロードします。プロジェクトパッケージの目的に関する詳細については、“プロジェクトパッケージ”を参照してください。

d **OK** をクリックします。

プロジェクトのスレッドレベルと連続クエリの構成

1 右ペインには、プロジェクトレベルのプロパティが表示されます。右ペインで、**属性**を展開します。

2 **スレッド**フィールドで、スレッドプールサイズを **4** に変更します。

これは、使用可能なスレッドプールから 4 つのスレッドが使用されることを意味します。プロジェクトでスレッドプールを使用すると、より効率的な処理が可能になります。

3 をクリックして、プロジェクトの連続クエリの設定を表示します。

4 右ペインの**名前**フィールドで、連続クエリのデフォルト名 **cq1** を **trades_cq** に変更します。

入力イベントを受信するソースウィンドウの構成

プロジェクトが作成されると、ソースウィンドウがワークスペースに 自動的に追加されました。

証券取引に関するイベントを受信するようにソースウィンドウを構成するには:

1 ソースウィンドウの名前を変更します。右ペインの**名前**フィールドで、デフォルト名を **Trades** に変更します。

2 Trades ウィンドウの出力スキーマを指定します。

a 右ペインで、をクリックします。

b をクリックします。

出力スキーマウィンドウが表示されます。

c スキーマテーブルには、キーフィールドの行がすでに 1 つ含まれています。スキーマテーブルに行を追加するには、をクリックします。

次の表に示す値を行に入力します。つまり、デフォルトで存在していた行の値を調整し、追加した新しい行に値を追加します。

キー	フィールド名	種類
Y	tradeID	文字列
N	security	文字列

キー	フィールド名	種類
N	quantity	int32
N	price	double
N	traderID	int64
N	time	stamp

d **OK** をクリックします。

e 右ペインで、をクリックします。

3 証券取引を含む **trades.csv** ファイルからイベントをストリーミングするように取引ウィンドウを構成します。

a **入力データ(パブリッシャー)コネクタ**を展開します。

b をクリックします。

コネクタ構成ウィンドウが表示されます。

c **名前**フィールドに **input_trades** と入力します。

d デフォルトでは、コネクタの種類がファイルおよびソケットコネクタに設定されていることに注目してください。この例では、CSV ファイルを使用してデータを例にフィードするため、このコネクタの種類が適しています。

e **Fsname** フィールドで、をクリックします。

ファイルの選択ウィンドウが表示されます。

f をクリックして **trades.csv** ファイルをアップロードします。

ファイルはプロジェクトパッケージの **test_files** フォルダーにアップロードされます。

g **OK** をクリックします。

h **Fstype** ドロップダウンリストで、**csv** を選択します。

i input_trades コネクタのプロパティを構成します。

1 **すべてのプロパティ** をクリックします。

すべてのプロパティ ウィンドウが表示されます。

2 **dateformat** プロパティの**値**フィールドに**%d/%b/%Y:%H:%M:%S** と入力します。

3 **レート**プロパティの**値**フィールドに **100** を入力します。

4 **OK** をクリックします。

j **OK** をクリックします。

4 Trades ウィンドウの状態とイベントの種類を指定します。

a **状態とイベントの種類**を展開します。

b **ウィンドウの状態とインデックス**ドロップダウンリストで、**ステートレス(pi_EMPTY)**を選択します。

プライマリインデックスタイプ pi_EMPTY を使用するウィンドウは、非ステートフルまたはステートレスです。すべての受信イベントのパススルーとして機能します。このインデックスはイベントを格納しません。

c **Insert イベントのみを受け入れる**チェックボックスを選択します。

このオプションは、ウィンドウが非挿入イベントを他のウィンドウに渡すことを防ぎます。

入力イベントを受信する別のソースウィンドウの作成

1 左の**ウィンドウ**ペインに**入力ストリーム**を展開し、ワークスペースに 別のソースウィンドウをドラッグします。

右ペインにはソースウィンドウのプロパティが表示されます。

このウィンドウは株式市場のトレーダーに関する情報を受け取ります。

2 ソースウィンドウの名前を指定します。右ペインの**名前**フィールドで、デフォルト名を **Traders** に変更します。

3 Traders ウィンドウの出力スキーマを指定します。

a 右ペインで、をクリックします。

b をクリックします。

出力スキーマウィンドウが表示されます。

c スキーマテーブルに行を追加するには、をクリックします。行を追加したら、もう一度をクリックして次の行を追加します。

次の値を入力します。

キー	説明	種類
Y	ID	int64
N	name	文字列

d **OK** をクリックします。

e 右ペインで、をクリックします。

4 **traders.csv** ファイルから情報を受信するように Traders ウィンドウを構成します。このファイルには株式市場トレーダーの詳細が含まれています:

a **入力データ(パブリッシャー)コネクタ**を展開します。

b をクリックします。

コネクタ構成ウィンドウが表示されます。

c **名前**フィールドに **input_trades** と入力します。

d **Fsname** フィールドで、をクリックします。

ファイルの選択ウィンドウが表示されます。

e をクリックし **traders.csv** ファイルをアップロードします。

ファイルはプロジェクトパッケージの **test_files** フォルダーにアップロードされます。

f **OK** をクリックします。

g **Fstype** ドロップダウンリストで、**csv** を選択します。

h **OK** をクリックします。

5 Traders ウィンドウの状態とイベントの種類を指定します。

a **状態とイベントの種類**を展開します。

b **ウィンドウの状態とインデックス**ドロップダウンリストで、**ステートレス(pi_EMPTY)**を選択します。

コネクタのオーケストレーションの追加

コネクタのオーケストレーションを追加して、トレーダーウィンドウからのイベントがトレーダーウィンドウからのイベントより前にプロジェクトに入力されるようにします。詳細については、*"オーケストレーションコネクタ" (SAS Event Stream Processing: コネクタとアダプター)*を参照してください。

1 をクリックし、特定のウィンドウのプロパティではなく、プロジェクトレベルのプロパティを表示します。

2 右ペインで、**コネクタオーケ**を展開します。

3 取引ウィンドウのコネクタグループを作成します:

a **コネクタグループ**テーブルで、をクリックします。

b **コネクタグループ**ウィンドウで、デフォルト名を **Trades に**に変更します。

c をクリックします。

d コネクタ列で、 **trades_cq/Trades/input_trades** を選択します。

e ターゲットの状態列で、ターゲットの状態が**完了**であることを確認します。

f **OK** をクリックします。

4 Traders ウィンドウのコネクタグループを作成します:

a **コネクタグループ**テーブルで、をクリックします。

b **コネクタグループ**ウィンドウで、デフォルト名を **Traders** に変更します。

c をクリックします。

d コネクタ列で、 **trades_cq/Traders/input_trades** を選択します。

e ターゲットの状態列で、ターゲットの状態が**完了**であることを確認します。

f **OK** をクリックします。

5 コントロールグループの設定:

a **依存関係ルール**テーブルで、をクリックします。

b コントロールグループ列で、**トレーダー**を選択します。

c 依存グループ列で、**トレード**を選択します。

トレーダー ウィンドウのコネクタのステータスが**完了**になるまで、トレード ウィンドウのイベントはプロジェクトに入力されなくなりました。

フィルタウィンドウを作成して大規模トレードをフィルタリングする

1 左の**ウィンドウ**ペイン上で**変換**を展開し、ワークスペースにフィルターウィンドウをドラッグします。

このウィンドウを構成して、大規模なトレードを識別します。この例では、取引される株式の数量が 100 に等しいかそれを超えると、トレードは大規模なトレードとみなされます。

2 右ペインで、フィルターウィンドウの名前を指定します。**名前**フィールドで、デフォルト名を **LargeTrades に**変更します。

3 Trades ウィンドウを LargeTrades ウィンドウにエッジを使用して接続します。

a カーソルを Trades ウィンドウの下部にあるアンカーポイントの上に置くと、アンカーポイントの色が白に変わります。

b 白いアンカーポイントをクリックし、マウスボタンを押したままにして、LargeTrades ウィンドウのアンカーポイントに線を引いてください。

LargeTrades ウィンドウは、Trades ウィンドウからのトレードを受け入れるようになりました。

4 ワークスペースの LargeTrades ウィンドウをクリックします。

5 右ペインで、**フィルター**を展開します。

6 フィルタータイプを Lua に設定します。

7 **Lua コードで使用するフィールド**フィールドの**フィールドを含める**ラジオボタンの下にあるドロップダウンリストで、**数量**フィールドを選択します。

8 **フィルター関数**フィールドが**フィルター**に設定されていることを確認します。このフィールドは、フィルター関数として機能する Lua コードアクションの関数を識別します。

9 Lua コード領域で、デフォルトのコードを次のコードに置き換えます。

```
function filter(data)
    return data.quantity >= 100
end
```

10 をクリックして、Lua コードを検証します。

コードの有効性を通知するメッセージが表示されます。

11 LargeTrades ウィンドウの状態を設定します。

a **状態**を展開します。

b **ウィンドウ状態とインデックス**フィールドで、ドロップダウンリストから**ステートレス (pi_EMPTY)**を選択します。

結合ウィンドウを作成して大規模トレードを対応するトレーダーと結合する

1 左の**ウィンドウ**ペイン上で**変換**を展開し、結合ウィンドウをワークスペースにドラッグします。

このウィンドウを構成して、大規模なトレードを、そのトレードを行ったトレーダーと一致させます。

2 結合ウィンドウの名前と説明を指定します。右ペインの**名前**フィールドで、デフォルト名を **AddTraderName** に変更します。

3 LargeTrades ウィンドウを AddTraderName ウィンドウにエッジを使用して接続します。

AddTraderName ウィンドウは、LargeTrades ウィンドウからのトレードを受け入れるようになりました。

4 Traders ウィンドウを AddTraderName ウィンドウにエッジを使用して接続します。

AddTraderName ウィンドウは Traders ウィンドウからトレーダー名を受け入れるようになりました。

5 ワークスペースの AddTraderName ウィンドウをクリックします。

右ペインに AddTraderName ウィンドウのプロパティが表示されます。

6 AddTraderName ウィンドウの構成設定を確認します。

a 右ペインで**設定**を展開し、LargeTrades ウィンドウが左側のウィンドウとみなされ、Traders ウィンドウが右側のウィンドウとみなされていることを確認します。これは、エッジを追加した順序によるものです。

b **出力フィールド計算メソッド**フィールドで、ドロップダウンリストから**フィールド選択**が選択されていることを確認します。

選択フィールドオプションは、新しい入力イベントが到着すると、非キーフィールドの結合は結合選択文字列を使用して計算されていることを保証します。この選択文字列は結合フィールドと入力フィールドの 1 対 1 のマッピングです。

c **設定**を折りたたみます。

7 AddTraderName ウィンドウの結合条件を構成します。

a **結合条件**を展開します。

b **結合条件**セクションで、  をクリックしてテーブルに行を追加します。

c 左側のセルをクリックします。LargeTrades 列を選択し、**traderID** を選択します。

d 右側のセルをクリックします。トレーダー列で **ID** を選択します。

e **結合条件**を折りたたみます。

8 AddTraderName ウィンドウの結合基準を確認します。

a **結合基準**がまだ展開されていない場合は展開します。

b **結合の種類**ドロップダウンリストのデフォルト値が**左側**であることに注意してください。

c **"no-regenerates"オプションを設定**チェックボックスを選択します。

このオプションの詳細については、"Using Regeneration versus No Regeneration" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

9 AddTraderName ウィンドウの状態を設定します。

a **状態**を展開します。

b **ウィンドウ状態とインデックス**フィールドで、ドロップダウンリストから**ステートレス (pi_EMPTY)**を選択します。

10 AddTraderName ウィンドウの出力ルールを構成します。

a **出力ルール**を展開します。

b **重複する更新を折りたたむ**チェックボックスをオフにします。チェックボックスをオフにすると、複数の更新ブロックが単一の更新ブロックに折りたたまれなくなります。

11 AddTraderName ウィンドウのスキーマを指定します。

a 右ペインで、をクリックします。

b をクリックします。

出力スキーマの編集 - キー以外のフィールドウィンドウが表示されます。このウィンドウを使用して、次の表に示すフィールドを構成します。必要なスキーマフィールドはすでに定義済みです。をクリックして、**入力スキーマからフィールドをコピー**ウィンドウを開きます。次のスキーマフィールドを選択します。

ウィンドウ	フィールド	種類
LargeTrades	security	文字列
LargeTrades	quantity	int32
LargeTrades	price	double
LargeTrades	traderID	int64
LargeTrades	time	stamp
Traders	name	文字列

c **OK** をクリックします。

出力スキーマの編集 – キー以外のフィールドウィンドウに戻ります。

d **OK** をクリックします。

12 右ペインで、をクリックします。

コピーウィンドウを作成して大規模トレードを保持する

1 左の**ウィンドウ**ペイン上で**変換**を展開し、コピーウィンドウをドラッグします。

右ペインにはコピーウィンドウのプロパティが表示されます。

2 コピーウィンドウの名前と説明を指定します。右ペインの**名前**フィールドで、デフォルト名を **LargeTradesCopy** に変更します。

3 LargeTradesCopy ウィンドウの保持プロパティを設定します。

a **保持**を拡大します。

b **種類**フィールドが**時間による、スライド**に設定されていることを確認します。これはデフォルト値です。

c **時間の制限**フィールドに **5** を入力し、ドロップダウンリストから**分**を選択します。

4 LargeTradesCopy ウィンドウの出力ルールを構成します。

a **出力ルール**を展開します。

b **重複する更新を折りたたむ**チェックボックスをオフにします。チェックボックスをオフにすると、複数の更新ブロックが単一の更新ブロックに折りたたまれなくなります。

5 LargeTrades ウィンドウを LargeTradesCopy ウィンドウにエッジを使用して接続します。

LargeTradesCopy ウィンドウは、LargeTrades ウィンドウからのトレードを受け入れるようになりました。

計算ウィンドウを作成して各トランザクションの合計コストを計算する

1 左の**ウィンドウ**ペイン上で**変換**を展開し、ワークスペースへの 計算ウィンドウをドラッグします。

右ペインに計算ウィンドウのプロパティが表示されます。

このウィンドウを構成して、大規模なトレードの総コストを計算します。

2 計算ウィンドウの名前を指定します。**名前**フィールドで、デフォルト名を **TotalCost** に変更します。

3 AddTraderName ウィンドウを TotalCost ウィンドウにエッジを使用して接続します。

TotalCost ウィンドウは、AddTraderName ウィンドウからのトレードを受け入れるようになりました。

4 TotalCost ウィンドウをクリックして、ウィンドウのプロパティを右ペインに再度表示します。

5 TotalCost ウィンドウの状態を設定します。

a 右ペインで、**状態**を展開します。

b **ウィンドウ状態とインデックス**フィールドで、ドロップダウンリストから**ステートレス (pi_EMPTY)**を選択します。

6 TotalCost ウィンドウの出力スキーマを指定します。

a 右ペインで、  をクリックします。

b をクリックします。

出力スキーマウィンドウが表示されます。

c スキーマテーブルに行を追加するには、をクリックします。

次の値を入力します。

注: あるいは、以前に定義したスキーマフィールドをコピーすることもできます。をクリックして、**入力スキーマからフィールドをコピー**ウィンドウを開きます。コピーするスキーマフィールドを選択し、**OK** をクリックします。

以前に作成したスキーマフィールドをコピーした場合でも、すべての新しいフィールドとその値を手動で入力する必要があります。

キー	フィールド名	種類	式
Y	tradeID	文字列	(なし)
N	security	文字列	security
N	quantity	int32	quantity
N	price	double	price
N	totalCost	double	price*quantity
N	traderID	int64	traderID
N	time	stamp	time
N	name	文字列	名前

d **OK** をクリックします。

7 右ペインで、をクリックします。

コピーウィンドウを作成して合計計算額を保持する

1 左の**ウィンドウ**ペイン上で**変換**を展開し、コピーウィンドウをドラッグします。

右ペインにはコピーウィンドウのプロパティが表示されます。

2 コピーウィンドウの名前と説明を指定します。右ペインの**名前**フィールドで、デフォルト名を **TotalCostCopy** に変更します。

3 TotalCostCopy ウィンドウの保持プロパティを設定します。

a **保持**を拡大します。

b **種類**フィールドが**時間による、スライド**に設定されていることを確認します。

c **時間の制限**フィールドに **1** を入力し、ドロップダウンリストから**日**を選択します。

4 TotalCostCopy ウィンドウの出力ルールを構成します。

a **出力ルール**を展開します。

b **重複する更新を折りたたむ**チェックボックスをオフにします。チェックボックスをオフにすると、複数の更新ブロックが単一の更新ブロックに折りたたまれなくなります。

5 TotalCost ウィンドウを TotalCostCopy ウィンドウにエッジを使用して接続します。

TotalCostCopy ウィンドウは、TotalCost ウィンドウからのトレードを受け入れるようになりました。

集計ウィンドウを作成して大規模トレードとその合計コストを集計する

1 左の**ウィンドウ**ペイン上で**変換**を展開し、ワークスペースに集計ウィンドウをドラッグします。

右ペインには、集計ウィンドウのプロパティが表示されます。

このウィンドウを構成して、大規模なトレードの総コストを計算します。

2 集計ウィンドウの名前を指定します。**名前**フィールドで、デフォルト名を **BySecurity** に変更します。

3 BySecurity ウィンドウの出力ルールを構成します。

a **出力ルール**を展開します。

b **重複する更新を折りたたむ**チェックボックスをオフにします。チェックボックスをオフにすると、複数の更新ブロックが単一の更新ブロックに折りたたまれなくなります。

4 TotalCostCopy ウィンドウを BySecurity ウィンドウにエッジを使用して接続します。

BySecurity ウィンドウは、TotalCostCopy ウィンドウからのトレードを受け入れるようになりました。

5 BySecurity ウィンドウをクリックして、右ペインに集計ウィンドウのプロパティを再度表示します。

6 BySecurity ウィンドウの出力スキーマを指定します。

a 右ペインで、をクリックします。

b をクリックします。

出力スキーマウィンドウが表示されます。

c スキーマテーブルに行を追加するには、をクリックします。次の値を入力します。

キー	フィールド名	種類	集計関数	パラメーター
Y	security	文字列	(なし)	(なし)
N	quantityTotal	double	ESP_aSum	quantity

キー	フィールド名	種類	集計関数	パラメーター
N	costTotal	double	ESP_aSum	totalCost

d **OK** をクリックします。

モデルをテストモードで実行する

1 モデルは完成しました。をクリックして、モデルを保存します。

2 ✓テストモードへ移行をクリックします。

テスト: trades_proj_lua という新しいページが表示されます。

3 ▷ テストの実行をクリックします。

> ヒント　Kubernetes クラスターで ESP サーバーを作成するために使用される設定を調整する場合、▷ テストの実行をクリックする代わりに、▷ テストの実行の横にある▾をクリックし、テストの構成と実行　をクリックします。**プロジェクトをクラスターにロードして開始**ウィンドウが表示されます。必要に応じて設定を調整し、**OK** をクリックします。このウィンドウで使用できる設定の詳細については、"Kubernetes クラスター内にある ESP サーバーでのプロジェクトのロードと開始"を参照してください。

各ウィンドウの結果は、別々のタブに表示されます。

- **Trades** タブには、証券取引のリストが表示されます。
- **Traders** タブには、トレーダーのリストが表示されます。
- **LargeTrades** タブには、大規模なトレードのリストが表示されます。
- **AddTraderName** タブには、大規模なトレードのリストが表示され、トレーダー名を示す追加の列が表示されます。
- **LargeTradesCopy** タブには、保持ポリシーに従って保持された大規模取引が一覧表示されます。
- **TotalCost** タブには、各トランザクションの合計コストを示す追加の列が含まれています。この情報を使用して、価値の高い取引を特定することができます。
- **TotalCostCopy** タブには、保持ポリシーに従って、各トランザクションの保持された合計コストが一覧表示されます。
- **BySecurity** タブには、大規模なトレードのすべての挿入、削除、および更新ブロックが表示されます。最新のイベントがテーブルの上部に表示されます。

4 テストを停止するには、■ 停止をクリックします。

SAS Model Manager で作成したモデルの SAS Event Stream Processing Studio へのインポート

SAS Model Manager の統合の概要

モデルを SAS Model Manager から SAS Event Stream Processing Studio へインポートするには 、次の方法のいずれかを選択します。

- “モデルペインを使用して SAS Model Manager からモデルをインポート”
- “SAS Model Manager ZIP ファイルのインポート”
- “モデルの SAS Model Manager から SAS Event Stream Processing へのパブリッシュ”

モデルペインを使用して SAS Model Manager からモデルをインポート

この方法でモデルをインポートする方法は、DS2 モデルと Python モデルで使用できます。

SAS Event Stream Processing Studio の**モデル**ペインを使用して、SAS Model Manager からモデルをインポートできます。プロジェクトが配置されると、そのモデルは SAS Model Manager 共通モデルリポジトリから取得され、ESP サーバーを実行する Kubernetes ポッドに書き込まれます。

- DS2 モデルをインポートすると、モデルに対応するために SAS Micro Analytic Service モジュールが作成されます。ソースウィンドウと計算ウィンドウが作成されます。SAS Micro Analytic Service モジュールは、計算ウィンドウの入力ハンドラーから参照されます。その後、SAS Event Stream Processing Studio を使用して新しいチャンピオンモデルをチェックして受け入れます。詳細については、“SAS Micro Analytic Service モジュールの更新”を参照してください。
- Python モデルをインポートすると、ソースウィンドウと Python ウィンドウが作成されます。モデルのスコアコードが Python ウィンドウにコピーされます。

