



# SAS<sup>®</sup> Event Stream Processing: 概要

## 2020.1 - 2020.1.4

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 とは

---

### 製品の概要

SAS Event Stream Processing を使用すると、多数のイベントの連続フローを迅速に処理および分析することができます。それは、様々な SAS ソリューションと連携し、製造業、ヘルスケア、エネルギー、公益事業、金融サービス、サイバーセキュリティ、不正検知などのリアルタイムストリーミングイベント分析を提供します。

イベントは、イベントストリームと呼ばれる高スループット、待ち時間が少ないデータフローを通じて配信されます。イベントストリームは、次を使用するアプリケーションでパブリッシュされます。

- コネクタクラスまたはアダプタ実行可能ファイル
- C、JAVA、または Python パブリッシュ/サブスクライブ API
- SAS Event Stream Processing Streamviewer
- SAS Event Stream Processing Studio

Jupyter Lab では、Python のオープンソースパッケージである ESPPy を使用して、Event Stream Processing モデルをプログラミングし、ESP サーバーに接続できます。

Event Stream Processing アプリケーションは、イベントストリームに対してリアルタイムな分析を実行できます。Event Stream Processing の一般的な使用例には、次のものが含まれますが、これらに限定されません。

- センサーデータの監視と管理
- 運用システムの状態監視と管理
- オブジェクトの検出と分類
- サイバーセキュリティ分析
- 資本市場取引システム
- 不正検知と防止
- パーソナライズドマーケティング

次の表では、製品の 2 つのバリエーションについて説明します。

表 1 製品バリエーション

製品バリエーション	説明
SAS Event Stream Processing	このバリエーションは、ESP サーバー、コマンドラインクライアント、および次のコンポーネントを含むポートフォリオを提供します。
コンポーネント	コンポーネントの説明
SAS Event Stream Processing Analytics	<p>高度なアナリティクスと機械学習のテクニックで、イベントストリーム処理のプロジェクトを管理することができます。</p> <p>詳細は、<a href="#">SAS Event Stream Processing: ストリーミング分析の適用</a>を参照してください。</p>
SAS Event Stream Processing Studio	<p>Web ベースのクライアントで、これを使用すると、Event Stream Processing プロジェクトの作成、編集、アップロード、テストを行うことができます。</p> <p>詳細は、<a href="#">SAS Event Stream Processing: SAS Event Stream Processing Studio の使用</a>を参照してください。</p>
SAS Event Stream Processing Streamviewer	<p>Web ベースのダッシュボードツールで、これを使用すると、モデルを流れるイベントを視覚化できます。</p> <p>詳細は、<a href="#">SAS Event Stream Processing: SAS Event Stream Processing Streamviewer の使用</a>を参照してください。</p>
SAS Event Stream Manager	<p>SAS Event Stream Processing 環境を管理できるようにする Web ベースのクライアントです。</p> <p>詳細については、<a href="#">"SAS Event Stream Manager について"</a>を参照してください。</p>
	<p>このバリエーションを使用するには、製品コンポーネントの構築済みの Docker イメージを取得し、Kubernetes クラスターに配置します。2つの方法のいずれかの方法で配置することができます。</p>
	<ul style="list-style-type: none"> <li>■ 他の SAS 製品と一緒に。詳細は、<a href="#">SAS Vija: 配置ガイド</a>を参照してください。</li> <li>■ <b>LightWeight 版</b>の製品として。</li> </ul>
SAS Event Stream Processing(エッジコンピューティング用)	<p>このバリエーションは、コンポーネントを小規模なエッジシステムに簡単に配置するための構成を提供します。このバリエーションをオンプレミスに配置することができます。あるいは、2つの Docker イメージのどちらかをインストールすることができます。</p> <ul style="list-style-type: none"> <li>■ 1つの Docker イメージで基本的なイベントストリーム処理機能を提供します。</li> <li>■ もう一つの Docker イメージには、SAS Event Stream Processing でパッケージ化された分析アルゴリズムが含まれており、分析ストアファイルに含まれるオフラインモデルの実行にはライブラリと拡張機能が必要です。</li> </ul> <p>エッジコンピューティング用の SAS Event Stream Processing の詳細については、<a href="#">SAS Event Stream Processing for Edge Computing: 配置ガイド</a>を参照してください。</p>