  注: Python モデルをインポートする場合、SAS Event Stream Processing Studio は新しいチャンピオンモデルを確認できません。SAS Model Manager で新しいチャンピオンモデルを識別する場合、モデルを SAS Event Stream Processing Studio にインポートするプロセスを繰り返すことができます。

> 重要　SAS Model Manager からモデルをインポートするこの方法では、SAS Event Stream Processing を他の SAS 製品と共に展開するときにオーバーレイを使用する必要があります。これらのオーバーレイは、永続ボリュームと分析ストアにアクセスするように SAS Event Stream Processing を構成するために使用されます。詳細については、*"永続ボリュームの管理" (SAS Event Stream Processing: Kubernetes 環境での SAS Event Stream Processing の使用)*を参照してください。

SAS Model Manager からインポートするモデルは、次の条件を満たしている必要があります:

- モデルが SAS Model Manager プロジェクトに存在。
- モデルがプロジェクトのチャンピオン。
- モデルのスコアコードの種類は、次の基準を満たす DATA ステップ、DS2 パッケージ、DS2 マルチタイプ、または Python です。
  - DATA ステップ、DS パッケージ、または Python モデルが、スコアコードのファイルロールを持つスコアファイルを含みます。
  - DS2 マルチタイプファイルには、 **dmcas_packagescorecode.sas** という名前の DS2 スコアコードファイルを含む分析ストアコンテンツが含まれています。
- DS2 モデルの場合、モデルには、単一の DS2 コードファイル(DS2 パッケージ)、または 1 つ以上の分析ストアファイルを含む DS2 コードファイルが含まれます。

モデルペインを使用してモデルをインポートするには、次の操作を実行します。

1. SAS Event Stream Processing Studio において、モデルをインポートするプロジェクトを開きます。プロジェクトはプロジェクトパッケージの一部である必要があります。詳細については、"プロジェクトパッケージの作成"を参照してください。
2. 左側のツールバーの  をクリックします。SAS Model Manager が使用できない場合、またはプロジェクトがプロジェクトパッケージの一部ではない場合、このボタンはツールバーに表示されません。
3. **モデル**ペインで、検索およびフィルタリング機能を使用して、インポートするモデルを見つけます。たとえば、  をクリックしてリポジトリ別にモデルをフィルターし、SAS Model Manager でプロジェクトの名前を入力して検索をさらに絞り込みます。また、モデルをダブルクリックすると、その詳細情報が表示されます。
4. 目的のモデルを**モデル**ペインからワークスペースにドラッグします。
5. モデルに **requirements.txt** ファイルが含まれており、プロジェクトの仮想環境がファイル内の要件を満たしていない場合、ツールバーに警告が表示されます。**解決**をクリックし、**仮想環境のファイルの検出**ウィンドウを使用して、適切な仮想環境を選択します。詳細については、"仮想環境ファイルが検出された場合に一致する環境を選択する"を参照してください。
6. モデルが DS2 モデルの場合、ソースウィンドウと計算ウィンドウがワークスペースに 表示されます。計算ウィンドウは、モデルに対応するために作成された SAS Micro Analytic Service モジュールを参照します。モジュールの詳細を表示するには、  をクリックし、**SAS Micro Analytic Service modules** モジュールを展開して、SAS Model Manager プロジェクトの名前をダブルクリックします。
7. モデルが Python モデルの場合、ワークスペースにソースウィンドウと Python ウィンドウが表示されます。Python ウィンドウには、モデルのスコアコードが含まれています。コードを表示する

には、Python ウィンドウを選択し、右ペインで **Python 設定**を展開し、Python コードと関連する設定を表示します。

注: インポートされたモデルのスコアコードには、エントリポイント関数の戻り値が、期待される出力変数名をキーとするディクショナリであることを確認するなどの追加の構成が必要になる場合があります。

他のモデルファイルはプロジェクトパッケージに追加されます。これらを表示するには、左側のペインで をクリックし、**プロジェクトパッケージ**ペインで**分析**フォルダーを展開します。詳細については、“プロジェクトパッケージに含まれるフォルダーの使用”を参照してください。

8 必要に応じてプロジェクトの残りの部分を構成し、 をクリックします。

SAS Model Manager ZIP ファイルのインポート

モデルをインポートするこの方法は、DS2 モデルで使用できます。

ESP プロジェクトは、からエクスポートしたモデルを参照できます SAS Model Manager から ZIP ファイルとしてエクスポートし、SAS Event Stream Processing Studio のプロジェクトパッケージにインポートしたモデルを参照できます。プロジェクトが配置されると、モデルは ESP サーバーを実行する Kubernetes ポッドに書き込まれます。SAS Micro Analytic Service モジュールは、このようなモデルに対応するために使用されます。モジュールは、計算ウィンドウの入力ハンドラーから参照されます。

注: この方法でモデルをインポートすると、SAS Event Stream Processing Studio は新しいチャンピオンモデルを選択できません。SAS Model Manager で新しいチャンピオンモデルを識別する場合、モデルを SAS Event Stream Processing Studio にインポートするプロセスを繰り返すことができます。

SAS Model Manager からインポートするモデルは、次の条件を満たしている必要があります:

- モデルが SAS Model Manager プロジェクトに存在。
- モデルがプロジェクトのチャンピオン。
- モデルのスコアコードの種類は、次の基準を満たす DATA ステップ、DS2 パッケージ、または DS2 マルチタイプです。
  - DS2 パッケージモデルが、スコアコードのファイルロールを持つスコアファイルを含みます。
  - DS2 マルチタイプファイルには、 **dmcas_packagescorecode.sas** という名前の DS2 スコアコードファイルを含む分析ストアコンテンツが含まれています。
- DS2 モデルの場合、モデルには、単一の DS2 コードファイル(DS2 パッケージ)、または 1 つ以上の分析ストアファイルを含む DS2 コードファイルが含まれます。

SAS Model Manager ZIP ファイルをインポートする方法次のとおりです。

1 モデルのスコアコードの種類が DATA ステップの場合、SAS Model Manager を使用してモデルを SAS Micro Analytic Service のパブリッシュ先にパブリッシュします。モデルを公開すると、DATA ステップコードが DS2 コードに変換されます。詳細については、“Publish a Project Champion Model” (*SAS Model Manager: User's Guide*)を参照してください。

次のファイルがモデルに追加されます。**ScoreCodeGen-ds2Package_code.sas**、 **ScoreCodeGen-ds2Package.log**、および **ScoreCodeGen-ds2Package.sas**。

2 SAS Model Manager からモデルを ZIP ファイルとしてエクスポートします。詳細については、"Export a Model" (*SAS Model Manager: User's Guide*)を参照してください。

3 SAS Event Stream Processing Studio において、モデルをインポートするプロジェクトを開きます。プロジェクトはプロジェクトパッケージの一部である必要があります。詳細については、"プロジェクトパッケージの作成"を参照してください。

4 右ペインで、**SAS Micro Analytic Service モジュール**を展開し、 をクリックします。

> ヒント　または、**プロジェクト パッケージ**ペインから **SAS Model Manager ZIP ファイルからのインポート**ウィンドウにアクセスすることもできます。ペインのツールバーでをクリックし、**SAS Micro Analytic Service モジュールのインポート**を選択します。

5 **SAS Model Manager ZIP ファイルからのインポート**ウィンドウで、次の手順を実行します:

a SAS Model Manager からエクスポートした ZIP ファイルを選択します。

b **名前**フィールドには、SAS Micro Analytic Service モジュールのデフォルト名が表示されます。この名前は編集できます。

c デフォルトでは、**ソース ウィンドウの追加**チェックボックスと**計算ウィンドウの追加**チェックボックスが選択されています。これらのチェックボックスを選択すると、これらのウィンドウの出力スキーマが自動的に作成され、計算ウィンドウに入力ハンドラーが追加されます。これらのウィンドウを追加したくない場合は、チェックボックスをオフにします。

d **OK** をクリックします。

6 モジュールは、右ペインの **SAS Micro Analytic Service モジュール**セクションに表示されます。

a モジュールをダブルクリックすると、**SAS Micro Analytic Service モジュール**ウィンドウでその詳細が表示されます。コードファイルは**外部ファイル**フィールドで参照され、分析ストアファイルはモジュールメンバーテーブルに 表示されます。DATA ステップコードを DS2 コードに変換した場合、**外部ファイル**フィールドは **ScoreCodeGen-ds2Package_code.esp.sas** ファイルを参照します。このファイルについては、このトピックの後半で詳しく説明します。

b **OK** をクリックして、**SAS Micro Analytic Service モジュール**ウィンドウを閉じます。

7 ソースウィンドウと計算ウィンドウを追加するためのデフォルトオプションを維持した場合、これらのウィンドウはワークスペースに 表示され、エッジで相互に接続されます。

ウィンドウを選択し、をクリックして、その出力スキーマを表示します。スキーマには、SAS Model Manager からエクスポートした ZIP ファイル内の **inputVars.json** ファイルと **outputVars.json** ファイルで指定されたフィールドが設定されます。ZIP ファイルで指定された変数は、SAS Event Stream Processing と互換性のある型に変換されます。たとえば、decimal 型は double 型です。同様に、boolean および integer 型は、int32 型に変換されます。

ソースウィンドウに、id というキーフィールドが自動的に追加されます。id フィールドの存在は、プロジェクトを実行する準備ができていることを意味します。必要に応じて、id フィールドを調整できます。SAS Micro Analytic Service モジュールがプロジェクト内の他のウィンドウで機能するには、スキーマをさらに調整する必要がある場合があります。ソースウィンドウと計算ウィンドウの間にウィンドウを追加する必要がある場合もあります。

8 左ツールバーのをクリックします。

9 **プロジェクトパッケージ**ペインで、モジュールが**/home/analytics/mas-modules/** ***moduleName*** **/1.0** フォルダーに保存されていることを確認します。store-location ESP サーバー構成プロパティはが別の場所を指定している場合、モジュールは常にこの場所に保存されます。

DATA ステップコードを DS2 コードに変換した場合、プロジェクトパッケージには **ScoreCodeGen-ds2Package_code.esp.sas** ファイルが含まれます。このファイルは、SAS Model Manager によって生成された DS2 コードに、次のパッケージ名のプレースホルダーが含まれている可能性があるため必要です: package &SCOREPACK/overwrite=yes;。このプレースホルダーは、ESP サーバーランタイムと互換性がありません。DS2 スコアコードの変更されたコピーは、**ScoreCodeGen-ds2Package_code.esp.sas** ファイルに保存されます。

10 をクリックします。

モデルの SAS Model Manager から SAS Event Stream Processing へのパブリッシュ

モデルを SAS Model Manager から SAS Event Stream Processing へパブリッシュできます。

ソースウィンドウと計算ウィンドウを含むプロジェクトパッケージは、SAS Event Stream Processing Studio で作成されます。モデルに対応するために SAS Micro Analytic Service モジュールが作成され、このモジュールは計算ウィンドウの入力ハンドラーから参照されます。

また、プロジェクトコンテナーは、SAS Event Stream Manager に作成されます。詳細については、"Working with Project Containers" (*SAS Event Stream Processing: Using SAS Event Stream Manager*)を参照してください。

注: この方法でモデルをパブリッシュすると、SAS Event Stream Processing Studio または SAS Event Stream Manager は新しいチャンピオンモデルを確認できません。SAS Model Manager で新しいチャンピオンモデルを識別する場合、モデルをパブリッシュするプロセスを繰り返すことができます。

重要　SAS Model Manager からモデルをインポートするこの方法では、SAS Event Stream Processing を他の SAS 製品と共に展開するときにオーバーレイを使用する必要があります。これらのオーバーレイは、永続ボリュームと分析ストアにアクセスするように SAS Event Stream Processing を構成するために使用されます。詳細については、"永続ボリュームの管理" (*SAS Event Stream Processing: Kubernetes 環境での SAS Event Stream Processing の使用*)を参照してください。

SAS Model Manager からパブリッシュするモデルは、次の条件を満たしている必要があります。

- モデルが SAS Model Manager プロジェクトに存在。
- モデルのスコアコードの種類は、次の基準を満たす DATA ステップ、DS2 パッケージ、または DS2 マルチタイプです。
  - DS2 パッケージモデルが、スコアコードのファイルロールを持つスコアファイルを含みます。

  - DS2 マルチタイプファイルには、 **dmcas_packagescorecode.sas** という名前の DS2 スコアコードファイルを含む分析ストアコンテンツが含まれています。
- モデルには、単一の DS2 コードファイル(DS2 パッケージ)、または 1 つ以上の分析ストアファイルを含む DS2 コードファイルが含まれています。

SAS Event Stream Manager は、プロジェクトコンテナーを作成できるコンテナーレジストリへの接続が含まれている必要があります。詳細については、"Add Container Registries" (*SAS Event Stream Processing: Using SAS Event Stream Manager*)を参照してください。コンテナーレジストリの名前は、Event Stream Processing パブリッシュ先の一部として構成されている名前と一致する必要があります。詳細については、"Configure Event Stream Processing Publishing Destinations" (*SAS Viya Platform: Publishing Destinations*)を参照してください。構成された Event Stream Processing パブリッシュ先は、SAS Model Manager の**パブリッシュ**ウィンドウで表示できます。たとえば、構成されたパブリッシュ先で使用されるコンテナーレジストリ名が Model Publishing の場合、SAS Event Stream Manager にはこの名前のコンテナーレジストリへの接続が含まれている必要があります。

モデルのパブリッシュは、次のとおりです。

1 SAS Model Manager からのモデルのパブリッシュ。宛先を **Event Stream Processing** に設定していることを確認してください。詳細については、"Publish a Project Champion Model" (*SAS Model Manager: User's Guide*)を参照してください。

2 SAS Model Manager でモデルのステータスが正常に Published successfully に変更されると、モデルが SAS Event Stream Processing Studio の**プロジェクト**ページに表示されます。プロジェクトを開いて、次のアイテムを表示できます。

   a ワークスペースの SAS Event Stream Processing Studio モデラーにはソースウィンドウと計算ウィンドウが含まれています。

   b 作成された SAS Micro Analytic Service モジュールを表示するには、をクリックし、右側のペインで **SAS Micro Analytics Service Models** を展開します。

   c SAS Micro Analytic Service モジュールのファイルを表示するには、左側のツールバーのをクリックし、**/home/analytics/mas-modules/*moduleName*** フォルダーを**プロジェクトパッケージ**ペインで表示します。

   d プロジェクトバージョンを表示するには、をクリックし、**バージョン 1** を**プロジェクトバージョン**ペインから選択します。右側のペインの情報は、プロジェクトがすでにテストプロモーションレベルにプロモートされていることを示しています。プロジェクトをテストにプロモートすることで、SAS Event Stream Manager で使用可能になります。

3 SAS Event Stream Manager で、ナビゲーションバーの**コンテナージョブ**をクリックします。ステータス列には、プロジェクトコンテナーを作成するジョブの進行状況が表示されます。ステータスがの場合、プロジェクトコンテナーが作成されていました。詳細については、"Deploy and Manage Project Containers" (*SAS Event Stream Processing: Using SAS Event Stream Manager*)を参照してください。

SAS Event Stream Processing Studio での SAS Micro Analytic Service モジュールの操作

概要

SAS Micro Analytic Service モジュールを使用すると、Python、DS2、および C を使用して SAS Event Stream Processing Studio で入力ハンドラー関数を作成できます。入力ハンドラーにコードを埋め込んだり、外部ファイルにコードを配置したりできます。

SAS Micro Analytic Service モジュールに含まれる Python コードは、ESP プロジェクトの他の場所の Python コードとは別に制御および実行されます。

コードを埋め込んだり、外部コードファイルを使用したりせずに、SAS Model Manager で作成されたモデルを参照できます。詳細については、“SAS Model Manager の統合の概要”を参照してください。

SAS Micro Analytic Service モジュールの作成

1 プロジェクトを開き、  をクリックします。

2 右ペインで、**SAS Micro Analytic Service モジュール**を展開します。

3  をクリックします。

SAS Micro Analytic Service モジュールウィンドウが表示されます。

4 **名前**フィールドに、モジュールの名前を入力します。

5 **言語**ドロップダウンリストで、モジュールの記述に使用する言語を選択します。

注: Python コードを使用する場合、Python コードを SAS Micro Analytic Service モジュールに配置する代わりに、Python ウィンドウを使用できます。詳細については、“Python ウィンドウの操作 SAS Event Stream Processing Studio”を参照してください。

6 **説明**フィールドに、モジュールの説明を入力します。

7 **関数名**フィールドに、カンマ区切りの関数名のリストを入力します。

8 **コードソース**フィールドで、次のいずれかのオプションを選択します。

- あなた自身のコードを入力する**埋め込みコード**
- 外部ファイルにあるコードを使用する**外部ファイル**
- 分析ストアファイルでコードを使用する **SAS Micro Analytic Service ストア**

> **注:** このオプションを使用すると、SAS Model Manager からモデルが自動的にインポートされます。このオプションは、SAS Event Stream Processing が他の SAS Viya プラットフォーム製品と共に導入されている場合にのみ関係します。詳細については、"SAS Model Manager の統合の概要"を参照してください。

9 **コードソース**フィールドで**埋め込みコード**を選択した場合は、**埋め込みコード**フィールドにコードを入力します。コード言語を C または DS2 に設定すると、コードの入力時にエディターによってキーワードが提案されます。コード言語を Python に設定すると、エディターは Python コードを検証します。

10 **コードソース**フィールドで**外部ファイル**を選択した場合は、**外部ファイル**フィールドに外部ファイルへのファイルパスを入力します。

11 **コードソース**フィールドで **SAS Micro Analytic Service ストア**を選択した場合は、次の操作を実行します。

   a **外部ファイル**フィールドに、分析ストアファイルへのファイルパスを入力します。

   b **SAS Micro Analytic Service ストア**のフィールドに、モジュールストアの場所を入力します。

   c **SAS Micro Analytic Service ストアバージョン**フィールドに、モジュールストアの場所のバージョンを入力します。

   d **Module Members** フィールドに、モジュールのメンバー名を入力します。新しいモジュールメンバーを追加するには、をクリックし、該当するフィールドに入力します。

12 **OK** をクリックします。

作成したモジュールは、SAS Micro Analytic Service モジュールテーブルに表示されます。

SAS Micro Analytic Service Module の削除

1 プロジェクトを開き、をクリックします。

2 右ペインで、**SAS Micro Analytic Service モジュール**を展開します。

3 削除したいモジュールを選択し、をクリックします。

4 削除を確認します。

プロジェクトがプロジェクトパッケージの一部であり、モデルをプロジェクトパッケージに手動でインポートした結果として SAS Micro Analytic Service モジュールが作成された場合、SAS Micro Analytic Service モジュールを削除すると、プロジェクトパッケージ内の関連ファイルも削除されます。

プロジェクトがプロジェクト XML ファイルの場合、モジュールはプロジェクトから削除されます。これは、プロジェクトの SAS Micro Analytic Service ストアへの参照が削除されることを意味します。

ただし、SAS Micro Analytic Service ストアは削除されません。管理者グループのメンバーは、SAS Event Stream Manager の**設定**ページを使用して、SAS Micro Analytic Service ストアを削除できます。詳細については、"Delete SAS Micro Analytic Service Stores" (*SAS Event Stream Processing: Using SAS Event Stream Manager*)を参照してください。

SAS Micro Analytic Service モジュールの更新

このトピックは、SAS Model Manager から自動的にインポートされたモデルに関連します。詳細については、"SAS Model Manager の統合の概要"を参照してください。

SAS Micro Analytic Service モジュールを含む SAS Event Stream Processing プロジェクトは、新しいチャンピオンモデルが宣言されたときに更新できます。SAS Event Stream Processing Studio は、新しいチャンピオンモデルを自動的にチェックしません。

新しいチャンピオンモデルをチェックするには、以下のようにします。

1 プロジェクトを開きます。

2 ⋮をクリックし、**新しいチャンピオンモデルをチェックする**を選択します。

3 **新しいチャンピオンモデルが利用可能**ウィンドウで、 **はい**をクリックして更新を承認します。

注: プロジェクトに複数の SAS Micro Analytic Service モジュールが含まれている場合には、新しいチャンピオンモデルが利用可能なモジュール毎に**新しいチャンピオンモデルが利用可能**ウィンドウが再表示されます。

入力ハンドラの操作

概要

プロシジャおよび計算ウィンドウのイベントストリーム入力ハンドラーを登録することができます。*入力ハンドラ*では、モデル内の受信イベントストリームが処理されます。SAS Model Manager で作成したスコアコードを、モデルの特定の計算ウィンドウに直接インポートすることができます。関数を入力ウィンドウにバインドするには、計算ウィンドウで SAS Micro Analytic Service マップを定義します。このバインドは、計算ウィンドウの入力ハンドラとして機能します。

プロシ計算ウィンドウでの入力ハンドラー関数の作成

1 関連するプロジェクトを開きます。

2 関連する計算ウィンドウをクリックします。

右ペインには計算ウィンドウのプロパティが表示されます。

3 右ペインで、**設定**を展開します。

4 **計算**ドロップダウンリストで、**ユーザー指定**を選択します。

5 **ハンドラー**セクションで、インポートハンドラー関数のリンク先となる入力ウィンドウの行をクリックします。

6 💉をクリックします 。

入力ハンドラーウィンドウが表示されます。

7 **ハンドラーのタイプ**フィールドで、次のいずれかのハンドラーのタイプを選択します:

- **SAS Micro Analytic Service** – SAS Micro Analytic Service モジュールを参照することができます。
- **モジュールを SAS Model Manager からインポート** – SAS Micro Analytic Service モジュールを参照することができます。。

> **注:** このオプションを使用すると、SAS Model Manager からモデルが自動的にインポートされます。詳細については、“SAS Model Manager の統合の概要”を参照してください。

入力ハンドラーウィンドウがリロードされ、選択したハンドラーのタイプに関連するフィールドが表示されます。

> **注:** **関数**ドロップダウンリストで、デフォルトで自動的に選択されている関数がドロップダウンリストの最初の機能です。または、ドロップダウンリストに関数が含まれていない場合は、スコアメソッドが表示されます。

8 必要に応じて他のフィールドを更新します。

9 **OK** をクリックします。

ハンドラー関数セクションが更新され、インポートされた関数が表示されます。

プロシジャウィンドウでの入力ハンドラ関数の作成

1 関連するプロジェクトを開きます。

2 関連するプロシジャウィンドウをクリックします。

右ペインにはプロシジャウィンドウのプロパティが表示されます。

3 右側のペインで、**入力ハンドラ**を展開します。

4 インポートハンドラ関数をリンクする入力ウィンドウの行をクリックします。

5 🖉をクリックします 。

入力ハンドラウィンドウが表示されます。

6 プラグインライブラリの名前と、参照する関数を入力します。

参照する関数は、デフォルトで受信イベントブロック内の単一のイベントで動作します。関数は、イベントごとに繰り返し呼び出されます。または、各イベントではなく、受信イベントブロック全体で関数を実行することもできます。この機能を有効にするには、 **受信イベントブロックでこの機能を 1 回実行する**チェックボックスを選択します。

注意
イベントではなくイベントブロックで関数を実行する場合は注意が必要です。 プラグインライブラリから実行している関数は SAS Event Stream Processing 環境の外部にあるため、ESP サーバーがイベントの処理を停止する可能性があります。

---

7 **OK** をクリックします。

入力ハンドラー関数セクションが更新され、インポートされた関数が表示されます。

SAS Event Stream Processing Studio での ONNX モデルの使用

SAS Event Stream Processing Studio を使用して、プロジェクトから Open Neural Network Exchange(ONNX)モデルを参照するには、次の操作を実行します:

1 ONNX モデルを取得します。

2 ONNX モデルを SAS Event Stream Processing Studio がアクセスできる場所に配置します。この場所は永続ボリューム上にある必要があります。

プロジェクトがプロジェクトパッケージの一部である場合、ONNX モデルを適切な場所に配置する簡単な方法は、プロジェクトパッケージの**分析**フォルダーにアップロードすることです。詳細については、“プロジェクトパッケージを管理する”を参照してください。

> 重要　Kubernetes 環境では、ONNX モデルを使用するには、SAS Event Stream Processing 配置するときに永続ボリュームにアクセスするように構成する必要があります。この構成では、オーバーレイを適用します。詳細については、“永続ボリュームの管理” (*SAS Event*

Stream Processing: Kubernetes 環境での SAS Event Stream Processing の使用)を参照してください。

3 プロジェクトでモデルリーダーとスコアウィンドウを使用して、ONNX モデルを参照します。

SAS Event Stream Processing Studio を使用してプロジェクトから ONNX モデルを参照する方法の例については、Computer Vision with ONNX、Pose Estimation with ONNX、および Voice Transcription with ONNX のサンプルをインストールしてください。詳細については、"サンプルプロジェクトのインストール"を参照してください。

次のドキュメントを参照してください。SAS Event Stream Processing の ONNX モデルのサポートの詳細については、サポートされている実行プロバイダーに関する情報も含め、"ONNX モデルの取り扱い" (*SAS Event Stream Processing: ストリーミング分析の適用*)を参照してください。

ソースウィンドウにイベントを転送

イベント転送を使用して、Lua ウィンドウまたは Python ウィンドウからソースウィンドウにイベントを送信できます。イベントの転送は、転送を行うウィンドウの 1 つが 1 つのイベントから複数のイベントを生成する場合に特に便利です。この場合、ダイレクトエッジの代わりにイベント転送を使用すると、使用されるメモリが少なくなり、ストリーミングモデルの同時実行性が高くなります。

このトピックでは、SAS Event Stream Processing Studio でイベント転送を設定する方法について説明します。Lua ウィンドウからイベントを転送する方法の詳細については、"Event Forwarding" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。Python ウィンドウからイベントを転送する方法の詳細については、"Event Forwarding" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

イベント転送を使用するプロジェクトの例については、転送のサンプルをインストールします。詳細については、"サンプルプロジェクトのインストール"を参照してください。次の説明では、転送の例の資料を使用して、イベント転送がプロジェクトダイアグラムでどのように表されるかを示します。

転送の例には、2 つの連続クエリ(内部クエリと外部クエリ)が含まれています。内部連続クエリの convertTemp ウィンドウは、イベントを外部連続クエリの readsFahrenheitExternal ウィンドウに転送します。Lua ウィンドウまたは Python ウィンドウが、別の継続的なクエリ内のソース ウィンドウにイベントを転送する場合、このアクションは、エッジではなく、Lua ウィンドウまたは Python ウィンドウの アイコンで示されます。同様に、ソース ウィンドウが別の継続クエリから転送されたイベントを受信すると、このアクションは アイコンで示されます。

図 33　内部を呼び出す連続クエリ

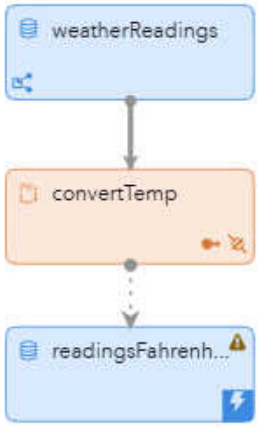
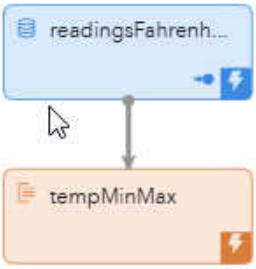


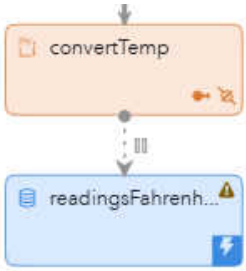
図 34　外部を呼び出す連続クエリ



⋯➤。転送の例では、別の連続クエリにイベントを転送することに加えて、convertTemp ウィンドウはイベントを同じ連続クエリ内のウィンドウに転送します。

イベント転送のための接続は、アクティブまたは非アクティブになります。プロジェクトの実行中は、イベント転送用の非アクティブな接続で、イベントをターゲットソースウィンドウに送信しません。イベント転送エッジが非アクティブな場合、アイコン⫼がエッジに表示されます。次の図は、convertTemp と次のウィンドウの間のエッジを非アクティブにした場合の転送の例を示しています。

図 35　非アクティブなイベント転送エッジの例



ソースウィンドウにイベントを送信するように Lua ウィンドウまたは Python ウィンドウを構成するには、次のようにします。

- “Lua の Windows の操作 SAS Event Stream Processing Studio”
- “Python ウィンドウの操作 SAS Event Stream Processing Studio”

複数の Lua ウィンドウと Python ウィンドウが、同じソースウィンドウにイベントを渡すことができます。特定のソースウィンドウのイベント転送の受信接続を表示するには、次のようにします。

1 ソースウィンドウを選択します。

2 右ペインで、**イベントの転送**を展開します。

SAS Event Stream Processing Studio での Lua の使用

Lua 仮想環境の操作

Lua 仮想環境では、Lua モジュールのセットが指定されます。ユーザーが管理者が作成した仮想環境を選択すると、その仮想環境で指定された Lua モジュールでプロジェクトが実行されます。

管理者が仮想環境を作成する方法については、“仮想環境の指定”を参照してください。仮想環境の選択方法については、“プロジェクトプロパティの更新”を参照してください。

Lua の Windows の操作 SAS Event Stream Processing Studio

Lua ウィンドウでは、Lua プログラムで SAS Event Stream Processing イベントを消費および生成することができます。このトピックでは、SAS Event Stream Processing Studio で Lua ウィンドウを設定する方法について説明します。Lua ウィンドウの使用の詳細については、“Using Lua Windows” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

Lua ウィンドウを使用したプロジェクトの例については、Lua Compute、Lua Parse、および Lua State のサンプルをインストールします。詳細については、“サンプルプロジェクトのインストール”を参照してください。

SAS Event Stream Processing Studio で Lua ウィンドウを構成するには、次のようにします。

1 プロジェクトを開きます。

2 左の**ウィンドウ**ペイン上で**変換**を展開し、ワークスペースに Lua ウィンドウをドラッグします。

3 右ペインで、**Lua 設定**を展開します。

4 **コピーするフィールド**フィールドを使って、入力イベントから出力イベントにコピーするフィールドを指定します。より少ないフィールドを選択すると、パフォーマンスが向上します。

a 次のステップで指定したフィールドを除くすべてのフィールドを 使用するには、**除外するフィールド**ラジオボタンを選択します。次のステップで指定したフィールドだけをコピーするには、**含めるフィールド**ラジオボタンを選択します。

b ドロップダウンリストを展開して、除外または含めるフィールドのチェックボックスを選択します。

> ヒント　選択できるフィールドの数が多い場合は、フィールド名を手動で入力することもできます。ドロップダウンリストにフィールド名の入力を開始すると、一致するフィールドが表示されます。

5 **Lua コードで使用するフィールド**フィールドには、ESP サーバーから Lua コードに渡すフィールドを指定します。**除外するフィールド**と**含めるフィールド**のラジオボタン、およびドロップダウンリストは、**コピーするフィールド**の場合と同様に機能します。

> ヒント　**パラメーターの展開**チェックボックスを選択して、これらのフィールドを Lua コードに追加できます。フィールドはイベント関数のパラメーターとして追加されます。
>
> **パラメーターの展開**チェックボックスを選択した場合、**イベントをブロック単位で処理**チェックボックスは使用できません。

6 **イベント関数**フィールドに、SAS Event Stream Processing イベントの生成に使用する Lua 関数の名前を入力します。

この関数は、これらの手順で後で入力する Lua コードに存在する必要があります。Lua コードには複数の関数を含めることができます。**イベント関数**フィールドは、どの関数がその Lua コードへのモデルのエントリポイントであるかを指定します。

7 必要に応じて、**初期化関数**フィールドを使用して、Lua コード内のどの関数が初期化関数として機能するかを指定します。詳細については、“Initialization of Lua Entities” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

8 必要に応じて、**ハートビート機能**フィールドを使用して、ハートビートごとに呼び出される Lua 関数の名前を指定します。

ハートビート間隔は、Lua ウィンドウではなく、プロジェクト全体に対して指定されます。Lua ウィンドウのハートビートの詳細については、“Configuring a Lua Window” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。プロジェクトレベルでのハートビートの詳細については、“XML Language Elements for the Structure of a Model” (*SAS Event Stream Processing: XML Language Reference for Event Stream Processing Models*)を参照してください。

9 必要に応じて、**クリーンアップ関数**フィールドを使用して、Lua コード内のどの関数がクリーンアップ関数として機能するかを指定します。クリーンアップ機能は、プロジェクトの停止時に実行されます。クリーンアップ関数は、必要な Lua クリーンアップタスクを実行する必要があります。

10 必要に応じて、**イベントをブロックで処理する**チェックボックスを選択します。イベントをブロックで処理する場合、イベント関数はブロック内のイベントを表す Lua テーブルの配列を受け取ります。それ以外の場合、イベント関数は単一のイベントを受け取ります。

11 必要に応じて、**プロパティの永続化**チェックボックスを選択します。詳細については、"Persistent Properties" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

12 必要に応じて、**バイナリデータのエンコード**チェックボックスを選択します。このオプションは、Lua コードでデータが使用される前に、Base64 エンコードを使用してバイナリデータをテキストとしてエンコードします。

13 必要に応じて、 **必須モジュール**フィールドを使用して、Lua コード内で参照する Lua モジュールを選択します。詳細については、"SAS Event Stream Processing Studio での Lua モジュールの使用"を参照してください。

14 コードソースを選択します:

- このウィンドウに新しい Lua コードを入力する場合は、**ウィンドウ**を選択します。
- 既存の Lua スニペット内の共有コードを参照する場合は、**スニペット**を選択します。Lua ウィンドウがスニペット内で共有コードを使用する場合、そのスニペットを指す他の Lua エンティティとすべてのデータ 、関数、および変数を共有します。共有コードは、現在それを使用しているエンティティによって 常にロックされます。詳細については、"SAS Event Stream Processing Studio での Lua スニペットの使用"を参照してください。
- ファイル内の Lua コードを参照する場合は、**ファイル**を選択します。**コードファイルへのパス**フィールドを使用して、ファイルを指定します。プロジェクトがプロジェクトパッケージの一部である場合は、参照するファイルもプロジェクトパッケージ内にある必要があります。詳細については、"プロジェクトパッケージに含まれるフォルダーの参照ファイル"を参照してください。

15 このウィンドウに新しい Lua コードを入力する場合:

a  をクリックします。

式エディタウィンドウが表示されます。

b コードを入力します。Lua ウィンドウでの適切な Lua コードの詳細については、"Using Lua Windows" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。コードエディタの詳細については、"コードと式を編集する"を参照してください。

c **OK** をクリックします。

16 必要に応じて、**致命的なエラーが発生した場合のエラー処理の方法**フィールドを使用して、エラー処理の方法を選択します。

Lua コードは例外が発生する可能性があります。init 関数からの例外により、プロジェクトの読み込みに失敗します。デフォルトでは、create 関数、heartbeat 関数、およびスニペット(スレッド化された場合)からの例外は、プロジェクトレベルのエラー処理設定に基づいて処理されます。必要に応じて、Lua ウィンドウ用に別のエラー処理の戦略を選択して、プロジェクトレベルの設定を無効にできます。エラーをログに記録してプロジェクトの実行を続行するか、致命的なエラーをログに記録してプロジェクトを停止するかを選択できます。プロジェクトレベルの設定を確認するには、  をクリックし、右側のペインで**詳細**を展開して、選択されているエラー処理戦略を確認します。

17 必要に応じて、イベントを Lua ウィンドウからソースウィンドウに転送できます。詳細については、"ソースウィンドウにイベントを転送"を参照してください。

a **イベント転送**を展開します。

b 必要に応じて、**イベント作成の抑制**チェックボックスを選択します。このオプションを選択すると、Lua ウィンドウはイベントを転送するだけであり、イベントを生成しません。

c をクリックします。

d **イベント転送先**ウィンドウで、連続クエリとソースウィンドウを選択します。

e 必要に応じて、ブロックサイズ(つまり、ソースウィンドウにパブリッシュされる各イベントブロックに含めるイベントの数)を調整します。

f イベント転送のための接続は、アクティブまたは非アクティブになります。必要に応じて、**アクティブ**チェックボックスを使用して、接続のステータスを変更します。

g **OK** をクリックします。

18 必要に応じて、右ペインでその他のウィンドウのプロパティを編集します。

ヒント　プロジェクトがプロジェクトパッケージの一部であり、Lua ウィンドウを新しいウィンドウの種類として他のユーザーが利用できるようにしたい場合は、ウィンドウをカスタムウィンドウに変換できます。詳細については、“Lua ウィンドウまたは Python ウィンドウをカスタムウィンドウに変換する” (153 ページ)を参照してください。

SAS Event Stream Processing Studio での Lua スニペットの使用

Lua スニペットは、他の Lua エンティティをサポートするために使用できる Lua コードの自己完結型ブロックです。Lua スニペットはプロジェクトレベルで作成します。その後、Lua スニペットに含まれるプロパティを、プロジェクト内の他の場所の Lua コードから参照できます。

このトピックでは、SAS Event Stream Processing Studio で Lua スニペットを作成する方法について説明します。一般的な情報については、“Using Lua Snippets” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

次の手順では、Lua スニペットを追加するプロジェクトが既にあることが前提となっています。この手順では、Lua スニペットを追加する方法について説明します。Lua スニペットを含む完全なプロジェクトの例については、Lua スニペットのサンプルをインストールします。詳細については、“サンプルプロジェクトのインストール”を参照してください。

1 プロジェクトを開きます。

2 をクリックします。

3 右側のペインで、**スニペット**を展開します。

4 **Lua スニペット**セクションの **Lua ルートパス**フィールドには、プロジェクトの Lua ルート ディレクトリが表示されます。この情報は、永続プロパティを使用する場合に関係します。詳細については、“Persistent Properties” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

プロジェクトがプロジェクトパッケージの一部である場合、パスは**@ESP_PROJECT_OUTPUT@/luaroot** に設定され、このパスを変更することはできません。パスに含まれる ESP_PROJECT_OUTPUT 環境変数の詳細については、“プロジェクトパッケージを管理する”を参照してください。

プロジェクトがプロジェクトパッケージの一部ではない場合は、プロジェクトの Lua ルートディレクトリを入力します。書き込み可能なディレクトリを指定してください。Kubernetes 環境では、永続ボリューム内に場所を指定する必要があります。例えば、このフィールドを、**/mnt/data** の適切なサブディレクトリに設定することもできます。

5 をクリックします。

6 **Lua コードスニペットの編集**ウィンドウで、スニペットの名前を調整します。

7 Lua スニペットの説明を入力します。**Lua コードスニペットの編集**ウィンドウを閉じると、入力した説明が右ペインの **Lua スニペット**セクションに表示されています。この説明は、プロジェクト内の他の場所で Lua コードを編集し、コードエディターを使用してスニペットへの参照を追加する場合にも使用できます。説明を入力すると、適切なスニペットを選択するのに役立ちます。詳細については、“コードと式を編集する”を参照してください。

8 コードソースを選択します。

- このスニペットに新しい Lua コードを入力する場合は、**コードエディター**を選択します。
- ファイル内の Lua コードを参照する場合は、**ファイル**を選択します。**コードファイルへのパス**フィールドを使用して、ファイルを指定します。プロジェクトがプロジェクトパッケージの一部である場合は、参照するファイルもプロジェクトパッケージ内にある必要があります。詳細については、“プロジェクトパッケージに含まれるフォルダーの参照ファイル”を参照してください。

9 必要に応じて、**初期化関数**フィールドを使用して、Lua コード内のどの関数が初期化関数として機能するかを指定します。詳細については、“Initialization of Lua Entities” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

10 必要に応じて、選択した間隔で反復される Lua 関数を指定します。

- **繰り返し関数**フィールドは、Lua コード内のどの関数が繰り返し関数として機能するかを指定します。
- この関数からの戻りコードは、その関数の反復を維持するかどうかをサーバーに指示するブール値です。true を返せば、その関数が反復されます。false を返せば、その関数が反復されません。
- 反復関数の使用(スレッドの開始とも呼ばれる)に関する詳細は、“Using Lua Snippets” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

11 必要に応じて、**クリーンアップ関数**フィールドを使用して、Lua コード内のどの関数がクリーンアップ関数として機能するかを指定します。クリーンアップ機能は、プロジェクトの停止時に実行されます。クリーンアップ関数は、必要な Lua クリーンアップタスクを実行する必要があります。

12 必要に応じて、**必須モジュール**フィールドを使用して、スニペットの Lua コード内で参照する Lua モジュールを選択します。詳細については、“SAS Event Stream Processing Studio での Lua モジュールの使用”を参照してください。

13 **コードソース**フィールドを**コードエディター**に設定した場合は、コードエディターに Lua コードを入力します。詳細については、“コードと式を編集する”を参照してください。

14 **OK** をクリックします。

15 必要に応じて、スニペットを追加します。

16 スニペットが **Lua スニペット**セクションの表内に表示される順序を確認します。Lua スニペットは、テーブルに記載されている順序でロードされます。他のスニペットよりも前に宣言されるスニペットは、後続スニペット内に変数を必要とする場合がありますが、それらは利用できません。

17 プロジェクトの他の部分にある Lua コードのスニペット内のプロパティを参照します。このような参照を追加するには、コードエディターの**スニペットプロパティ**タブを使用します。詳細については、“コードと式を編集する”を参照してください。Lua ウィンドウでは、**コードソース**フィールドを使用してスニペットを参照することもできます。詳細については、“Lua の Windows の操作 SAS Event Stream Processing Studio”を参照してください。

18 その他、プロジェクトの残りの部分に必要な他の設定を行います。

SAS Event Stream Processing Studio での Lua モジュールの使用

Lua モジュールは、他の Lua エンティティで再利用できる自己完結型の Lua コードブロックです。Lua モジュールはプロジェクトレベルで作成します。Lua モジュールを使用すると、コードを一度記述するだけで、同じプロジェクト内の他の場所にある Lua コードからそのコードを参照できます。たとえば、便利な共通関数を定義する Lua モジュールを作成し、プロジェクト内の他の Lua コードからそれらの関数を参照できます。

このトピックでは、SAS Event Stream Processing Studio で Lua モジュールを作成する方法について説明します。詳細については、“Using Lua Modules” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

次の手順は、Lua モジュールを追加するプロジェクトがすでにあることを前提としています。この手順では、Lua モジュールを追加する方法を説明します。Lua モジュールを含む完全なプロジェクトの例については、Lua モジュールのサンプルをインストールします。詳細については、“サンプルプロジェクトのインストール”を参照してください。

1 プロジェクトを開きます。

2 をクリックします。

3 右側のペインで、**モジュール**を展開します。

4 **Lua モジュール**セクションで、をクリックします。

5 **Lua コードモジュールの編集**ウィンドウで、モジュールの名前を調整します。

6 Lua モジュールの説明を入力します。

7 **必須モジュール**ドロップダウンフィールドで、この新しいモジュールから参照する他の Lua モジュールを指定します。

8 Lua コードを入力するには、次のようにします: コードエディタの詳細については、“コードと式を編集する”を参照してください。

9 **OK** をクリックします。

10 必要に応じて、モジュールを追加します。

11 プロジェクトの他の部分にある Lua コードからモジュールを参照します。コードエディターの右ペインを使用して、そのような参照を追加します。詳細については、“コードと式を編集する”を参照してください。