## Event Stream Processing アプリケーションの作成

概念的には、イベントは、フィールドの集合として記録できる、特定可能な時間に発生するものです。Event Stream Processing アプリケーションを作成する際に、次の質問に教えてください。

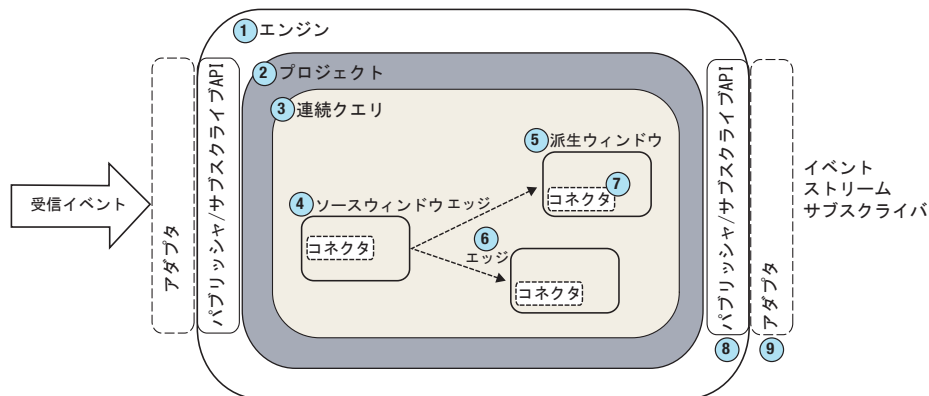
- アプリケーションにはどのようなイベントストリームが、どのようなプロトコルやフォーマットでパブリッシュされますか。
- データはどうなりますか？ イベントストリームはどのように変換され、分析されますか？
- 結果として得られるイベントストリームは何ですか？ どのようなアプリケーションがこれらのイベントストリームをどのフォーマットとプロトコルでサブスクライブしていますか。

これらの質問に回答することにより、イベントストリーム処理モデルの構造を作成できます。

## Event Stream Processing モデルとは

Event Stream Processing モデルは、パブリッシャからの入力イベントストリームを、サブスクライバが結果的に消費する、意味のあるイベントストリームに変換し分析する方法を指定します。次の図は、モデル階層を示しています。

図 1 Event Stream Processing のモデル階層



- 1 モデル階層の最上位にエンジンがあります。ESP サーバーはエンジンインスタンスです。
- 2 エンジンには、それぞれ一意の名前が付けられた 1 つ以上のプロジェクトが含まれています。プロジェクトは、サイズがプロジェクト属性として定義されている専用スレッドプールで実行されます。プロジェクトでスレッドの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 プロジェクトは 1 つ以上の連続クエリを含むテナです。連続クエリは有向グラフで表されます。このグラフは、1 つまたは複数のパラレルパスの方向に続く結合ノードのセットです。連続クエリは、データ変換であり、受信イベントストリームの分析であるデータフローです。
- 4 各クエリには固有の名前が付けられ、1 つ以上のソースウィンドウから始まります。
- 5 ソースウィンドウは、通常、1 つ以上の派生ウィンドウに接続されます。派生ウィンドウは、データ内のパターンを検出したり、データを変換したり、データを集計したり、データを分析したり、データに基づいて計算を実行したりすることができます。それらは他の派生ウィンドウに接続することができます。

- 6 Windows はエッジによって接続されており、それは関連する方向を持っています。このコンテキストでは、エッジは2つ以上のウィンドウ間の接続を指定するプログラム要素です。
- 7 コネクタは、イベントストリームをエンジンとの間でパブリッシュまたはサブスクライブします。コネクタはエンジンに対してインプロセスになっています。
- 8 パブリッシュ/サブスクライブAPIを使用して、同じマシンまたはネットワーク上の別のマシンからイベントストリームウィンドウにサブスクライブすることができます。同様に、パブリッシュ/サブスクライブ API を使用して、実行中のイベントストリームプロセッサプロジェクトのソースウィンドウにイベントストリームをパブリッシュすることができます。
- 9 アダプタは、ネットワーク接続可能なスタンドアロンの実行可能プログラムです。アダプタは、パブリッシュ/サブスクライブ API を使用して、次を行うイベントストリームをパブリッシュします。
  - ソースウィンドウにイベントストリームをパブリッシュします。
  - 任意のウィンドウからイベントストリームをサブスクライブします。