12 プロジェクトの残りの部分について、その他の必要な設定を構成します。

SAS Event Stream Processing Studio での Lua ベースフィルターウィンドウの操作

フィルターウィンドウを使用すると、どの入力イベントのパススルーを許可するかを決定できます。フィルター条件を定義するには、Lua、Python、プラグイン関数、または Expression Engine Language (EEL)を使用できます。

注: フィルターウィンドウで EEL を使用する機能は、従来の機能です。

このトピックでは、SAS Event Stream Processing Studio で Lua ベースフィルターウィンドウを作成する方法について説明します。一般的な情報については、“Using Filter Windows” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

次の手順では、出力スキーマを含むソースウィンドウが既に含まれているプロジェクトがあることを前提としています。この手順では、フィルターウィンドウを追加する方法について説明します。Lua ベースのフィルターウィンドウを使用した完全なプロジェクトの例については、Trades および Sailing のサンプルをインストールしてください。詳細については、“サンプルプロジェクトのインストール”を参照してください。

1 プロジェクトを開きます。

2 左側の**ウィンドウ**ペインで**変換**を展開し、ワークスペースにフィルターウィンドウをドラッグします。

3 右ペインで、**フィルター**を展開します。

4 フィルターの種類を Lua に設定します。

5 **Lua コードで使用するフィールド**フィールドを使用して、ESP サーバーから Lua コードに渡すフィールドを指定します。より少ないフィールドを選択すると、パフォーマンスが向上します。

   a 次のステップで指定したフィールドを除くすべてのフィールドを 使用するには、**除外するフィールド**ラジオボタンを選択します。次のステップで指定したフィールドだけを使用するには、**含めるフィールド**ラジオボタンを選択します。

   b ドロップダウンリストを展開して、除外または含めるフィールドのチェックボックスを選択します。

> ヒント　**パラメーターの展開**チェックボックスを選択して、選択したフィールドを Lua コードに追加できます。フィールドはフィルター関数のパラメーターとして追加されます。

6 **フィルター関数**フィールドを使用して、Lua コード内のどの関数がフィルター関数として機能するかを指定します。デフォルト値は filter です。

7 必要に応じて、**初期化関数**フィールドを使用して、Lua コード内のどの関数が初期化関数として機能するかを指定します。詳細については、"Initialization of Lua Entities" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

8 必要に応じて、**ハートビート機能**フィールドを使用して、ハートビートごとに呼び出される Lua 関数の名前を指定します。

ハートビート間隔は、フィルターウィンドウではなく、プロジェクト全体に対して指定されます。プロジェクトレベルでのハートビートの詳細については、"XML Language Elements for the Structure of a Model" (*SAS Event Stream Processing: XML Language Reference for Event Stream Processing Models*)を参照してください。

9 必要に応じて、**クリーンアップ関数**フィールドを使用して、Lua コード内のどの関数がクリーンアップ関数として機能するかを指定します。クリーンアップ機能は、プロジェクトの停止時に実行されます。クリーンアップ関数は、必要な Lua クリーンアップタスクを実行する必要があります。

10 必要に応じて、 **必須モジュール**フィールドを使用して、フィルター条件の Lua コード内で参照する Lua モジュールを選択します。詳細については、"SAS Event Stream Processing Studio での Lua モジュールの使用"を参照してください。

11 コードソースの選択:

- フィルター条件に新しい Lua コードを入力する場合は、**ウィンドウ**を選択します。
- 既存の Lua スニペットで共有コードを参照する場合は、**スニペット**を選択します。Lua ベースのフィルターウィンドウがスニペット内で共有コードを使用すると、すべてのデータ、関数、および変数が、そのスニペットを指す他の Lua エンティティと 共有されます。共有されたコードは、現在それを使用しているエンティティによって 常にロックされます。詳細については、"SAS Event Stream Processing Studio での Lua スニペットの使用"を参照してください。

12 フィルター条件に新しい Lua コードを入力する場合:

a  をクリックします。

b **Lua コードの編集**ウィンドウで、ご使用の Lua コードを入力します。コードには、ブール値を返すフィルター関数を含める必要があります。ESP サーバーは、入力イベント毎にこの関数を呼び出して、戻りコードを使用して、出力イベントを生成するかかどうかを決定します。この関数の名前が、**関数のフィルター**フィールドで指定されている名前と一致することを確認してください。

フィルターウィンドウでの適切な Lua コードの詳細については、"Using Lua in a Filter Window" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。コードエディタの詳細については、"コードと式を編集する"を参照してください。

c **OK** をクリックします。

13 フィルターウィンドウとプロジェクトの残りの部分に必要なその他の設定を構成します。

SAS Event Stream Processing Studio での Lua ベースパターンウィンドウの操作

パターンウィンドウを使用すると、対象イベント(EOI)がプロジェクトを通過する際にそれらのイベントを検出し、パターンロジックを適用した結果として出力イベントを生成できます。EOI を定義するには、Lua、Python、または Expression Engine Language(EEL)を使用できます。

注: パターンウィンドウでは Lua または Python を使用することをお勧めします。パターンウィンドウで EEL を使用する機能は、従来の機能です。

このトピックでは、SAS Event Stream Processing Studio で Lua ベースパターンウィンドウを作成する方法について説明します。パターンウィンドウの目的に関する一般的な情報については、"Using Pattern Windows" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

次の手順では、出力スキーマを含むソースウィンドウが既に含まれているプロジェクトがあることを前提としています。この手順では、パターンウィンドウを追加する方法について説明します。Lua ベースのパターンウィンドウを含む完全なプロジェクトの例については、Lua パターンのサンプルをインストールしてください。詳細については、"サンプルプロジェクトのインストール"を参照してください。

1 プロジェクトを開きます。

2 左の**ウィンドウ**ペインの**ユーティリティ**を展開し、ワークスペースにパターンウィンドウをドラッグします。

3 右側のウィンドウで、**パターン**を展開します。

デフォルトでは、パターンタイプは Lua に設定されています。

4 をクリックします。

新しいパターンウィンドウが表示されます。**新しいパターン**ウィンドウは、4 ページで構成されるウィザードです。

5 **初期化**ページを完了します:

a 必要に応じてパターンの名前を調整します。

b 必要に応じて、インデックスフィールドを選択します。入力ウィンドウからのこれらのフィールドは、インデックス生成関数を形成します。入力ウィンドウは、ソースウィンドウまたは別の種類のウィンドウである可能性があります。すべての着信イベントは、指定されたインデックスによってグループ化されます。

c 必要に応じて、**イベントのインデックス値に関係なく、オープンパターンが着信イベントごとにタイムアウトになる可能性を許可する**チェックボックスをオンにします。

d 必要に応じて、1 つ以上の時間フィールドを指定します。ウィンドウごとに、時間を指定するために使用するフィールドを選択できます。時間フィールドが指定されていない場合は、システム時間が使用されます。

e **次へ**をクリックします。

6 **Lua コード**ページを完了させます。

a 必要に応じて、**必須モジュール**フィールドを使用して、パターンの Lua コード内で参照する Lua モジュールを選択します。詳細については、"SAS Event Stream Processing Studio での Lua モジュールの使用"を参照してください。

b パターンに必要な EOI 関数と出力関数を含む、パターンの Lua コードを入力します。

コードエディタの使用の詳細については、"コードと式を編集する"を参照してください。EOI 関数の詳細については、"Writing EOI Functions in Lua" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。出力関数の詳細については、"Writing Output Functions in Lua" (*SAS Event Stream Processing: Using Source and Derived Windows*) を参照してください。

c **次へ**をクリックします。

7 **出力とイベント**ページに記入します。

a 必要に応じて、**初期化関数**フィールドを使用して、Lua コード内のどの関数が初期化関数として機能するかを指定します。詳細については、"Initialization of Lua Entities" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

b **出力関数**で、出力関数の名前を選択します。出力関数は、**Lua コード**ページで追加した Lua コードに存在する必要があります。

c 必要に応じて、**クリーンアップ関数**フィールドを使用して、Lua コード内のどの関数がクリーンアップ関数として機能するかを指定します。クリーンアップ機能は、プロジェクトの停止時に実行されます。クリーンアップ関数は、必要な Lua クリーンアップタスクを実行する必要があります。

d **イベント**サブセクションで、をクリックして EOI を作成します。

新しい行がテーブルに追加されます。

e 名前列で、必要に応じてイベントの名前を調整します。

f フィールドを含む列で、イベントに渡す Lua フィールドを選択します。この列を空のままにすると、すべての Lua フィールドがイベントに渡されます。

g Lua 関数列で、イベントに関連付ける EOI 関数を選択します。**Lua 関数の編集**ウィンドウで入力したすべての関数は、この列で使用できます。出力関数ではなく、適切な EOI 関数を選択します。

h イベントの送信列で、以前に一致したすべてのイベントを関数に送信する場合は、チェックボックスをオンにします。

i 必要に応じて EOI を追加します。

j **次へ**をクリックします。

8 **論理式**ページを使用してパターンロジックを作成します。論理式は Lua 言語を使用していません。終了したら、**OK** をクリックします。

コードエディタの使用の詳細については、"コードと式を編集する"を参照してください。論理式の記述の詳細については、"Applying Pattern Logic" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

9 必要に応じて、パターンを追加します。

10  をクリックします。

11 をクリックします。

出力スキーマウィンドウが表示されます。

12 **出力スキーマ**ウィンドウを使用して、キーフィールドを指定し、出力フィールドとフィールドの種類を指定します。終了したら、**OK** をクリックします。

注: パターンウィンドウ内のすべてのパターンには、同じ出力フィールドがあります。特定のパターンに対して異なる出力フィールドが必要な場合は、プロジェクトにパターンウィンドウを追加できます。

13 パターンウィンドウとプロジェクトの残りの部分に必要なその他の設定を構成します。

SAS Event Stream Processing Studio での Python の使用

Python 仮想環境の操作

Python 仮想環境では、Python パッケージのセットが指定されます。ユーザーが管理者が作成した仮想環境を選択すると、その仮想環境で指定された Python パッケージでプロジェクトが実行されます。

管理者が仮想環境を作成する方法については、“仮想環境の指定”を参照してください。仮想環境の選択方法については、“プロジェクトプロパティの更新”を参照してください。

Python ウィンドウの操作 SAS Event Stream Processing Studio

Python ウィンドウでは、Python プログラムで SAS Event Stream Processing イベントを消費および生成することができます。Python コードを使用する場合、Python コードを SAS Micro Analytic Service モジュールに配置する代わりに、Python ウィンドウを使用できます。詳細については、“Using Python Windows” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

注: Python ウィンドウを使用する場合、コードは Python 3.11 と互換性がある必要があります。

Python ウィンドウを含むプロジェクトの例については、Python Compute サンプルをインストールします。詳細については、“サンプルプロジェクトのインストール”を参照してください。

SAS Event Stream Processing Studio で Python ウィンドウを構成するには、次のようにします。

1 プロジェクトを開きます。

2 左の**ウィンドウ**ペインの**変換**を展開し、ワークスペースに Python ウィンドウをドラッグします。

3 右ペインで、**Python 設定**を展開します。

4 **コピーするフィールド**フィールドを使用して、入力イベントから出力イベントにコピーするフィールドを指定します。選択するフィールドの数を減らすと、パフォーマンスが向上します。

   a 次のステップで指定したフィールドを除くすべてのフィールドをコピーするには、**除外するフィールド**ラジオボタンを選択します。次のステップで指定したフィールドだけをコピーするには、**含めるフィールド**ラジオボタンを選択します。

   b ドロップダウンリストを展開して、除外または含めるフィールドのチェックボックスを選択します。

   > ヒント　選択できるフィールドの数が多い場合は、フィールド名を手動で入力することもできます。ドロップダウンリストにフィールド名の入力を開始すると、一致するフィールドが表示されます。

5 **Python コードで使用するフィールド**フィールドには、ESP サーバーから Python コードに渡すフィールドを指定します。**除外するフィールド**と**含めるフィールド**のラジオボタン、およびドロップダウンリストは、**コピーするフィールド**の場合と同様に機能します。

   > ヒント　**パラメーターを展開**チェックボックスを選択して、これらのフィールドを Python コードに追加できます。フィールドは、イベント関数のパラメーターとして追加されます。
   >
   > **パラメーターを展開する**チェックボックスをオンにすると、**イベントをブロックで処理する**チェックボックスは使用できなくなります。

6 **関数**フィールドに、SAS Event Stream Processing イベントの生成に使用する Python 関数の名前を入力します。

   この関数は、後の手順で入力する Python コードに存在する必要があります。Python コードには複数の関数を含めることができます。**イベント関数**フィールドでは、どの関数が Python コードへのモデルのエントリポイントであるかを指定します。

7 必要に応じて、**初期化関数**フィールドを使用して、Python コード内のどの関数が初期化関数として機能するかを指定します。詳細については、“Initialization of Python Entities” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

8 必要に応じて、**ハートビート機能**フィールドを使用して、ハートビートごとに呼び出される Python 関数の名前を指定します。

   ハートビート間隔は、Python ウィンドウではなく、プロジェクト全体に対して指定されます。Python ウィンドウのハートビートの詳細については、“Configuring a Python Window” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。プロジェクトレベルでのハートビートの詳細については、“XML Language Elements for the Structure of a

Model" (*SAS Event Stream Processing: XML Language Reference for Event Stream Processing Models*)を参照してください。

9 必要に応じて、**クリーンアップ関数**フィールドを使用して、Python コード内のどの関数が初期化関数として機能するかを指定します。クリーンアップ機能は、プロジェクトの停止時に実行されます。クリーンアップ関数は、必要な Python クリーンアップタスクを実行する必要があります。

10 必要に応じて、**イベントをブロックで処理する**チェックボックスを選択します。ブロック内のイベントを処理する場合、イベント関数はブロック内のイベントを表す Python テーブルの配列を受け取ります。それ以外の場合、イベント関数は単一のイベントを受け取ります。

11 必要に応じて、**エンコードバイナリデータ**チェックボックスを選択します。このオプションは、Python コードでデータが使用される前に、Base64 エンコードを使用してバイナリデータをテキストとしてエンコードします。

12 コードソースを選択します:

- このウィンドウに新しい Python コードを入力する場合は、**ウィンドウ**を選択します。
- 既存の Python スニペット内の共有コードを参照する場合は、**スニペット**を選択します。Python ウィンドウがスニペット内で共有コードを使用する場合、そのスニペットを指す他の Python エンティティとすべてのデータ 、関数、および変数を共有します。共有コードは、現在それを使用しているエンティティによって 常にロックされます。詳細については、"SAS Event Stream Processing Studio での Python スニペットの操作"を参照してください。
- ファイル内の Python コードを参照する場合は、**ファイル**を選択します。**コードファイルへのパス**フィールドを使用して、ファイルを指定します。プロジェクトがプロジェクトパッケージの一部である場合は、参照するファイルもプロジェクトパッケージ内にある必要があります。詳細については、"プロジェクトパッケージに含まれるフォルダーの参照ファイル"を参照してください。

13 このウィンドウに新しい Python コードを入力する場合:

a  をクリックします。

式エディタウィンドウが表示されます。

b コードを入力します。Python ウィンドウに適した Python コードの詳細については、"Using Python Windows" (*SAS Event Stream Processing: Using Source and Derived Windows*)の Python ウィンドウを参照してください。コードエディタの詳細については、"コードと式を編集する"を参照してください。

c **OK** をクリックします。

14 必要に応じて、**致命的なエラーが発生した場合のエラー処理の方法**フィールドを使用して、エラー処理の方法を選択します。

Python コードは例外が発生する可能性があります。init 関数からの例外により、プロジェクトの読み込みに失敗します。デフォルトでは、create 関数、heartbeat 関数、およびスニペット(スレッド化された場合)からの例外は、プロジェクトレベルのエラー処理設定に基づいて処理されます。必要に応じて、Python ウィンドウ用に別のエラー処理の戦略を選択して、プロジェクトレベルの設定を無効にできます。エラーをログに記録してプロジェクトの実行を続行するか、致命的なエラーをログに記録してプロジェクトを停止するかを選択できます。プロジェクトレベルの設定を確認するには、  をクリックし、右側のペインで**詳細**を展開して、選択されているエラー処理戦略を確認します。

15 必要に応じて、Python ウィンドウからソースウィンドウにイベントを転送できます。詳細については、“ソースウィンドウにイベントを転送”を参照してください。

   a **イベント転送**を展開します。

   b 必要に応じて、**イベント作成の抑制**チェックボックスを選択します。このオプションを選択すると、Python ウィンドウはイベントを転送するだけであり、イベントを生成しません。

   c をクリックします。

   d **イベント転送先**ウィンドウで、連続クエリとソースウィンドウを選択します。

   e 必要に応じて、ブロックサイズ(つまり、ソースウィンドウにパブリッシュされる各イベントブロックに含めるイベントの数)を調整します。

   f イベント転送のための接続は、アクティブまたは非アクティブになります。必要に応じて、**アクティブ**チェックボックスを使用して、接続のステータスを変更します。

   g **OK** をクリックします。

16 必要に応じて、右ペインでその他のウィンドウのプロパティを編集します。

> ヒント　プロジェクトがプロジェクトパッケージの一部であり、Python ウィンドウを新しいウィンドウの種類として他のユーザーが利用できるようにしたい場合は、ウィンドウをカスタムウィンドウに変換できます。詳細については、“Lua ウィンドウまたは Python ウィンドウをカスタムウィンドウに変換する” (153 ページ)を参照してください。

SAS Event Stream Processing Studio での Python スニペットの操作

Python スニペットは、他の Python エンティティをサポートするために使用できる Python コードの自己完結型ブロックです。Python スニペットはプロジェクトレベルで作成します。その後、Python スニペットに含まれるプロパティを、プロジェクト内の他の場所にある Python コードから参照できます。

このトピックでは、SAS Event Stream Processing Studio で Python スニペットを設定する方法について説明します。一般については、“Using Python Snippets” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

次の手順は、Python スニペットを追加するプロジェクトがすでにあることを前提としています。この手順では、Python スニペットを追加する方法について説明します。Python スニペットを含む完全なプロジェクトの例については、Python スニペットのサンプルをインストールしてください。詳細については、“サンプルプロジェクトのインストール”を参照してください。

1 プロジェクトを開きます。

2 をクリックします。

3 右側のペインで、**スニペット**を展開します。

4 **Python スニペット**セクションで、をクリックします。

5 **Python コードスニペットの編集**ウィンドウで、スニペットの名前を調整します。

6 Python スニペットの説明を入力します。**Python コードスニペットの編集**ウィンドウを閉じると、入力した説明が右側のペインの **Python スニペット**セクションに表示されます。この説明は、プロジェクト内の他の場所で Python コードを編集し、コードエディターを使用してスニペットへの参照を追加する場合にも使用できます。説明を入力すると、適切なスニペットを選択するのに役立ちます。詳細については、"コードと式を編集する"を参照してください。

7 コードソースを選択します。

- このスニペットに新しい Python コードを入力する場合は、**コードエディター**を選択します。
- ファイル内の Python コードを参照する場合は、**ファイル**を選択します。**コードファイルへのパス**フィールドを使用して、ファイルを指定します。プロジェクトがプロジェクトパッケージの一部である場合は、参照するファイルもプロジェクトパッケージ内にある必要があります。詳細については、"プロジェクトパッケージに含まれるフォルダーの参照ファイル"を参照してください。

8 必要に応じて、**初期化関数**フィールドを使用して、Python コード内のどの関数が初期化関数として機能するかを指定します。詳細については、"Initialization of Python Entities" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

9 必要に応じて、選択した間隔で繰り返される Python 関数を指定します。

- **繰り返し関数**フィールドでは、Python コード内のどの関数が繰り返し関数として機能するかを指定します。
- 関数からの戻りコードは、スレッドを実行し続けるかどうかをサーバーに伝えるブールです。true を返すと、関数が繰り返されます。false を返すと、関数は繰り返されません。
- 繰り返し関数の使用(スレッドの開始とも呼ばれます)の詳細については、"Using Python Snippets" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

10 必要に応じて、**初期化関数**フィールドを使用して、Python コード内のどの関数が初期化関数として機能するかを指定します。クリーンアップ機能は、プロジェクトの停止時に実行されます。クリーンアップ関数は、必要な Python クリーンアップタスクを実行する必要があります。

11 **コードソース**フィールドを**コードエディター**に設定した場合は、コードエディターに Python コードを入力します。詳細については、"コードと式を編集する"を参照してください。

12 **OK** をクリックします。

13 必要に応じて、スニペットを追加します。

14 **Python スニペット**セクションの表にスニペットが表示される順序を確認してください。Python スニペットは、表にリストされている順序でロードされます。別のスニペットの前に宣言されたスニペットでは、後続のスニペットで変数が必要になる場合がありますが、それらは使用できません。

15 プロジェクトの他の部分の Python コードからスニペット内のプロパティを参照します。このような参照を追加するには、コードエディターの**スニペットプロパティ**タブを使用します。詳細については、"コードと式を編集する"を参照してください。Python ウィンドウでは、**コードソース**フィールドを使用してスニペットを参照することもできます。詳細については、"Python ウィンドウの操作 SAS Event Stream Processing Studio"を参照してください。

16 プロジェクトの残りの部分について、その他の必要な設定を構成します。

SAS Event Stream Processing Studio での Python モジュールの操作

Python モジュールは、他の Python エンティティで使用できる Python コードの自己完結型ブロックです。Python モジュールはプロジェクトレベルで作成します。Python モジュールを使用すると、コードを一度作成すれば、同じプロジェクト内の他の場所にある Python コードからそのコードを参照できます。たとえば、便利な共通関数を定義する Python モジュールを作成し、プロジェクト内の他の Python コードからそれらの関数を参照できます。