モデリング層のいくつかのオブジェクトは、時間間隔をマイクロ秒単位で測定します。次の間隔はミリ秒単位で測定されます。

- パターンのタイムアウト期間
- 時間ベースの保持における保持期間
- 周期的なウィンドウ出力のためのパルス間隔

ほとんどの非リアルタイムなオペレーティングシステムは、約 10 ミリ秒の割り込み粒度を持っています。したがって、10 ミリ秒より短い時間間隔を指定すると、予測できない結果につながる可能性があります。

---

注: 実際には、これらの間隔の最小値は 100 ミリ秒でなければなりません。値を大きくするほど予測可能な結果が得られます。

---

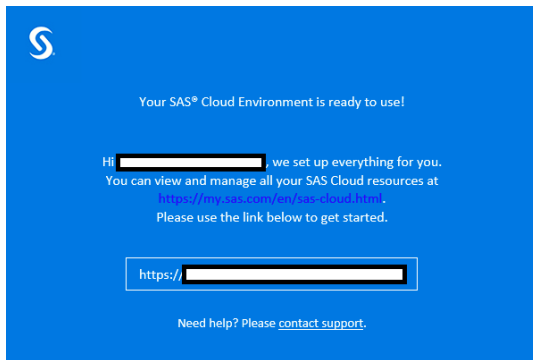
## SAS Event Stream Processing on SAS Cloud の使用

### 概要

SAS Cloud は、コンテナ技術を使用して、SAS ソフトウェアをサービスとして提供します。ソフトウェアまたは関連するインフラストラクチャをインストール、更新、または保守することなく、SAS ソフトウェアの機能を試すことができます。

### 無料トライアルにアクセスする

- 1 次を参照してください。 [www.sas.com/esp](http://www.sas.com/esp) ページ右上の **Get Free Trial** をクリックします。
- 2 ユーザープロファイルを設定しエンドユーザーライセンス契約(EULA)を確認して同意します。その後、受信トレイで追加の手順を確認するよう通知されます。
- 3 リンクを含むメールメッセージからトライアル環境にアクセスします。

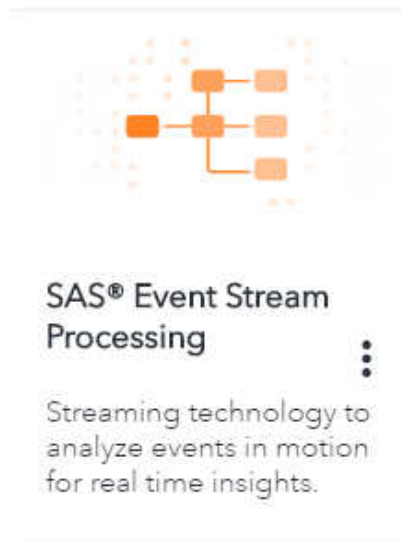


注: 最初のトライアルは 30 日間続きます。その後、さらに 30 日間トライアルを延長できます。

## SAS Event Stream Processing on SAS Cloud を開く

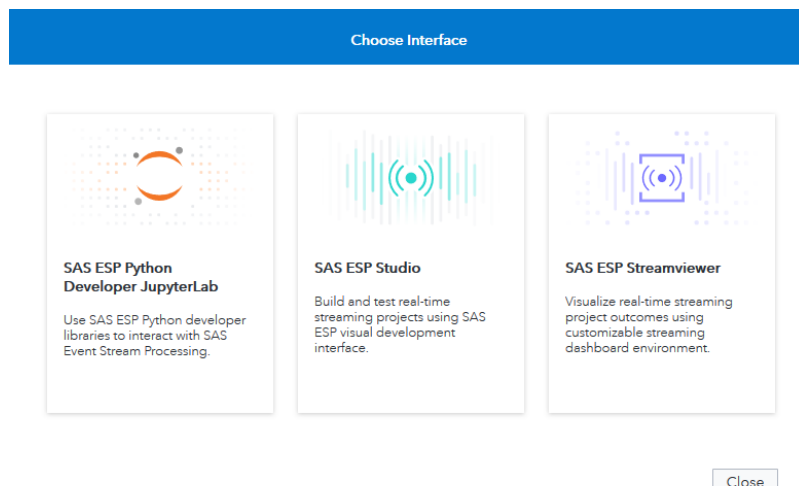
- 1 環境にサインインした後、製品タイルをクリックして、SAS Event Stream Processing on SAS Cloud を開きます。

図 2 製品タイル



- 2 無料トライアル期間が終了する前にファイルをダウンロードしてコンピュータに保存するには、次の 3 つのインターフェイスから 1 つ以上を選択します。

図 3 利用可能なインターフェイス



- 3 つのインターフェイスの操作に関する詳細については、次のトピックにアクセスしてください。
  - [SAS ESP Python Developer JupyterLab](#)
  - [SAS ESP Studio](#)
  - [SAS ESP Streamviewer](#)

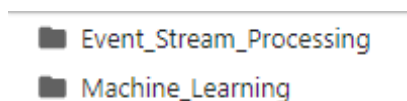
## SAS Event Stream Processing Developer JupyterLab での作業

- 1 このインターフェイスをクリックして、Web ベースのユーザーインターフェイスである JupyterLab で作業します。そこで、すぐ使える Jupyter Notebooks にアクセスして Python コードを実行できます。これにより、データサイエンティストが SAS Event Stream Processing プロジェクトを設計、テスト、ESP サーバーに配置するための Python のオープンソースパッケージである ESPPy を操作できます。

注: JupyterLab に精通していると、このインターフェイスを使用したエクスペリエンスが向上しますが、必須ではありません。

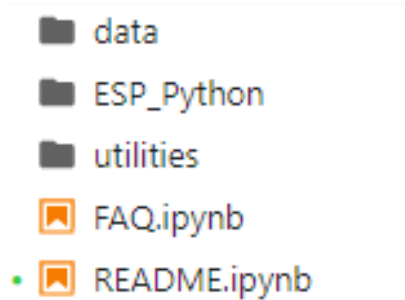
- 2 右側に表示される詳細な README ファイルをお読みください。このファイルは、環境に関する詳細情報と追加リソースのリストを提供します。
- 3 左側の **Event\_Stream\_Processing** フォルダをダブルクリックします。

図 4 フォルダの初期セット



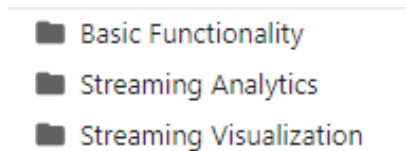
- 4 表示されるフォルダのセットから、**ESP\_Python** をダブルクリックします。

図 5 Event Stream Processing フォルダ



- 5 3つのフォルダが表示されます。利用可能な Jupyter Notebooks にアクセスするには、これらのフォルダのいずれかをダブルクリックします。

図 6 Jupyter Notebooks フォルダ



- 6 ノートブックをダブルクリックして開きます。

表 2 利用可能な Jupyter Notebooks

フォルダ	ノートブック	説明
基本機能	MAS ストア - ストアの管理	モジュールを作成してそのコンテンツを操作するために、SAS Micro Analytic Service(MAS)ストアと対話します。
	MAS ストア - ESP でのモジュールの使用	SAS Micro Analytic Service(MAS)オブジェクトを作成し、それを計算ウィンドウのモジュールとして使用します。
	プロジェクト定義(ファイル)	CSV ファイルからの SAS Event Stream Processing ストリーミングデータの基本機能を示します。
	プロジェクト定義(MQTT)	MQTT ブローカーからの SAS Event Stream Processing ストリーミングデータの基本機能を示します。



フォルダ	ノートブック	説明
ストリーミング分析	異常検出	SAS Event Stream Processing を使用して、SAS キャンパスの駐車投光器のエネルギー消費を監視する方法を示します。
	ジオフェンス	パリの街路を横切る"車"がジオフェンスに進入したことを検出します。
	画像分類 - モデル生成	機械学習 DLPy を使用して、新しい画像認識モデルを学習させます。次に、ESPPy を使用して、そのモデルで新しい画像をスコアリングします。
	画像分類 - モデルインференス	分析ストア(ASTORE)ファイルに含まれているモデルを使用して、データの画像処理とオンラインスコアリングを実行します。
	感情分析	非構造化データを使用したストリーミング分析を使用して、製品レビューの感情分析を実行します。
ストリーミングの可視化	時系列ラグモニタリング	時系列データのラグモニタリングを実行します(計算ウィンドウを使用したオンラインデータ変換)。
	ジオフェンス	パリの街路を横切る"車"がジオフェンスに進入したことを検出します。
	リアルタイム気象監視	さまざまな米国の都市のリアルタイムの気象データを表示する一連のウィジェットを作成します。