このトピックでは、SAS Event Stream Processing Studio で Python モジュールを設定する方法について説明します。詳細については、“Using Python Modules” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

次の手順は、Python モジュールを追加するプロジェクトがすでにあることを前提としています。この手順では、Python モジュールを追加する方法について説明します。Python モジュールを含む完全なプロジェクトの例については、Python モジュールのサンプルをインストールしてください。詳細については、“サンプルプロジェクトのインストール”を参照してください。

1 プロジェクトを開きます。

2 をクリックします。

3 右側のペインで、**モジュール**を展開します。

4 **Python モジュール**セクションで、をクリックします。

5 **Python コードモジュールの編集**ウィンドウで、モジュールの名前を調整します。

6 Python モジュールの説明を入力します。

7 Python コードを入力します。エディタコードの詳細については、“コードと式を編集する”を参照してください。

8 **OK** をクリックします。

9 必要に応じて、モジュールを追加します。

10 プロジェクトの他の部分の Python コードからモジュールを参照します。このような参照を追加するには、コードエディターの右側のペインを使用します。詳細については、“コードと式を編集する”を参照してください。

11 プロジェクトの残りの部分について、その他の必要な設定を構成します。

SAS Event Stream Processing Studio で Python ベースフィルターウィンドウの操作

フィルターウィンドウを使用すると、どの入力イベントのパススルーを許可するかを決定できます。フィルター条件を定義するには、Lua、Python、プラグイン関数、または Expression Engine Language (EEL)を使用できます。

注: フィルターウィンドウで EEL を使用する機能は、従来の機能です。

このトピックでは、SAS Event Stream Processing Studio で Python ベースフィルターウィンドウを作成する方法について説明します。詳細については、*"Using Python Windows" (SAS Event Stream Processing: Using Source and Derived Windows)*を参照してください。

注: Python ベースのフィルターウィンドウを使用する場合、コードは Python 3.11 と互換性がある必要があります。

次の手順では、出力スキーマを含むソースウィンドウが既に含まれているプロジェクトがあることを前提としています。この手順では、フィルターウィンドウを追加する方法について説明します。Python ベースのフィルターウィンドウによる完全なプロジェクトの例については、Geofence サンプルをインストールします。詳細については、"サンプルプロジェクトのインストール"を参照してください。

1 プロジェクトを開きます。

2 左側の**ウィンドウ**ペインで**変換**を展開し、ワークスペースにフィルターウィンドウをドラッグします。

3 右ペインで、**フィルター**を展開します。

デフォルトでは、フィルターの種類は Python に設定されています。

4 **Lua コードで使用するフィールド**フィールドを使用して、ESP サーバーから Python コードに渡すフィールドを指定します。より少ないフィールドを選択すると、パフォーマンスが向上します。

a 次のステップで指定したフィールドを除くすべてのフィールドを 使用するには、**除外するフィールド**ラジオボタンを選択します。次のステップで指定したフィールドだけを使用するには、**含めるフィールド**ラジオボタンを選択します。

b ドロップダウンリストを展開して、除外または含めるフィールドのチェックボックスを選択します。

> ヒント　選択できるフィールドの数が多い場合は、フィールド名を手動で入力することもできます。ドロップダウンリストにフィールド名の入力を開始すると、一致するフィールドが表示されます。

5 **フィルター関数**フィールドを使用して、Python コード内のどの関数がフィルター関数として機能するかを指定します。デフォルト値は filter です。

6 必要に応じて、**初期化関数**フィールドを使用して、Python コード内のどの関数が初期化関数として機能するかを指定します。詳細については、"Initialization of Python Entities" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

7 必要に応じて、**ハートビート機能**フィールドを使用して、ハートビートごとに呼び出される Python 関数の名前を指定します。

ハートビート間隔は、フィルターウィンドウではなく、プロジェクト全体に対して指定されます。プロジェクトレベルでのハートビートの詳細については、"XML Language Elements for the Structure of a Model" (*SAS Event Stream Processing: XML Language Reference for Event Stream Processing Models*)を参照してください。

8 必要に応じて、**クリーンアップ関数**フィールドを使用して、Python コード内のどの関数が初期化関数として機能するかを指定します。クリーンアップ機能は、プロジェクトの停止時に実行されます。クリーンアップ関数は、必要な Python クリーンアップタスクを実行する必要があります。

9 コードソースの選択:

- フィルター条件に新しい Python コードを入力する場合は、**ウィンドウ**を選択します。
- 既存の Python スニペットで共有コードを参照する場合は、**スニペット**を選択します。Python ベースのフィルターウィンドウがスニペット内で共有コードを使用すると、すべてのデータ、関数、および変数が、そのスニペットを指す他の Python エンティティと 共有されます。共有されたコードは、現在それを使用しているエンティティによって 常にロックされます。詳細については、"SAS Event Stream Processing Studio での Python スニペットの操作"を参照してください。

10 フィルター条件に新しい Python コードを入力する場合:

a $x=\frac{y}{z}$.をクリックします。

b **Python コードの編集**ウィンドウで、ご使用の Python コードを入力します。コードには、ブール値を返すフィルター関数を含める必要があります。ESP サーバーは、入力イベント毎にこの関数を呼び出して、戻りコードを使用して、出力イベントを生成するかかどうかを決定します。この関数の名前が、**関数のフィルター**フィールドで指定されている名前と一致することを確認してください。

フィルターウィンドウに適した Python コードの詳細については、"Using Python in a Filter Window" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。コードエディタの詳細については、"コードと式を編集する"を参照してください。

c **OK** をクリックします。

11 フィルターウィンドウとプロジェクトの残りの部分に必要なその他の設定を構成します。

SAS Event Stream Processing Studio で Python ベースパターンウィンドウの操作

パターンウィンドウを使用すると、対象イベント(EOI)がプロジェクトを通過する際にそれらのイベントを検出し、パターンロジックを適用した結果として出力イベントを生成できます。EOI を定義するには、Lua、Python、または Expression Engine Language (EEL)を使用できます。

注: パターンウィンドウでは Lua または Python を使用することをお勧めします。パターンウィンドウで EEL を使用する機能は、従来の機能です。

このトピックでは、SAS Event Stream Processing Studio で Python ベースパターンウィンドウを作成する方法について説明します。パターンウィンドウの目的に関する一般的な情報については、"Using Pattern Windows" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

次の手順では、出力スキーマを含むソースウィンドウが既に含まれているプロジェクトがあることを前提としています。この手順では、パターンウィンドウを追加する方法について説明します。Python ベースのパターンウィンドウを含む完全なプロジェクトの例については、Python パターンのサンプルをインストールしてください。詳細については、"サンプルプロジェクトのインストール"を参照してください。

1 プロジェクトを開きます。

2 左の**ウィンドウ**ペイン上で**ユーティリティ**を展開し、ワークスペースにパターンウィンドウをドラッグします。

3 右側のウィンドウで、**パターン**を展開します。

4 パターンタイプを **Python** に設定します。

5 をクリックします。

新しいパターンウィンドウが表示されます。**新しいパターン**ウィンドウは 4 ページからなるウィザードです。

6 **初期化**ページを完了します:

a 必要に応じてパターンの名前を調整します。

b 必要に応じて、インデックスフィールドを選択します。入力ウィンドウからのこれらのフィールドは、インデックス生成関数を形成します。入力ウィンドウは、ソースウィンドウまたは別の種類のウィンドウである可能性があります。すべての着信イベントは、指定されたインデックスによってグループ化されます。

c 必要に応じて、**イベントのインデックス値に関係なく、オープンパターンが着信イベントごとにタイムアウトになる可能性を許可する**チェックボックスをオンにします。

d 必要に応じて、1 つ以上の時間フィールドを指定します。ウィンドウごとに、時間を指定するために使用するフィールドを選択できます。時間フィールドが指定されていない場合は、システム時間が使用されます。

e **次へ**をクリックします。

7 **Python コード**ページを完成させます:

a パターンに必要な EOI 関数と出力関数を含む、パターンの Python コードを入力します。

エディタコード使用の詳細については、"コードと式を編集する"を参照してください。EOI 関数の詳細については、"Writing EOI Functions in Python" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。出力関数の詳細については、"Writing Output Functions in Python" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

b **次へ**をクリックします。

8 **出力とイベント**ページに入力します。

   a 必要に応じて、 **初期化関数**フィールドを使用して、Python コード内のどの関数が初期化関数として機能するかを指定します。詳細については、“Initialization of Python Entities” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

   b **出力関数**フィールドで、出力関数の名前を選択します。出力関数は、前の手順で追加した**Python コード**に存在する必要があります。

   c 必要に応じて、**初期化関数**フィールドを使用して、Python コード内のどの関数が初期化関数として機能するかを指定します。クリーンアップ機能は、プロジェクトの停止時に実行されます。クリーンアップ関数は、必要な Python クリーンアップタスクを実行する必要があります。

   d **イベント**サブセクションで、をクリックし EOI を作成します。

   新しい行がテーブルに追加されます。

   e 名前列で、必要に応じてイベントの名前を調整します。

   f フィールドを含む列で、イベントに渡す Python フィールドを選択します。この列を空のままにすると、すべての Python フィールドがイベントに渡されます。

   g Python 列で、イベントに関連付ける EOI 関数を選択します。**Python 関数の編集**ウィンドウで入力したすべての関数は、この列で使用できます。出力関数ではなく、適切な EOI 関数を選択します。

   h イベントの送信列で、以前に一致したすべてのイベントを関数に送信する場合は、チェックボックスをオンにします。

   i 必要に応じて EOI を追加します。

   j **次へ**をクリックします。

9 **ロジック式**ページを使用して、パターンロジックを作成します。論理式は Python 言語を使用していません。終了したら、**OK** をクリックします。

   エディタコード使用の詳細については、“コードと式を編集する”を参照してください。論理式の記述の詳細については、“Applying Pattern Logic” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

10 必要に応じて、パターンを追加します。

11 をクリックします。

12 をクリックします。

   出力スキーマウィンドウが表示されます。

13 **出力スキーマ**ウィンドウを使用して、キーフィールドを指定し、出力フィールドとフィールドの種類を指定します。終了したら、**OK** をクリックします。

   注: パターンウィンドウ内のすべてのパターンには、同じ出力フィールドがあります。特定のパターンに対して異なる出力フィールドが必要な場合は、プロジェクトにパターンウィンドウを追加できます。

14 パターンウィンドウとプロジェクトの残りの部分に必要なその他の設定を構成します。

SAS Event Stream Processing Studio での StateDB ウィンドウの操作

概要

StateDB ライターウィンドウと StateDB リーダーウィンドウを使用すると、外部に配置された高性能データストアまたはデータベースを使用して、結合と集計に必要な"ステート(状態)"を保存および維持できます。外部ストアを使用すると、RAM を使用する必要性が減少します。StateDB ウィンドウの目的と例の詳細については、“Using StateDB Windows” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

StateDB ライターウィンドウを含むプロジェクトには、データを受け取るソースウィンドウ、そのデータを処理する機能ウィンドウ、およびデータをデータベースに書き込む StateDB ライターウィンドウが含まれる場合があります。

StateDB ライターウィンドウにつながるエッジは 1 つだけです。プロジェクトが複数のソースウィンドウからデータを受け取る場合は、結合ウィンドウを使用してデータを StateDB ライターウィンドウに渡します。

StateDB リーダーウィンドウを含むプロジェクトでは、StateDB リーダーウィンドウはデータベースからデータを読み取り、そのデータに対して集計を実行することもできます。次に、機能ウィンドウなどの他のウィンドウを使用して、データをさらに処理できます。

StateDB ライターウィンドウの構成

1 プロジェクトを開きます。

2 左側の**ウィンドウ**ペインの**ユーティリティ** を展開し、ワークスペースに StateDB ライターウィンドウをドラッグします。

3 StateDB ライターウィンドウを、StateDB ライターウィンドウにデータを提供する親ウィンドウに接続します。

4 右ペインで、**データベース接続**を展開します。

5 データベースタイプを選択します。Redis または SingleStore を選択できます。

6 データベースにアクセスするための設定を構成します。

   a データベースの種類が Redis の場合、単一の Redis ノードを使用しているか、クラスターを使用しているかを示す必要があります。

b ホストおよびポート情報を入力します。Redis クラスターを使用している場合は、**ホスト**テーブルを使用して、複数のホストとそのポートを入力します。

c データベースにアクセスするためのユーザー名とパスワードを入力します。

d (オプション)データベースの種類が Redis の場合、接続を維持するために ping コマンドを送信する間隔を秒単位で指定できます。デフォルト値は 15 秒です。単一の Redis ノードを使用している場合、ping は Redis サーバーに送信されます。Redis クラスターを使用している場合、ping は Redis クラスター内のすべてのノードに送信されます。

e データベースタイプが SingleStore の場合は、SingleStore データベースの名前を入力します。

f (オプション) データベース接続の名前を入力します。この名前は、ESP サーバーとデータストア間の接続のラベルです。この名前にはプロジェクトに対するその他の影響はありません。

g (オプション)データベースの種類が Redis の場合、トランスポート層セキュリティ(TLS)を使用するオプションを選択できます。詳細については、"Using Transport Layer Security (TLS) in StateDB Windows When Redis Is the External Data Store" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。TLS を使用するオプションを選択した場合は、必要に応じて次の詳細を指定します。

- サーバー名表示(SNI)を指定します。SNI は、クライアントが接続しようとしているホスト名を示すことができる TLS プロトコルの拡張機能です。
- 認証局(CA)の証明書ディレクトリまたは CA 証明書ファイルのいずれかを指定します。

  信頼された CA 証明書ファイルが OpenSSL 互換の構造で保存される CA 証明書ディレクトリを指定できます。TLS は、そのディレクトリにある認証アーティファクトをすべて 使用します。

  あるいは、CA 証明書ファイルを指定することもできます。このファイルは、CA 証明書、または既知の CA 証明書のセットを指定するバンドルファイルです。
- 相互認証が必要な場合は、クライアント側の証明書と秘密キーファイルを選択します。これら 2 つのファイルは一緒に使用されます。クライアント側の証明書を選択した場合は、秘密キーファイルも選択する必要があります。

h データベースタイプが Redis の場合は、**接続のテスト**をクリックして、資格情報が機能することを確認します。

注: TLS を使用するオプションが選択されている場合、**接続のテスト**ボタンは使用できません。

7 **データベース書き込みプロパティ**を展開し、イベントをデータベースに書き込むための設定を構成します。

a データベースの種類が Redis の場合は、Redis 接頭辞を入力します。接頭辞は、データをグループ化するために使用され、テーブル名として機能します。

b データベースタイプが SingleStore の場合は、SingleStore データベースの名前を入力します。指定された表がまだ存在していない場合は、この名前をもつ表が作成されます。

c "存続時間"設定を指定できます。

- **存続時間**フィールドを使用して、レコードの有効期限がいつ切れるかを示します。時間フィールドの名前を入力するか、秒単位の値を入力できます。
- **存続時間オフセット**フィールドを使用して、レコード削除を遅延するためのオフセットを秒数で指定します。

- **時間フィールド**ドロップダウンリストを使用して、レコードの有効期限となる時間を指定するためのフィールドを選択します。時間フィールドが指定されていない場合は、システム時間が使用されます。

d **セカンダリインデックス**テーブルで、セカンダリインデックスの形成に使用されるフィールドを指定します。テーブルの各行は、異なるセカンダリインデックスに関連しています。

e データベースタイプが Redis の場合は、セカンダリインデックスの古いキーを削除するオプションを選択できます。

f **出力**テーブルで、データベースに書き込まれるイベントの出力フィールドを指定します。つまり、テーブルを使用して、ESP プロジェクトのフィールドをデータベースのフィールドにマップします。

g データベースタイプが SingleStore の場合は、書き込み前に SingleStore 表からすべての行を削除するオプションを選択できます。

注意
このオプションを使用する場合は注意してください。失われたデータを回復することはできません。例えば、このオプションは、本番環境で使用されるプロジェクトには適切でない場合があります。

これらの設定の詳細については、"Elements That Are Specific to window-statedb-writer" (*SAS Event Stream Processing: XML Language Reference for Event Stream Processing Models*)を参照してください。

8 StateDB ライターウィンドウに必要なその他の設定を構成します。

StateDB リーダーウィンドウの構成

1 プロジェクトを開きます。

2 左側の**ウィンドウ**ペインの**ユーティリティ** を展開し、ワークスペースに StateDB リーダーウィンドウをドラッグします。

3 右ペインで、**データベース接続**を展開します。

4 データベースタイプを選択します。Redis または SingleStore を選択できます。

重要　StateDB Reader ウィンドウが Singlestore テーブルから読み取る場合、そのテーブルには次の 2 つのフィールドが含まれている必要があります。

- esp_meta_timestamp
- esp_meta_ttd

これらのフィールドは、Singlestore テーブルからのフィールドの削除を管理するために SAS Event Stream Processing によって使用されます。これらは、StateDB ライターウィンドウによってテーブルが作成されるときに自動的に作成されます。別の方法で作成された

テーブルを使用する場合は、これらの 2 つのフィールドを BIGINT 型として手動で追加する必要があります。

5 データベースにアクセスするための設定を構成します。

a データベースの種類が Redis の場合、単一の Redis ノードを使用しているか、クラスターを使用しているかを示す必要があります。

b ホストおよびポート情報を入力します。Redis クラスターを使用している場合は、**ホスト**テーブルを使用して、複数のホストとそのポートを入力します。

c データベースにアクセスするためのユーザー名とパスワードを入力します。

d (オプション)データベースの種類が Redis の場合、接続を維持するために ping コマンドを送信する間隔を秒単位で指定できます。デフォルト値は 15 秒です。単一の Redis ノードを使用している場合、ping は Redis サーバーに送信されます。Redis クラスターを使用している場合、ping は Redis クラスター内のすべてのノードに送信されます。

e データベースタイプが SingleStore の場合は、SingleStore データベースの名前を入力します。

f (オプション) データベース接続の名前を入力します。この名前は、ESP サーバーとデータストア間の接続のラベルです。この名前にはプロジェクトに対するその他の影響はありません。

g (オプション)データベースの種類が Redis の場合、トランスポート層セキュリティ(TLS)を使用するオプションを選択できます。詳細については、"Using Transport Layer Security (TLS) in StateDB Windows When Redis Is the External Data Store" (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。TLS を使用するオプションを選択した場合は、必要に応じて次の詳細を指定します。

- サーバー名表示(SNI)を指定します。SNI は、クライアントが接続しようとしているホスト名を示すことができる TLS プロトコルの拡張機能です。
- 認証局(CA)の証明書ディレクトリまたは CA 証明書ファイルのいずれかを指定します。

  信頼された CA 証明書ファイルが OpenSSL 互換の構造で保存される CA 証明書ディレクトリを指定できます。TLS は、そのディレクトリにある認証アーティファクトをすべて 使用します。

  あるいは、CA 証明書ファイルを指定することもできます。このファイルは、CA 証明書、または既知の CA 証明書のセットを指定するバンドルファイルです。
- 相互認証が必要な場合は、クライアント側の証明書と秘密キーファイルを選択します。これら 2 つのファイルは一緒に使用されます。クライアント側の証明書を選択した場合は、秘密キーファイルも選択する必要があります。

h データベースタイプが Redis の場合は、**接続のテスト**をクリックして、資格情報が機能することを確認します。

注: TLS を使用するオプションが選択されている場合、**接続のテスト**ボタンは使用できません。

6 **データベースクエリ**を展開し、イベントをデータベースから読み取るための設定を構成します。

a データベースが Redis の場合は、クエリするデータの Redis 接頭辞を入力します。

b データベースタイプが SingleStore の場合は、クエリする表の名前を入力します。

c **クエリ**テーブルを使用して、データベースクエリを作成します。クエリは、データベースからレコードを取得する条件を指定します。

- **クエリ**表内で選択するのに利用できる入力フィールドは、StateDB Writer ウィンドウの親ウィンドウのスキーマを反映しています。
- データベースの種類が Redis の場合、使用できる唯一の演算子は==です。つまり、データベースクエリは入力フィールドを Redis のフィールドに一致させます。
- データベースタイプが SingleStore の場合、SingleStore でサポートされる追加演算子は、**クエリ**表で利用できます。

d 時間フィールドを選択して、レコードの有効期限を管理できます。時間フィールドが指定されていない場合は、システム時間が使用されます。

e 削除イベントの生成を選択できます。プロジェクトに、集計ウィンドウなどの後続ステートフルウィンドウが含まれている場合は、このオプションの選択を検討してください。

詳細については、*"Elements That Are Specific to window-statedb-reader" (SAS Event Stream Processing: XML Language Reference for Event Stream Processing Models)*を参照してください。

7 をクリックし、をクリックします。**出力スキーマ** ウィンドウを使用して、StateDB リーダーウィンドウの出力を指定します。

- ソース列を使用して、特定のフィールドのデータがデータベースクエリから取得されたのか、それともデータが入力イベントから取得されたのかを示します。
- データベースクエリから取得されたデータの場合は、ソースフィールド列を使用してデータベースフィールドの名前を示します。入力イベントから取得されたデータの場合、ソースフィールド列を使用して入力フィールドの名前を示します。
- 集計を実行する場合は、集計関数列を使用して集計関数を選択します。集計関数列で使用できる集計関数は、集計ウィンドウで使用できる関数のサブセットです。関数の動作は同じです。集計関数の詳細については、*"Using Aggregate Functions" (SAS Event Stream Processing: Using Source and Derived Windows)*を参照してください。

> ヒント　StateDB リーダーウィンドウの集計関数は、データベースクエリから取得されるデータで使用することを目的としています。別のソースから取得されるデータで集計を実行するには、代わりに集計ウィンドウを使用します。

- 該当する場合は、パラメーター列を使用して関数パラメーターを指定します。ESP_aCat 関数および ESP_aLag 関数のみがパラメーターを持ちます。

8 StateDB リーダーウィンドウに必要なその他の設定を構成します。

カスタムウィンドウ操作

カスタムウィンドウの概要

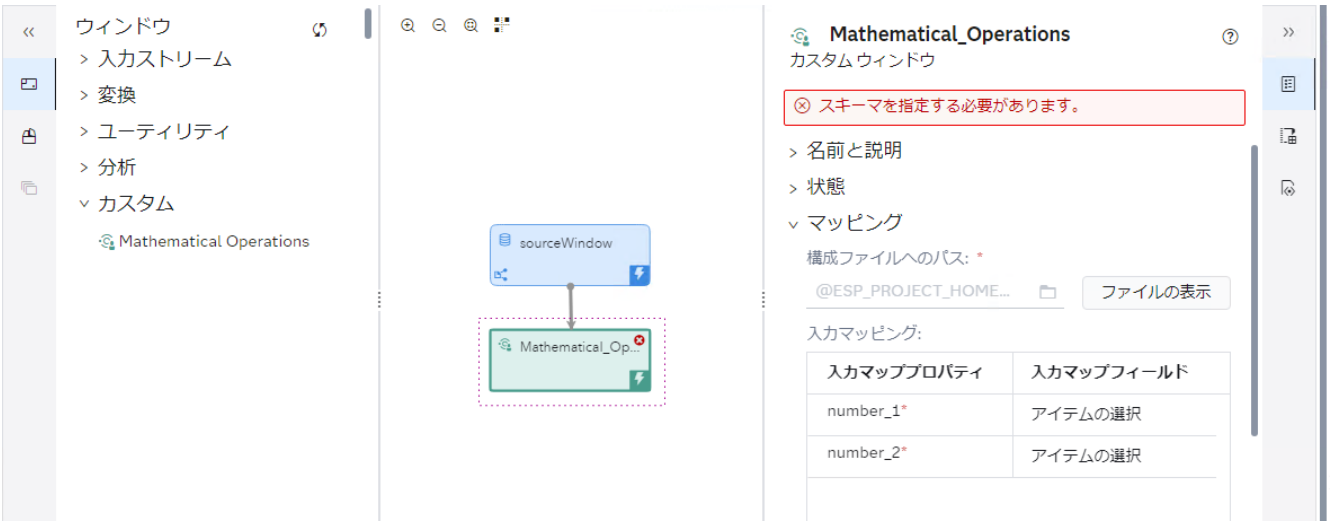
カスタムウィンドウを使用すると、開発者は、ユーザーがプロジェクトに簡単に追加できる新しいウィンドウの種類を定義できます。つまり、開発者は、SAS が提供するウィンドウの種類に隣接する**ウィンドウ**ペインの**カスタム**セクションに表示されるウィンドウの種類を作成できます。

カスタムウィンドウにより、プロジェクト間でコードを再利用できます。カスタムウィンドウを作成するには、開発者が次の構成を指定する必要があります。

- 受信および送信イベントに対してウィンドウが実行するアクション
- ユーザーがカスタムウィンドウを使用するときに SAS Event Stream Processing Studio Modeler で構成できる設定

カスタムウィンドウを指定するコードには、Python コードまたは Lua コードと JSON コードが含まれます。開発者は、ウィザードを使用してカスタムウィンドウを作成します。このウィザードには、開発者がサードパーティのコードエディターを使用して作成した構成ファイルをアップロードするオプションが含まれています。開発者は、既存の Python ウィンドウまたは Lua ウィンドウをカスタムウィンドウに変換することもできます。

図 36　右ペインでユーザーが利用できるカスタムウィンドウ設定の例



開発者がカスタムウィンドウを作成する方法については、“カスタムウィンドウの作成と管理”を参照してください。

カスタムウィンドウの使用方法については、“カスタムウィンドウの使用”を参照してください。

カスタムウィンドウの例と、SAS Event Stream Processing Studio に追加できるカスタムウィンドウの取得については、SAS Event Stream Processing Studio Custom Windows GitHub を参照してください。

カスタムウィンドウの作成と管理

次のトピックでは、**カスタムウィンドウ**ページの使用方法について説明します。カスタムウィンドウの作成および管理は、ユーザーではなく、開発者向けのタスクです。

カスタムウィンドウの作成

このトピックでは、**新規カスタムウィンドウ**ウィザードの使用方法に関する一般的な情報を提供します。

> ヒント　2 つの数字を乗算するカスタムウィンドウの作成方法を示す例については、SAS Event Stream Processing Studio Custom Windows GitHub の入門ガイドの例を参照してください。**カスタムウィンドウ**ページで**カスタムウィンドウ GitHub の表示**をクリックして、この GitHub にアクセスすることもできます。

カスタムウィンドウの作成は次のとおりです:

カスタムウィンドウページで  をクリックします。

> ヒント　一からカスタムウィンドウを作成するのではなく、既存の Lua ウィンドウや Python ウィンドウをカスタムウィンドウに変換することもできます。詳細については、“Lua ウィンドウまたは Python ウィンドウをカスタムウィンドウに変換する” (153 ページ)を参照してください。

新規カスタムウィンドウウィンドウは 3 ページからなるウィザードです。

1 **初期化**ページを完了します。

   a (オプション)お好みのコードエディターを使用して作成した構成ファイル、または SAS Event Stream Processing Studio Custom Windows GitHub から取得した構成ファイルをアップロードします。

      構成ファイルをアップロードする場合は、そのフォーマットが要件を満たしていることを確認してください。詳細については、“Creating the Configuration File” (*SAS Event Stream Processing: Using Source and Derived Windows*)を参照してください。

      構成ファイルをアップロードすると、ウィザード全体のフィールドにデータが入力されます。構成ファイルをアップロードした後、ウィザードを使用して、カスタムウィンドウでさらに構成を変更できます。これらのさらなる変更は、コンピューターからアップロードした元の構成ファイルには保存されません。ただし、**新規カスタムウィンドウバージョン**ウィンドウの使用が終了したら、カスタムウィンドウの現在の構成を表す構成ファイルをエクスポートできます。詳細については、“構成ファイルのエクスポート”を参照してください。

   b カスタムウィンドウの名前を入力します。

c 必要に応じて、カスタムウィンドウの説明、タグ、バージョンのメモを入力します。この情報は、**ウィンドウ**ペインのカスタムウィンドウにカーソルを合わせたときにユーザーに表示され、適切なウィンドウを選択するのに役立ちます。

d **次へ**をクリックします。

2 **構成**ページで、**入力変数**タブを使用して、入力データをカスタムウィンドウの create 関数にマップします。

注: **変数マッピングを有効にする**チェックボックスが選択されていない場合、親ウィンドウのすべてのフィールドが渡されます。

- **入力変数の説明**フィールドに、カスタムウィンドウのユーザーに表示される説明を入力します。この説明が表示される場所を示す例を次に示します。

- テーブルの名前の列に、create 関数に渡されるオブジェクトの入力に使用される名前を入力します(パラメーターが展開されていない場合)。
- 説明列に、フィールドの説明を入力します。
- 必要に応じて、種類列で、カスタムウィンドウのユーザーが各入力変数にマップできるフィールドのデータ型を選択します。
- 必要に応じて、オプション列を使用してフィールドにオプションとしてマークします。フィールドがオプションの場合、カスタムウィンドウのユーザーは、そのフィールドのマッピングエントリを入力する必要はありません。

3 **構成**ページの**出力変数**タブを使用して、create 関数からの出力データを出力のイベントフィールドにマップします。

- **出力変数の説明**フィールドに、カスタムウィンドウのユーザーに表示される説明を入力します。
- テーブルで、各フィールドの名前と説明を入力します。
- 必要に応じて、種類列で、カスタムウィンドウのユーザーが各出力変数にマップできるフィールドのデータ型を選択します。

4 **構成**ページの**設定**タブを使用して、設定を有効または無効にします。

- **設定の説明**フィールドに、これらの設定の使用方法の説明を入力します。この説明は、このカスタムウィンドウを管理する開発者に注意を向けるためのものであり、カスタムウィンドウのユーザーには表示されません。

- **パラメーターの展開(expand-parms)**設定は、create 関数は、各入力変数に対して個別のパラメータを受け取るか、入力変数の名前にマップされるフィールドを含むオブジェクトを受け取るかを決定します。ウィザードの**コード**ページの create 関数を指定します。

  注: **パラメーターの展開(expand-parms)**設定が選択されている場合、**入力変数**タブを使用して入力変数を定義する必要があります。

- **ブロックのイベントの処理(process-blocks)**設定では、create 関数は、複数のイベントまたは 1 つのイベントを含むブロックのイベントを一度に受け取ります。

  注: **パラメーターの展開(expand_parms** 設定と **process_blocks** 設定の両方を同時に有効にすることはできません。

- **バイナリデータのエンコード(Pythonencode-binary)**設定は、create 関数に渡されるバイナリデータが Base64 でエンコードされるかどうかを決定します。

5 **構成**ページの**初期化**タブを使用して、init 関数に渡されるオブジェクトに関する情報を追加します。カスタムウィンドウは値を、単一のデータオブジェクトに配置し関数に渡します。**初期化**タブにアイテムを追加する場合は、ウィザードの**コード**ページで init 関数を指定する必要があります。

注: **変数マッピングを有効化**チェックボックスが選択されていない場合、ウィンドウでは、Python または Lua コードが返すオブジェクトと一致するように出力イベントのフィールドが設定されます。

- **初期化の説明**フィールドに、カスタムウィンドウのユーザーに表示される説明を入力します。
- テーブルで、各プロパティの名前と説明を入力します。
- 入力の種類列で、初期化値をテキストフィールドとしてカスタムウィンドウのユーザーに表示するか、ドロップダウンリストで表示するかを選択します。
- 入力タイプ列が**ドロップダウン**に設定されている場合、値列が有効になります。値列を使用して、カスタムウィンドウのユーザーが選択できる初期化値を指定します。
- デフォルト値列で、各プロパティのオプションのデフォルト値を指定します。この値は、モデルが<plugin>要素に値を提供しない場合に使用されます。デフォルト値が指定されておらず、<plugin>要素にエントリが含まれていない場合、プロパティは init 関数に送信されません。

6 **コード**ページを使用して、Python コードまたは Lua コードを追加します。必須の create 関数およびその他の入力できるコードについては、*"Working with Custom Windows" (SAS Event Stream Processing: Using Source and Derived Windows)*を参照してください。

コードを実行するために必要な Python パッケージまたは Lua モジュールを指定することもできます。ユーザーがカスタムウィンドウをプロジェクトに追加するか、カスタムウィンドウを含むプロジェクトを開くと、適切な仮想環境がまだ選択されていない場合は、要件に一致する仮想環境を選択するようにユーザーに求められます。

表 4　バージョンを指定するためのオペレーター

オペレーター	説明	関連する仮想環境
==	正確なパッケージバージョンまたはモジュールバ	このオペレーターは、Python 仮想環境および

オペレーター	説明	関連する仮想環境
	ージョンを使用してください。	Lua 仮想環境に関連します。
~=	適切なマイナーバージョンを使用してください。たとえば、演算子列で**~=**を選択し、インストールされたバージョン列に **1.2** と入力すると、SAS Event Stream Processing は仮想環境でバージョン 1.2.1 を使用できますが、バージョン 1.3 は使用できません。	このオペレーターは Python 仮想環境に関連します。
>=	指定された最小バージョンまたは新しいバージョンを使用してください。	このオペレーターは Python 仮想環境に関連します。
<	指定されたバージョンよりも古いバージョンを使用してください。	このオペレーターは Python 仮想環境に関連します。

7 **作成**をクリックします。

作成したウィンドウの種類は、**カスタムウィンドウ**ページのテーブルにリストされ SAS Event Stream Processing Studio モデラーの**ウィンドウ**ペインで使用できます。

Lua ウィンドウまたは Python ウィンドウをカスタムウィンドウに変換する

プロジェクトがプロジェクトパッケージの一部であり、プロジェクトに Lua ウィンドウまたは Python ウィンドウが含まれている場合、そのウィンドウをカスタムウィンドウに変換できます。

1 プロジェクトを開いて変換するウィンドウを選択します。

2 右ペインで、をクリックします。

3 必要に応じてカスタムウィンドウの構成を調整するには、**新規カスタムウィンドウ**の作成ウィザードを使用します。ウィザードの使用方法の詳細については、“カスタムウィンドウの作成” (150 ページ)を参照してください。

作成をクリックしてウィザードを完了すると、プロジェクト内の Lua ウィンドウまたは Python ウィンドウがカスタムウィンドウに変更されます。

新しいカスタムウィンドウは、**ウィンドウ**ペインにも表示されます。**カスタムウィンドウ**ページでカスタムウィンドウを管理できます。

カスタムウィンドウのバージョンの作成

カスタムウィンドウのバージョンを作成すると、以前のバージョンを保持しながらカスタムウィンドウに更新を行うことができます。

個別のカスタムウィンドウを作成するのではなく、同じカスタムウィンドウの複数のバージョンを作成することを検討してください。作成した各カスタムウィンドウは、SAS Event Stream Processing Studio モデラーの**ウィンドウ**ペインに表示されます。**ウィンドウ**ペインに多くのカスタムウィンドウが存在すると、ユーザーが混乱する可能性があります。

注: プロジェクトに同じカスタムウィンドウのインスタンスが複数含まれている場合、それらのカスタムウィンドウすべてのバージョンが同じである必要があります。

バージョンの作成は次のとおりです:

1 **カスタムウィンドウ**ページで、テーブルの関連する行を選択し、をクリックします。

2 **新規カスタムウィンドウバージョン**ウィンドウで、更新された構成ファイルを選択するか、必要に応じて他のフィールドを調整します。

3 **新規バージョン**をクリックします。

作成した更新をエンドユーザーが取得する方法については、“カスタムウィンドウのバージョンの管理”を参照してください。

構成ファイルのエクスポート

カスタムウィンドウの構成ファイルをエクスポートできます。

1 **カスタムウィンドウ**ページで、テーブルの関連する行を選択します。

2 ツールバーのをクリックし、**構成ファイルのエクスポート**を選択します。

カスタムウィンドウ定義を含む構成ファイルがコンピューターにエクスポートされます。

注: エクスポートファイルの場所は、ブラウザの構成によって異なる場合があります。

構成ファイルのインポート

以前にエクスポートした構成ファイル、または SAS Event Stream Processing Studio Custom Windows GitHub から取得した構成ファイルをインポートできます。

1 **カスタムウィンドウ**ページで、をクリックし、**カスタムウィンドウのインポート**を選択します。

新規カスタムウィンドウウィンドウが表示されます。

2 **初期化**タブで、**構成ファイル**フィールドを使用して構成ファイルを選択します。

構成ファイルを選択すると、ウィザード全体のフィールドにデータが入力されます。これらのフィールドの詳細については、“カスタムウィンドウの作成”を参照してください。

3 **作成**をクリックします。

構成ファイルで定義されたカスタムウィンドウは、**カスタムウィンドウ**ページの表にリストされ、SAS Event Stream Processing Studio モデラーの**ウィンドウ**ペインで使用できます。

カスタムウィンドウの削除

カスタムウィンドウページで、テーブルの関連する行を選択し、  をクリックします。

そのカスタムウィンドウのすべてのバージョンが削除され、ユーザーは SAS Event Stream Processing Studio Modeler の**ウィンドウ**ペインでカスタムウィンドウを選択できなくなります。

カスタムウィンドウの使用

次のトピックは、開発者が作成したカスタムウィンドウを使用するユーザーを対象としています。

プロジェクトにカスタムウィンドウを追加

1 カスタムウィンドウを追加するプロジェクトを開きます。

2 **ウィンドウ**ペインで**カスタム**セクションを展開します。

3 目的のカスタムウィンドウをワークスペースにドラッグします。

4 カスタムウィンドウでは、特定の Python パッケージまたは Lua モジュールを仮想環境で使用できるようにする必要がある場合があります。必要なライブラリと選択した仮想環境の間の不一致を解決するようにメッセージが表示された場合は、**解決**をクリックし、**仮想環境**ウィンドウを使用して一致する仮想環境を選択します。システム管理者が仮想環境を作成する方法の詳細については、“仮想環境の指定”を参照してください。

5 親ウィンドウをカスタムウィンドウに接続してカスタムウィンドウの入力イベントと入力スキーマを提供します。

6 右ペインの**カスタム**セクションで、次の設定を構成します。

注:

- これらの設定について質問がある場合は、カスタムウィンドウを作成した開発者にお問い合わせください。
- カスタムウィンドウの構成ファイルへのパスは、読み取り専用フィールドです。

a 入力マップを構成します。つまり、ウィンドウの Python または Lua コード内の create 関数に渡されるプロパティにマップする入力スキーマのフィールドを選択します。

> ヒント　入力マッププロパティ列で、赤いアスタリスクは、このプロパティを必須プロパティに設定したこのカスタムウィンドウの開発者を示します。この列にⓘが表示されている場合、それをマウスを合わせると、そのプロパティに関するカスタムウィンドウの開発者からの情報が表示されます。

b 出力マップを構成します。つまり、 create 関数から返されたデータが、出力スキーマのフィールドにどのようにマップされるのかを指定します。

> ヒント　出力マッププロパティ列で、赤いアスタリスクは、このプロパティを必須プロパティに設定したこのカスタムウィンドウの開発者を示します。この列にⓘが表示されている場合、それをマウスを合わせると、そのプロパティに関するカスタムウィンドウの開発者からの情報が表示されます。

Python コードまたは Lua コードの出力をウィンドウにマップできるようにウィンドウの出力スキーマにさらにフィールドを追加する必要がある場合があります。

c 初期化マップを構成します。つまり、ウィンドウの Python または Lua コード内の init 関数に渡される値を選択します。

7 必要に応じて、**致命的なエラーが発生した場合のエラー処理の方法**フィールドを使用して、エラー処理の方法を選択します。

Lua または Python コードは例外が発生する可能性があります。init 関数からの例外により、プロジェクトの読み込みに失敗します。デフォルトでは、create 関数、heartbeat 関数、およびスニペット(スレッド化された場合)からの例外は、プロジェクトレベルのエラー処理設定に基づいて処理されます。必要に応じて、カスタムウィンドウ用に別のエラー処理の戦略を選択して、プロジェクトレベルの設定を無効にできます。エラーをログに記録してプロジェクトの実行を続行するか、致命的なエラーをログに記録してプロジェクトを停止するかを選択できます。プロジェクトレベルの設定を確認するには、をクリックし、右側のペインで**詳細**を展開して、選択されているエラー処理戦略を確認します。

8 必要に応じて、右ペインでその他のウィンドウのプロパティを編集します。

9 プロジェクトを保存します。

プロジェクトを保存すると、カスタムウィンドウと同じ名前の読み取り専用フォルダーが、プロジェクトパッケージの **custom_windows** フォルダーに追加されます。フォルダーには、カスタムウィンドウの構成ファイルが含まれます。ワークスペースに追加したウィンドウの名前を変更する場合(右側のペインの**名前**フィールド)、プロジェクトパッケージ内のフォルダーは、カスタムウィンドウの元の名前をまだ反映しています。プロジェクトからカスタムウィンドウを削除すると、フォルダーと構成ファイルも削除されます。

カスタムウィンドウのバージョンの管理

開発者がカスタムウィンドウのバージョンを作成し、そのカスタムウィンドウを**ウィンドウ**ペインからプロジェクトのワークスペースにドラッグすると、最新バージョンのカスタムウィンドウが使用されます。詳細については、“カスタムウィンドウのバージョンの作成”を参照してください。

さらに、そのカスタムウィンドウをワークスペースにすでに含んでいるプロジェクトを開くと、新しいバージョンが利用可能であることを知らせるメッセージが表示されます。カスタムウィンドウのバージョンを更新するには、次のようにします。

1 カスタムウィンドウを含むプロジェクトを開き、**バージョンマネージャー**をクリックします。

2 **カスタムウィンドウバージョンマネージャー**ウィンドウを使用して、影響を受けるカスタムウィンドウの新しいバージョンを選択します。

3 既存の入力、出力、および初期化のマッピングを保持しない場合は、マッピングを保持しないカスタムウィンドウごとにマッピングの保持列のチェックボックスをオフにします。デフォルトでは、このチェックボックスは選択されています。

4 **更新**をクリックします。

プロジェクトを開き、ツールバーの⋮をクリックして**カスタムウィンドウバージョンの管理**を選択すると、いつでも**カスタムウィンドウバージョンマネージャー**ウィンドウにアクセスできます。たとえば、カスタムウィンドウを以前のバージョンに戻したい場合は、**カスタムウィンドウバージョンマネージャー**ウィンドウを使用して以前のバージョンを選択できます。

注: プロジェクトに同じカスタムウィンドウのインスタンスが複数含まれている場合、それらのカスタムウィンドウすべてのバージョンが同じである必要があります。

設定の管理

Overview of Settings

Use the Settings window to edit user preferences or customize accessibility settings for all SAS web applications. For more information, see “Settings Window”.

Use the **ESP Settings** page to adjust how certain features of SAS Event Stream Processing Studio and SAS Event Stream Manager work. For more information, see “設定ページへのアクセス”.

Settings Window

Use the Settings window to edit user preferences or customize accessibility settings for all SAS web applications. Changing these settings does not impact other users.

To access these settings, click the user button on the application bar, and select **Settings**.

General

The **General** section includes settings that enable users to change the appearance of the web applications, enable warning and information messages to be displayed, and choose a profile picture. Here are the settings:

- You can change the appearance of the web applications by using the **Theme** setting. The default theme is set by an administrator. The theme specifies the collection of colors, graphics, and fonts that appear in the application.

  Select a theme from the drop-down list to change the look of the applications. The theme change takes effect immediately.

  SAS themes:

  High Contrast
  Presents a black background with high-contrast foreground elements to meet the needs of users with low vision.

  Dark
  Presents an all-dark color palette for a better low-light experience.

  Light
  Presents a light and clean color palette. The Light theme is the default application theme.

- If you want messages to display that you previously asked not to display, click **Reset Messages**. By default, all warning and information messages are displayed.

  Note: This setting relates to messages from other SAS products, rather than messages from SAS Event Stream Processing Studio or SAS Event Stream Manager.

- You can select a profile picture to display in the application bar, as well as in other places within the application that use profile pictures. The default is the first initial or character of your first name.

  Note: When standalone SAS Event Stream Processing is deployed, you cannot select a profile picture.

  Click **Choose image** and then select an image file to upload. The file size of the image can be up to 1 MB. The valid file types are BMP, GIF, JPEG, JPG, and PNG.

  To remove an existing profile picture, click the down arrow and select **Remove image**.

Region and Language

The **Region and Language** section enables users to specify the locale for regional formats and sorting.

The **Locale for regional formats and sorting** setting specifies the locale that is used for sorting data and formatting values such as dates, times, numbers, and currency. By default, the browser locale is used. For example, you might use this setting to display the user interface in

English while using a different locale, such as Japanese, to sort and format dates, times, numbers, and currency.

Note: Select the *Browser locale* option only if your browser's locale is one of the supported locales listed in the **Locale for regional formats and sorting** setting. This avoids errors when sorting or formatting dates, times, numbers, and currency.

TIP Changes take effect after you sign out and sign back in.

Notifications

The **Notifications** section enables users to adjust how notifications from other SAS products are displayed.

Select **Show a pop-up message** to display a pop-up message when a notification arrives from another SAS product.

Note: When standalone SAS Event Stream Processing is deployed, the **Notifications** section is not available.

Accessibility

Several settings in the **Accessibility** section can assist people who rely on assistive technologies.

- Select **Adjust the display duration for pop-up notifications** to specify how long temporary pop-up notification messages are displayed. Increasing the amount of time that notifications are displayed can help users discover and read messages that disappear automatically.
- Select **Invert application colors** to make the user interface easier to see for users with sensitivity to certain bright colors. You can also use the Ctrl+` (Ctrl+back quote) keyboard shortcut to invert the application colors.
- Select **Display tooltips when using the keyboard to navigate** to enable keyboard users to access tooltips. When this option is selected, putting keyboard focus on a control also displays the tooltip on the screen. You can also select the location in the browser window to display the tooltip. By default, the tooltip is displayed in the bottom right corner of the browser window.
- The focus indicator is an outline that indicates which user interface component is active. You can make the focus indicator easier to see by selecting **Customize the focus indicator** and adjusting its color, thickness, and opacity.

設定ページへのアクセス

設定ページでは、SAS Event Stream Processing Studio と SAS Event Stream Manager の特定の機能がどのように動作するなどのタスク調整できます。

設定ページにアクセスするには、アプリケーションバーで、表示名またはユーザー ID の最初の文字をクリックして、**設定**を選択します。

SAS Event Stream Processing Studio の**設定**ページで利用できるコンテンツは、SAS Event Stream Manager の**設定**ページで利用できるコンテンツとは異なります。各アプリケーションのコンテンツは、これらのアプリケーションで実行するタスクに関連しています。

設定ページで使用できるコンテンツは、管理者グループのメンバーかどうかによって 異なります。デフォルトでは、管理者グループは SASAdministrators です。

グローバル設定

グローバル設定の概要

設定ページの**グローバル**セクションでは、アプリケーションのすべてのユーザーに影響する設定を変更できます。これの例外は、**テーマ**設定で、ブラウザーにのみ適用されます。

アプリケーションテーマの変更

テーマ設定を使用して、SAS Event Stream Processing Studio および SAS Event Stream Manager の表示を変更できます。テーマは、アプリケーションに表示される色、グラフィック、およびフォントのコレクションを指定します。**テーマ**設定はブラウザーにのみ適用されます。

アプリケーションテーマを変更するには、**設定**ページで、**グローバル**セクションの**全般**ページに移動します:

- **ライト**テーマは、明るいクリーンなカラーパレットを提供します。ライトテーマはデフォルトのアプリケーションテーマです。
- **ダーク**テーマは、暗い場所での体験を向上させるために、暗めのカラーパレットを提供します。
- **ハイコントラスト**テーマは、ロービジョンのユーザーのニーズを満たすために、ハイコントラストの前景要素を持つ暗い背景を提供します。

アクセシビリティ 設定の変更

キーボード使用でナビゲートする際、ツールチップを表示するオプションを選択して、キーボードユーザーがツールチップにアクセスできるようにします。このオプションを選択した場合、コントロールにキーボードフォーカスを置くと、画面上ににツールチップも表示されます。

この設定を変更するには、次のようにします。

1 **設定**ページで、**グローバル**セクションの**全般**ページに移動します。

2 **アクセシビリティ**セクションで、**キーボード使用でナビゲートする際、ツールチップを表示する**チェックボックスを選択します。

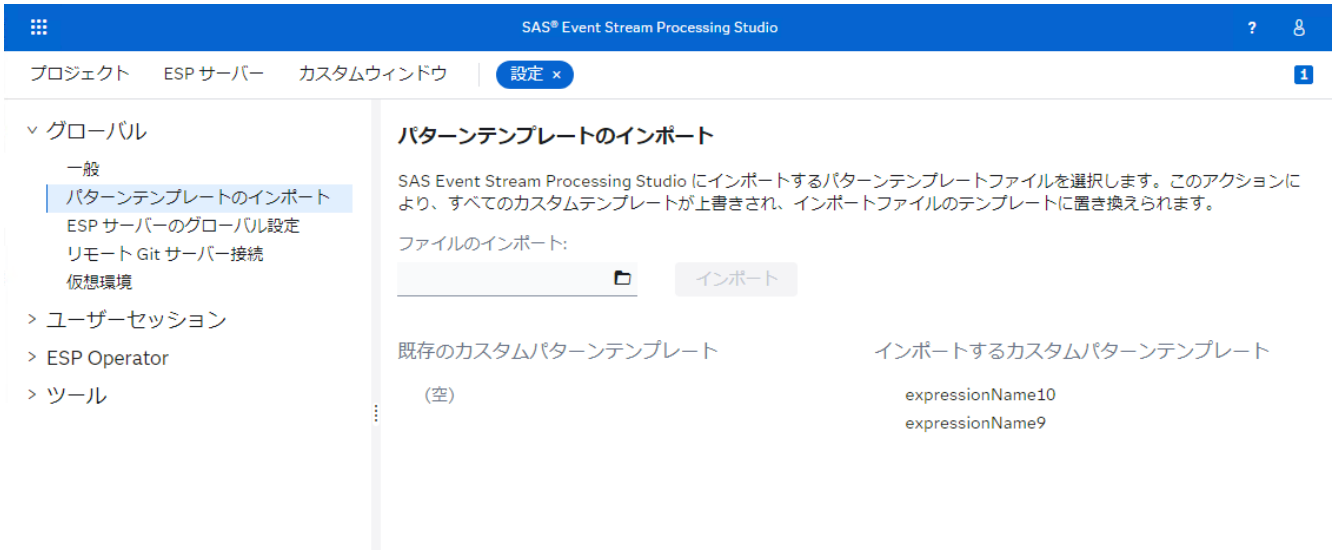
SAS Event Stream Processing Studio へのパターンテンプレートのインポート

SAS Event Stream Processing Studio に、カスタムパターンテンプレートをインポートできます。パターンテンプレートは、対象のイベント(EOI)を組み合わせるための論理演算子式を表すテキスト文字列を指定します。囲まれたテキストは、パターンを定義する式を指定します。

インポートするパターンテンプレートは、インポートファイルにある必要があります。一連のカスタムパターンテンプレートをインポートすると、以前にインポートされた既存のカスタムパターンテンプレートが上書きされて置換されます。

下図は、2 つのカスタムパターンテンプレートを含むパターンテンプレートファイルの例を示しています。

図 37　パターンテンプレートのインポート



注: 空のパターンテンプレートをインポートすると、それまでにインポートされたすべてのパターンは、新しい空のパターンテンプレートで上書きされます。

1 **設定**ページの**グローバル**セクションで**パターンテンプレートのインポート**を選択します。

右ペインには、**パターンテンプレートのインポート**ページが表示されます。

2 **ファイルのインポート**フィールドで、🗀をクリックします。

3 インポートするパターンテンプレートを含むファイルに移動し、**開く**をクリックします。

パターンテンプレートのインポートページが最新の情報に更新され、ファイルに存在するカスタムパターンテンプレートのリストが表示されます。また、以前にインポートした既存のカスタムパターンテンプレートも表示されます。

次のコードは、パターンテンプレートの例を示しています。

```
<pattern-templates>
  <pattens-template name= "expression1">
    <pattern-expression><![CDATA[ and(fby{5 seconds}(A, B, C, D), notoccur(K))]]></pattern-expression>
  </pattens-template>
  <pattens-template name= "expression2">
    <pattern-expression><![CDATA[and(fby{5 minutes}(A, B, C, D), notoccur(L))]]></pattern-expression>
  </pattens-template>
</pattern-templates>
```

4 ファイルの選択をクリアするには、⊗をクリックします。

5 インポートするカスタムパターンテンプレートの上にポインタを置きます。

注: パターンテンプレートの論理演算子式がメッセージボックスに表示されます。

6 **インポート**をクリックします。

警告メッセージが表示され、このアクションにより既存のすべてのカスタムパターンテンプレートが上書きされ、それらがファイルのパターンテンプレートで置換されることが通知されます。

7 続ける場合は、**はい**をクリックします。

インポートが成功したかどうかを通知するメッセージが表示されます。インポートしたパターンテンプレートは、パターンウィンドウのイベントロジックを定義するときに使用できます。

クライアント構成サーバーの再起動

状況によっては、クライアント構成サーバーを再起動する必要があります。詳細については、"クライアント構成サーバーの理解と管理" (*SAS Event Stream Processing: Kubernetes 環境での SAS Event Stream Processing の使用*)を参照してください。

> 重要　Client-Configuration Server を再起動すると、アプリケーションのすべてのユーザーに影響があります。

1 **設定**ページの**グローバル**セクションで **ESP サーバーのグローバル設定**を選択します。

注: **ESP サーバーのグローバル設定**ページにアクセスできるのは管理者グループのメンバーのみです。デフォルトでは、管理者グループは SASAdministrators です。

右ペインには **ESP サーバーのグローバル設定**ページが表示されます。

2 をクリックします。

ステータスフィールドには次のメッセージが表示されます。 クライアント構成サーバーの再起動中です。

ステータスフィールドに次のメッセージが表示されると、再起動が完了します。クライアント構成サーバーが実行されています。

コネクタを含むか含まないか

ESP サーバーのグローバル設定ページの**コネクタ**タブを使用して、コネクタの ESP サーバーのプロパティを変更することができます。これらの ESP サーバーのプロパティは、**コネクタ構成**ウィンドウで SAS Event Stream Processing Studio のユーザーが利用できるコネクタを制御します。

注: **ESP サーバーのグローバル設定**ページにアクセスできるのは管理者グループのメンバーのみです。デフォルトでは、管理者グループは SASAdministrators です。

ヒント　**コネクタ構成**ウィンドウにアクセスするには、次の手順に従います。

1. SAS Event Stream Processing Studio でプロジェクトを開きます。ソースウィンドウを選択するか、新しいソースウィンドウを追加します。
2. 右ペインで、**入力データ(パブリッシャー)コネクタまたはサブスクライバーコネクタ**を展開します。
3. をクリックします。**コネクタ構成**ウィンドウが表示されます。**コネクタの種類**フィールドには、利用可能なコネクタの種類が表示されます。

どのコネクタを表示するかを制御する別の方法については、"コネクタを除外する" (*SAS Event Stream Processing: コネクタとアダプター*)を参照してください。

重要　**ESP サーバーのグローバル設定**ページでの変更は、アプリケーションのすべてのユーザーに影響します。代わりにユーザーセッションの設定を変更するには、**設定**ページの**ユーザーセッション**セクションで、**ESP サーバーのプロパティ**を選択します。

コネクタを含むか含まないかかを判断するプロセスには、クライアント構成サーバーの自動再起動が含まれます。

コネクタタブでは、コネクタの ESP サーバーのプロパティを変更できますが、他の ESP サーバーのプロパティは変更できません。アプリケーションのすべてのユーザーの 他の ESP サーバーのプロパティを変更するには、**プロパティ**タブを選択します。

コネクタを含む、または含まないには、次のようにします。

1. **設定**ページの**グローバル**セクションで **ESP サーバーのグローバル設定**を選択します。

   右ペインには **ESP サーバーのグローバル設定**ページが表示されます。
2. **コネクタ**タブを選択します。
3. **コネクタ構成**ウィンドウで使用したいコネクタのチェックボックスを選択し、使用したくないコネクタのチェックボックスをオフにします。
4. をクリックします。

変更を保存ウィンドウが表示されます。

5 **保存**をクリックします。

クライアント構成サーバーが再起動されます。**ステータス**フィールドに次のメッセージが表示されると、変更は完了です。 クライアント構成サーバーが実行されています。

また、コネクタの ESP サーバーのプロパティをすべてデフォルトに 戻すこともできます。

1 をクリックします。

コネクタの ESP サーバープロパティのリセット ウィンドウが表示されます。

2 **リセット**をクリックします。

クライアント構成サーバーが自動的に再起動されます。

すべてのユーザーの 環境変数および ESP サーバーのプロパティを指定する

すべてのユーザーの 環境変数と ESP サーバーのプロパティを追加、編集、または削除するには、**ESP サーバーのグローバル設定**ページの**環境変数**タブと**プロパティ** タブを使用します。変更はすぐに有効になり、Kubernetes クラスターで作成されたすべての新しい ESP サーバーは最新の設定を使用します。変更をクライアント構成サーバーに適用する場合、クライアント構成サーバーを再起動する必要があります。

注: **ESP サーバーのグローバル設定**ページにアクセスできるのは管理者グループのメンバーのみです。デフォルトでは、管理者グループは SASAdministrators です。

重要 **ESP サーバーのグローバル設定**ページでの変更は、アプリケーションのすべてのユーザーに影響します。代わりにユーザーセッションの設定を変更するには、**設定**ページの**ユーザーセッション**セクションで、**環境変数**または **ESP サーバーのプロパティ**を選択します。

すべてのユーザーの 環境変数および ESP サーバーのプロパティを指定するには:

1 **設定**ページの**グローバル**セクションで **ESP サーバーのグローバル設定**を選択します。

右ペインには **ESP サーバーのグローバル設定**ページが表示されます。

2 **環境変数**タブまたは**プロパティ** タブを選択します。

3 テーブルで、必要に応じて設定を指定します。

- 設定を追加するには、をクリックします。ウィンドウが表示され、設定名と設定値を指定するよう求めます。
- 設定を編集するには、設定の**値**フィールドをクリックし、新しい値を指定します。をクリックします。変更を保存するかどうかを確認するウィンドウが表示されます。
- 設定を削除するには、設定を含む行をクリックしてから をクリックします。ウィンドウが表示され、削除の確認を求めるメッセージが表示されます。

変更はすぐに有効になり、Kubernetes クラスターで作成されたすべての新しい ESP サーバーは最新の設定を使用します。

4 変更をクライアント構成サーバーに適用する場合、をクリックします。

ステータスフィールドには次のメッセージが表示されます。 クライアント構成サーバーの再起動中です。

ステータスフィールドに次のメッセージが表示されると、再起動が完了します。クライアント構成サーバーが実行されています。

リモート Git サーバーへの接続を指定する

リモート Git サーバー接続ページでは、リモート Git サーバーへの接続を追加、更新、および削除できます。リモート Git サーバーを定義した後、SAS Event Stream Processing 内部の Git リポジトリからリモート Git リポジトリにプロジェクトデータを同期できます。詳細については、“ファイルベースのバージョン管理”を参照してください。

注: 管理者グループのメンバーのみが**リモート Git サーバー接続**ページにアクセスできます。デフォルトでは、管理者グループは SASAdministrators です。

注意
リモート Git サーバー接続の指定は、システム管理者が実行する必要があるタスクです。 このタスクには、Git アクセストークンの指定と、リポジトリを整理するためのトピックの指定が含まれます。 プロジェクトデータはリモート Git サーバーに保存されるため、リモート Git サーバーに対してバックアッププロセスが実行されていることを確認する必要があります。バックアップを作成していない場合、データが失われる可能性があります。 リモート Git リポジトリと対話するには SAS Event Stream Processing のみを使用してください。リモート Git サーバー上に SAS Event Stream Processing プロジェクトを保存するリポジトリと直接対話すると、データ損失が発生する可能性があります。

注: GitHub、GitLab、および Gitea などの外部サービスを SAS Event Stream Processing で使用するには、Kubernetes クラスター管理者は、クラスターから適切な IP アドレスへのアクセスがあることを確認する必要があります。

リモート Git サーバーへの接続を指定するには:

1 **設定**ページの**グローバル**セクションで**リモート Git サーバー接続**を選択します。

右ペインに、**リモート Git サーバー接続**ページが表示されます。

2

リモート Git サーバープロパティの追加ウィンドウが表示されます。

3 接続を説明する名前を入力します。この名前はユーザーインターフェイスの他の場所に表示されます。たとえば、ユーザーがプロジェクトを作成し、それをリモート リポジトリに保存することを選択すると、 **リモート Git サーバー接続の追加**ウィンドウに入力した名前が、ユーザーがプロジェクトを保存できるリモート Git サーバーの 1 つとして表示されます。

4 サーバーの種類を選択します。

- **GitHub**

- **GitHub Enterprise Cloud**: GitHub.com 上の GitHub によってホストされている GitHub Enterprise。
- **GitHub Enterprise Server**: セルフホスト型の GitHub Enterprise。
- **GitLab**
- **Gitea**
- **Generic**: GitHub、GitLab でまたは Gitea 以外の任意のタイプの Git サーバー。

> 重要　Generic の Git サーバーを使用する場合、SAS Event Stream Processing Studio は Git サーバー上にリポジトリを作成できません。
>
> SAS Event Stream Processing Studio を使用してプロジェクトを作成し、それを Generic の Git サーバーに保存することを指定するには、作成しようとしているプロジェクトの名前と一致する名前の空のリポジトリが Git サーバー上にすでに存在している必要があります。同様に、SAS Event Stream Processing Studio を使用して既存のプロジェクトのリモートリポジトリを作成するには、その既存のプロジェクトの名前と一致する名前の空のリポジトリが Git サーバー上にすでに存在している必要があります。

5 サーバーの種類として GitLab、GitHub でまたは Gitea を選択した場合は、この Git サーバー上に作成されたリポジトリを表示できるユーザーを指定します。

- GitLab を使用している場合は、可視性を **Private**、**Internal**、または **Public** に設定できます。GitLab でのこれらの設定の詳細については、 GitLab のドキュメントを参照してください。
- GitHub または Gitea を使用している場合は、可視性を **Private** または **Public** に設定できます。GitHub でのこれらの設定の詳細については、 GitHub のドキュメントを参照してください。

6 サーバー URL を入力します。サーバーの種類として GitHub または GitHub Enterprise Cloud を選択した場合は、URL が入力されます。

7 ユーザー名を入力して下さい。

8 アクセストークンの名前の代わりにアクセストークンの値を入力します。たとえば、GitLab のアクセストークンの形式は次のとおりです: **glpat-xxxxxxxxxxxxxxxxxxxx** 。(この例では、一部の文字が **x** に変更されています。)

- GitLab を使用しているなら、アクセストークンはパーソナルアクセストークンである 必要があります。アクセストークンのスコープには、api を含める必要があります GitLab でのアクセストークンの作成の詳細については、 GitLab のドキュメントを参照してください。
- GitHub を使用している場合、アクセストークンはクラシックアクセストークンである 必要があります。アクセストークンのスコープには、repo を含める必要があります GitHub でのアクセストークンの作成の詳細については、 GitHub のドキュメントを参照してください。
- Gitea を使用している場合、アクセストークンのスコープに repo および user が含まれている必要があります。GitHub でのアクセストークンの詳細については、GitHub のドキュメントを参照してください。

9 サーバーの種類が Gitea の場合、Git Large File Storage(LFS)を有効にできます。詳細については、“ファイルベースのバージョン管理と Git Large File Storage”を参照してください。Git LFS を使用するようにリモート Git サーバー接続を構成するには、次のようにします。

a Gitea 管理者に連絡して、Gitea **app.ini** ファイルから LFS_JWT_SECRET のフィールドの値を取得してください。Gitea の LFS_JWT_SECRET フィールドの詳細については、 Gitea ドキュメントを参照してください。

b **リモート Git サーバー接続の追加**ウィンドウの **Gitea LFS シークレット**フィールドに値を入力します。

> 重要　Gitea が再起動すると、LFS_JWT_SECRET フィールドの値が再生成されています。**Gitea LFS シークレット**フィールドの値を更新する必要があります。

10 サーバーの種類として GitLab、GitHub でまたは Gitea を選択した場合は、トピックの名前を入力できます。次に例を示します: **esp-project-package**.トピックは、リポジトリを整理するために使用されるラベルです。トピックの指定はオプションです。

- ここで指定したトピックは、プロジェクトがリモートリポジトリに保存されるときに使用されます。つまり、プロジェクトを保存するためにリモートリポジトリが作成されると、リポジトリにはここで指定したトピックが含まれます。
- ここで指定したトピックは、プロジェクトがリモートリポジトリからインポートされるときにも使用されます。つまり、リモート Git サーバーからプロジェクトをインポートすると、指定されたトピックを持つリポジトリのみが使用可能なリポジトリとしてリストされます。
- GitLab でのトピックの詳細については、GitLab のドキュメントを参照してください。GitHub でのトピックの詳細については、 GitHub のドキュメントを参照してください。

11 **接続のテスト**をクリックして、資格情報が機能することを確認します。

12 **OK** をクリックします。

アクセストークンの有効期限が切れたときに、新しいアクセストークンが作成された場合は、接続を編集してアクセストークンの値を更新できます。をクリックして、新しいアクセストークンを入力します。他の接続プロパティは編集できません。

接続を削除するには、をクリックします。

プロジェクトがリモート Git サーバー上のリポジトリと同期されている場合、プロジェクトを永続ボリュームから削除してストレージスペースを 解放できます。詳細については、“永続ボリュームからのプロジェクトのクリーンアップ”を参照してください。

仮想環境の指定

仮想環境ページでは、仮想環境を表示、追加、更新、削除、およびエクスポートできます。仮想環境では、Python パッケージのセットまたは Lua モジュールのセットが指定されます。ユーザーがプロジェクトを作成し、管理者が作成した仮想環境を選択すると、プロジェクトはそれらの仮想環境で指定された Python パッケージまたは Lua モジュールで実行されます。詳細については、“プロジェクトプロパティの更新”を参照してください。

注: 管理者グループのメンバーのみが**仮想環境**ページにアクセスできます。デフォルトでは、管理者グループは SASAdministrators です。

仮想環境の詳細を表示するには、**仮想環境**フィールドで選択します。たとえば、**ビルトイン環境**を選択すると、Python パッケージまたは Lua モジュールのデフォルトセットに何が含まれているかを確認できます。

- 隣接するフィールドと**パッケージ**タブまたはモジュールタブが更新され、選択した環境の詳細が表示されます。
- 仮想環境内のパッケージまたはモジュールが他のパッケージまたはモジュールに依存している場合は、ページの下部にある**依存パッケージの表示**または依存モジュールの表示リンクをクリックして、これらのパッケージまたはモジュールを表示できます。依存パッケージまたはモジュールがない場合、このリンクは表示されません。

仮想環境の指定:

1 **設定**ページの**グローバル**セクションで**仮想環境**を選択します。右ペインに**仮想環境**ページが表示されます。

2 ページの右上隅で をクリックします。

3 **仮想環境の作成**ウィンドウで、仮想環境を説明する名前を入力します。この名前は、ユーザーがプロジェクトを作成し、仮想環境を選択したときに表示されます。

Python 仮想環境と Lua 仮想環境に同じ名前を付けることはできません。たとえば、Python 仮想環境 ComputerVision を作成した場合、その名前で Lua 仮想環境を作成することはできません。ただし、ComputerVisionPython および ComputerVisionLua のような名前なら環境を作成できます。

4 仮想環境の説明を入力します。

5 (オプション)以前にエクスポートした **requirements.txt** ファイルまたは modules.txt ファイルを選択します。**requirements.txt** ファイルは、Python 仮想環境の構成を指定します。modules.txt ファイルは、Lua 仮想環境の構成を指定します。このようなファイルを選択することは、**仮想環境**ページでパッケージまたはモジュールを手動で選択することの代わりになります。

6 **OK** をクリックします。

7 **パッケージ** またはモジュールタブで、 をクリックしてパッケージまたはモジュールを追加します。

8 Python パッケージ列または Lua モジュール列に、パッケージまたはモジュールの名前を入力します。

9 (オプション)オペレーター列とバージョン列を使用して、パッケージまたはモジュールのバージョンを指定します。これらの列を使用しない場合、SAS Event Stream Processing 適切なパッケージバージョンまたはモジュールバージョンを決定し、それを仮想環境に含めます。仮想環境が作成されると、インストールバージョン列に、インストールされたパッケージバージョンまたはモジュールバージョンが表示されます。

表 5　バージョンを指定するためのオペレーター

オペレーター	説明	関連する仮想環境
==	正確なパッケージバージョンまたはモジュールバージョンを使用してください。	このオペレーターは、Python 仮想環境および Lua 仮想環境に関連します。

オペレーター	説明	関連する仮想環境
~=	適切なマイナーバージョンを使用してください。たとえば、演算子列で**~=**を選択し、インストールされたバージョン列に **1.2** と入力すると、SAS Event Stream Processing は仮想環境でバージョン 1.2.1 を使用できますが、バージョン 1.3 は使用できません。	このオペレーターは Python 仮想環境に関連します。
>=	指定された最小バージョンまたは新しいバージョンを使用してください。	このオペレーターは Python 仮想環境に関連します。
<	指定されたバージョンよりも古いバージョンを使用してください。	このオペレーターは Python 仮想環境に関連します。

10 (オプション)パッケージまたはモジュールをさらに追加します。

11 をクリックします。

ステータスフィールドには、仮想環境の構築の進行状況が表示されます。

プロセスが失敗した場合は、**ログ**タブをクリックしてエラーメッセージを表示し、適切な変更を加えます。次に、ツールバーのをクリックして、仮想環境を再構築します。

> ヒント　仮想環境を指定する簡単な方法は、をクリックし、**仮想環境**を展開し、をクリックします。**仮想環境**ウィンドウを使用して仮想環境を作成します。この方法は、**requirements.txt** ファイルまたは **modules.txt** ファイルを含むプロジェクトパッケージをインポートした際に、適切な既存の仮想環境がない場合に役立つことがあります。ただし、**仮想環境**ウィンドウでは、**設定**ページの**仮想環境**セクションの**ログ**タブで使用可能なログを表示できません。**仮想環境**ウィンドウでは、既存の仮想環境を更新することもできません。
>
> 管理者グループのメンバーだけが、**仮想環境**ウィンドウを使用して仮想環境を作成します。他のユーザーは、このウィンドウを使用して既存の仮想環境を選択できます。デフォルトでは、管理者グループは SASAdministrators です。

イメージ処理のために仮想環境に含まれる可能性のある Python ライブラリの例を次に示します。

scikit-learn
scikit-image
scipy
mahotas
pillow
opencv-python

仮想環境の更新:

1 **仮想環境**フィールドで、更新する仮想環境を選択します。

2 **パッケージ**またはモジュールタブで変更を加えます。

3 をクリックします。

ステータスフィールドには、仮想環境の更新の進行状況が表示されます。

仮想環境を削除するには、ページの右上隅の をクリックします。

Python 仮想環境の構成を **requirements.txt** ファイルにエクスポートすることも、Lua 仮想環境の構成を modules.txt ファイルにエクスポートすることもできます。仮想環境をエクスポートすることで、同じ仮想環境を再度作成することができます。たとえば、同じ仮想環境を別の SAS Event Stream Processing 配置に追加したい場合があります。

仮想環境のエクスポート:

1 **仮想環境**フィールドで、エクスポートする仮想環境を選択します。

2 をクリックし、**構成のエクスポート**をクリックします。

requirements.txt ファイルまたは modules.txt ファイルがコンピューターにエクスポートされます。

注: エクスポートファイルの場所は、ブラウザの構成によって異なる場合があります。

以前にエクスポートした **requirements.txt** ファイルを別の仮想環境にインポートするには、をクリックします。これは、**パッケージ**またはモジュールタブにあります。

- ファイルで指定されたパッケージまたはモジュール仮想環境に追加されます。
- ファイルが仮想環境にすでに含まれているパッケージを指定している場合、ファイルの詳細で既存の詳細が上書きされます。たとえば、仮想環境に Python パッケージ numpy 1.26.3 が含まれており、numpy 1.26.4 を指定する **requirements.txt** ファイルをインポートすると、numpy 1.26.4 が使用されます。

ユーザーセッションの設定

ユーザーセッションの設定の概要

設定ページの**ユーザーセッション**セクションでは、ユーザーセッションの設定を変更することができます。

Kubernetes クラスターで ESP サーバーを作成する際に使用する特定の環境変数と特定の ESP サーバープロパティ (ログプロパティを含む)の値を追加、編集、リセット、および削除できます。SAS Event Stream Processing Studio または SAS Event Stream Manager で Kubernetes クラスターにプロジェクトを配置したり、SAS Event Stream Processing Studio でプロジェクトをテストしたりすると、ESP サーバーが作成されます。

重要　これらの設定を変更した場合、現在のユーザーにのみ影響します。このアプリケーションのすべてのユーザーの設定を変更するには、**設定**ページの**グローバル**セクションで、**グローバル環境変数**または**グローバル ESP サーバーのプロパティ**を選択します。

SAS Event Stream Processing Studio および SAS Event Stream Manager の**設定**ページの**ユーザーセッション**セクションで加えた変更は、互いに独立しています。両方のアプリケーションで同じ設定を使用したい場合は、両方のアプリケーションで設定を構成する必要があります。

ユーザーセッションの環境変数および ESP サーバーのプロパティを指定する

環境変数と ESP サーバープロパティ (ログプロパティを含む)を調整するには、次の手順を実行します。

1 **設定**ページの**ユーザーセッション**セクションで、**環境変数**または **ESP サーバーのプロパティ**を選択します。

右側のペインに、**環境変数**ページまたは **ESP サーバーのプロパティ**ページが表示されます。

2 テーブルで、必要に応じて設定を指定します。

- 設定を追加するには、をクリックします。**環境変数の追加**または **ESP サーバープロパティの追加**ウィンドウが表示されます。このウィンドウを使用して設定名と設定値を指定します。
- 設定を編集するには、設定の**値**フィールドをクリックし、新しい値を指定します。をクリックします。**変更の保存**ウィンドウが表示され、変更を保存するかどうかの確認を求められます。
- 設定を削除するには、設定を含む行をクリックしてから をクリックします。**環境変数の削除**または **ESP サーバープロパティの削除**ウィンドウが表示され、削除の確認を求めるメッセージが表示されます。
- すべての設定をデフォルト値にリセットするには をクリックします。**環境変数のリセット**または **ESP サーバープロパティのリセット**ウィンドウが表示され、リセットの確認を求められます。

ここでは、デフォルトで **ESP サーバーのプロパティ**ページに存在するいくつかのプロパティに関する追加情報を示します。

- server.disableTrace – この設定の値が偽の場合、プロジェクト内の特定のウィンドウの TRACE レベルのログ記録を有効にすることができます。SAS Event Stream Processing Studio でプロジェクトを開き、をクリックし、継続クエリの設定を表示します。**デバッグ**セクションでは、TRACE レベルのログを有効にするウィンドウを選択できます。
- server.maxws-queue – このプロパティを使用して、ESP サーバーが古いイベントの削除を開始する前に、WebSocket イベントキューに保持する最大メガバイト数を指定します。通常、このプロパティの値を変更する必要はありません。ただし、イベントが破棄されるという警告が ESP サーバーログに表示される場合は、このプロパティの値を増やすか、イベントの入力速度を遅くすることを検討してください。詳細については、*“イベントキューのサイズを設定するプロパティ” (SAS Event Stream Processing: Kubernetes 環境での SAS Event Stream Processing の使用)*を参照してください。

プロジェクトを現在のユーザーとして実行するかどうかを指定する

スタンドアロンの SAS Event Stream Processing が配置されると、プロジェクトは常にシステムユーザーとして Kubernetes クラスター内で実行されます。これを変更することはできません。

SAS Event Stream Processing を他の SAS 製品と共に配置する場合、状況によっては、Kubernetes クラスターでプロジェクトを実行する方法を指定できます。

- プロジェクトがプロジェクトパッケージの一部である場合、プロジェクトは常に Kubernetes クラスター内でシステムユーザーとして実行されます。
- デフォルトでは、プロジェクト XML ファイルは、現在のユーザーのアクセス権と権限を使用して Kubernetes クラスターで実行されます。この設定を調整して、現在のユーザーに使用するグループの権限を選択したり、代わりにシステムユーザーとしてプロジェクト XML ファイルを実行したりすることができます。プロジェクトをシステムユーザーとして実行することをお勧めする状況については、*"ファイルアクセスについて" (SAS Event Stream Processing: トラブルシューティング)*を参照してください。

Kubernetes クラスターでプロジェクト XML ファイルを実行する方法を指定するには:

1 **設定**ページの**ユーザーセッション**セクションで **ESP サーバーのプロパティ** を選択します。

右側のペインに、**ESP サーバーのプロパティ** ページが表示されます。

2 希望の設定を選択します:

- **システムユーザーとしてプロジェクト XML ファイルを実行します**。デフォルトでは、システムユーザーは、SAS ユーザーアカウントです。
- **現在のユーザーとしてプロジェクト XML ファイルを実行します**。権限が使用されるグループは、現在のユーザーの LDAP レコードのグループ ID(GID) です。複数のグループが利用可能な場合は、**グループを上書き**チェックボックスを選択して、別のグループを選択できます。

注: この設定が影響するのは、ユーザーセッションだけです。それらはグローバルには適用されません。

3 をクリックします。

ESP オペレータ

概要

設定ページでは、ESP オペレータログにアクセスする別の方法を表示します。これは、ポッドが起動せず、SAS Event Stream Processing Studio のテストモードまたは SAS Event Stream Manager の特定の配置のページでログメッセージにアクセスできない場合に役立ちます。

ESP オペレータログへのアクセス

設定ページの **ESP オペレータ**セクションで **ESP オペレータログ**を選択します。

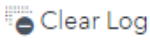
右側のペインには、ESP オペレータログからのメッセージが表示されます。

ログメッセージのフィルタリング

ログメッセージをメッセージの種類でフィルタリングするには、次のオプションを 1 つ以上選択します。

- 全部 – すべてのログメッセージの種類を表示します。
- 情報ー警告またはエラーではない一般的なログメッセージを表示します。
- 警告 - 警告メッセージのみを表示します。
- 致命的なエラーー致命的なエラーおよび標準のエラーを含むエラーメッセージのみを表示します。

ESP オペレータログページのクリア

ESP オペレータログの内容をクリアするには、 

をクリックします。

ESP オペレータログのエクスポート

ESP オペレータログをテキストファイルにエクスポートできます。 

をクリックします。ログがコンピュータにエクスポートされます。エクスポートしたログの場所は、ブラウザの構成によって異なる場合があります。

ログメッセージ内の検索

検索フィールドを使用して、ログメッセージで特定の用語を検索します。検索フィールドは、ログフィルタと連携して機能します。たとえば、**警告**をクリックしてから特定の用語を検索すると、警告メッセージ内でのみその用語が検出されます。検索では大文字と小文字は区別されません。

ツール

ツールの概要

設定ページの**ツール**セクションでは、アダプター内のパスワードの暗号化、永続ボリュームからのプロジェクトのクリーンアップ、プロジェクト XML ファイルのファイルベースのバージョン管理を使用するように変換できます。

アダプター内のパスワードを暗号化する

一部のアダプターでは、接続の構成詳細にパスワードを入力する必要があります。SAS Event Stream Processing Studio および SAS Event Stream Manager は、アダプター内のパスワードを暗号化できるようにします。

- コネクタ内のパスワードを暗号化するには、SAS Event Stream Processing Studio の**コネクタ構成**ウィンドウを使用します。詳細については、“コネクタのパスワードを暗号化する”を参照してください。
- RabbitMQ および Solace の代替トランスポートを含むアダプター内のパスワードを暗号化するには、SAS Event Stream Processing Studio または SAS Event Stream Manager の**設定**ページのパスワード暗号化ツールを使用します。

設定ページでパスワード暗号化ツールを使用する方法は次のとおりです:

1 **設定**ページの**ツール**セクションで、 **パスワードの暗号化**を選択します。

2 アダプターを選択します。

3 ユーザー ID を入力します。

4 暗号化するパスワードを入力し、**パスワードを暗号化**をクリックします。

5 **コピー**をクリックして、暗号化されたパスワードをクリップボードにコピーします。

6 データソースに接続するときに、コピーしたパスワードを使用します。

たとえば、MQTT アダプターを使用している場合は、暗号化されたパスワードを mqttpassword キーの値として使用します。アダプターによっては、他の関連キーも設定する必要がある場合があります。たとえば、MQTT アダプターの場合は、mqttpasswordencrypted キーを true に設定する必要もあります。

別の例を次に示します。データベースアダプタの場合は、暗号化されたパスワードを使用してシステムデータソース名(DSN)を作成します。

```
DSN=Oracle Wire Protocol;uid=exampleuserid;pwd=pkrrtO8hAlzcgkUUrO5zRhgZjePLR0HgqQYWtyLr5W
70yUjT3Wf1Eo72VGd4i6s3;hostname=example_DB_server_name.com;servicename=example.process_name"
```

アダプターの接続の詳細については、“ソース言語別アダプター” (*SAS Event Stream Processing: コネクタとアダプター*)を参照してください。

RabbitMQ および Solace の代替トランスポートの接続の詳細については、“Java クライアント用の代替トランスポートの使用” (*SAS Event Stream Processing: パブリッシュ/サブスクライブ API リファレンス*)を参照してください。

永続ボリュームからのプロジェクトのクリーンアップ

プロジェクトがリモート Git サーバー上のリポジトリと同期されている場合、プロジェクトを永続ボリュームから削除してストレージスペースを 解放できます。プロジェクトはリモート Git サーバーに残り、プロジェクトを再度開くと永続ボリュームに復元されます。

注: 管理者グループのメンバーのみが**永続ボリュームのクリーンアップ**ページにアクセスできます。デフォルトでは、管理者グループは SASAdministrators です。

永続ボリュームからプロジェクトを削除するには、次の操作を実行します。

1 **設定**ページの**ツール**セクションで、**永続ボリュームのクリーンアップ**を選択します。

このページには、リモート Git サーバー上のリポジトリと同期されており、実行されていないプロジェクトのみがリストされます。

2 削除するプロジェクトを選択し、を選択します。

3 **永続ボリュームのクリーンアップ**ウィンドウで、**はい**をクリックします。

プロジェクトをバルクでファイルベースのバージョン管理に変換

プロジェクトをファイルベースのバージョン管理に変換ページでは、管理者がすべてのプロジェクト XML ファイルをバルク変換して、ファイルベースのバージョン管理を使用できます。詳細については、“ファイルベースのバージョン管理を使用してプロジェクトバージョンをパブリッシュする”を参照してください。

注: 管理者グループのメンバーのみが、**プロジェクトをファイルベースのバージョン管理に変換**ページにアクセスできます。デフォルトでは、管理者グループは SASAdministrators です。

プロジェクト XML ファイルごとにプロジェクトパッケージが作成されます。詳細については、“プロジェクトパッケージ”を参照してください。

SAS Event Stream Processing が他の SAS 製品とともに配置されている場合、SAS Viya プラットフォームサービス(つまり、ファイルサービスとファイルストアサービス)を使用して以前にパブリッシュされたプロジェクト XML ファイルのバージョン履歴が保持されます。また、一般ユーザーがプロジェクトを一度に 1 つずつ手動で変換することも可能です。詳細については、“ファイルベースのバージョン管理を使用するようにプロジェクトを変換する”を参照してください。

プロジェクトをバルクで変換するには、次のようにします:

1 **ツール**セクションのの**設定**ページで、**プロジェクトをファイルベースのバージョン管理に変換**を選択します。

ステータス列のアイコンは、変換する準備ができているプロジェクトを示します。

2 **変換**をクリックして、変換を開始します。

3 変換プロセスのステータスを監視します:

- プロジェクトのステータスがに変更された場合 、その変換は完了です。

> ヒント をクリックし、変換が完了したプロジェクトをテーブルから削除し、変換が完了していないプロジェクトに集中できるようにします。

- プロジェクトのステータスが⊗と表示されている場合、ステータス アイコンにマウス カーソルを合わせてエラーメッセージを表示し、問題に対処します。たとえば、プロジェクトが SAS Event Stream Processing Studio で実行中のため、プロジェクトを変換できなかった場合、プロジェクトを停止してプロジェクトの変換を再度試行できます。SAS Event Stream Manager で実行中のプロジェクトを変換できます。

4 SAS Event Stream Processing が他の SAS 製品とともに配置される場合、元のプロジェクト XML ファイルのパブリッシュされたバージョンは、ファイルサービスおよびファイルストアサービスにまだ存在します。をクリックして削除します。

変換された各プロジェクトにプロジェクトパッケージが表示されます。プロジェクトにファイルへの参照が含まれている場合(たとえば、テストデータを含むファイル)、それらのファイルをプロジェクトパッケージに追加し、ファイルの新しい場所を指すように参照を調整することを検討してください。詳細については、“プロジェクトパッケージを管理する”を参照してください。

ファイルのアップロードおよびインポートのセキュリティに関する考慮事項

ファイルを SAS Event Stream Processing Studio にアップロードまたはインポートする場合、ファイルの内容が有害でないことを確認する必要があります。たとえば、プロジェクトをインポートしたり、ファイルをプロジェクトパッケージにアップロードしたりすることができます。SAS Event Stream Processing Studio は、アップロードまたはインポートされたファイルに有害なコンテンツがないか確認しません。

パスの操作を防ぐために、SAS Event Stream Processing Studio はアップロードまたはインポートされたファイルのファイル名に含めることができる文字を制限します。これらの制限は、ZIP ファイル内のファイル名にも適用されます。