- 7 ノートブックは、Python コードを含むセルのセットで構成されています。コードを実行するには、セルをクリックして強調表示します。

図 7 Jupyter Notebooks のセル

```
[ ]: import esppy
from esppy.espapi.visuals import Visuals
from inspect import getsource

import time
import datetime

import ipywidgets as widgets

from os import path
homedir = path.expanduser("~/")
```


- 8 ノートブックの上部にあるツールバーの  をクリックして、コードを実行します。実行に成功すると、セルの左側に数字が表示されます。

図 8 正常に実行されたセル

```
[1]: import esppy
from esppy.espapi.visuals import Visuals
from inspect import getsource

import time
import datetime

import ipywidgets as widgets

from os import path
homedir = path.expanduser("~/")
```

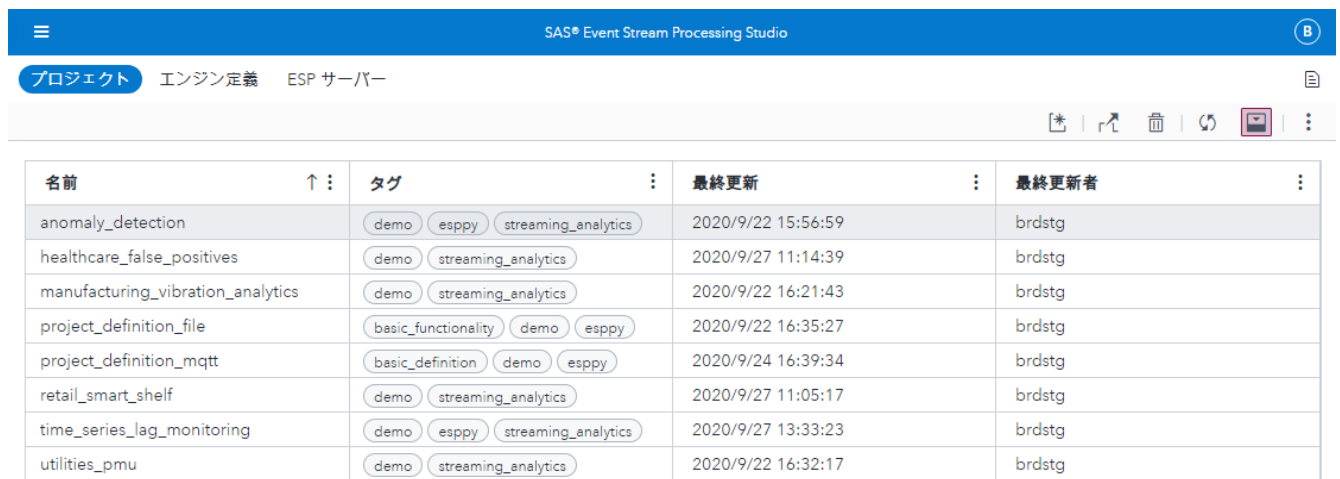
注: これらの Jupyter Notebooks は、随時更新される可能性のあるオープンソースライブラリを利用しています。トライアル期間中のある時点で、更新を取得するために、これらのライブラリをインポートするセルを再実行しなければならない場合があります。

ESPPy の詳細については、[SAS Event Stream Processing: Python インターフェイスの使用](#)を参照してください。

## SAS Event Stream Processing Studio での作業

- 1 このインターフェイスをクリックして、プロジェクトを開いてテストできる使いやすい Web ベースのクライアントである SAS Event Stream Processing Studio を操作します。ユーザーインターフェイスは、最初に、調査する既製のプロジェクトのリストを表示します。

図 9 ESP Studio の初期インターフェイス



名前	タグ	最終更新	最終更新者
anomaly_detection	demo esppy streaming_analytics	2020/9/22 15:56:59	brdstg
healthcare_false_positives	demo streaming_analytics	2020/9/27 11:14:39	brdstg
manufacturing_vibration_analytics	demo streaming_analytics	2020/9/22 16:21:43	brdstg
project_definition_file	basic_functionality demo esppy	2020/9/22 16:35:27	brdstg
project_definition_mqtt	basic_definition demo esppy	2020/9/24 16:39:34	brdstg
retail_smart_shelf	demo streaming_analytics	2020/9/27 11:05:17	brdstg
time_series_lag_monitoring	demo esppy streaming_analytics	2020/9/27 13:33:23	brdstg
utilities_pmu	demo streaming_analytics	2020/9/22 16:32:17	brdstg

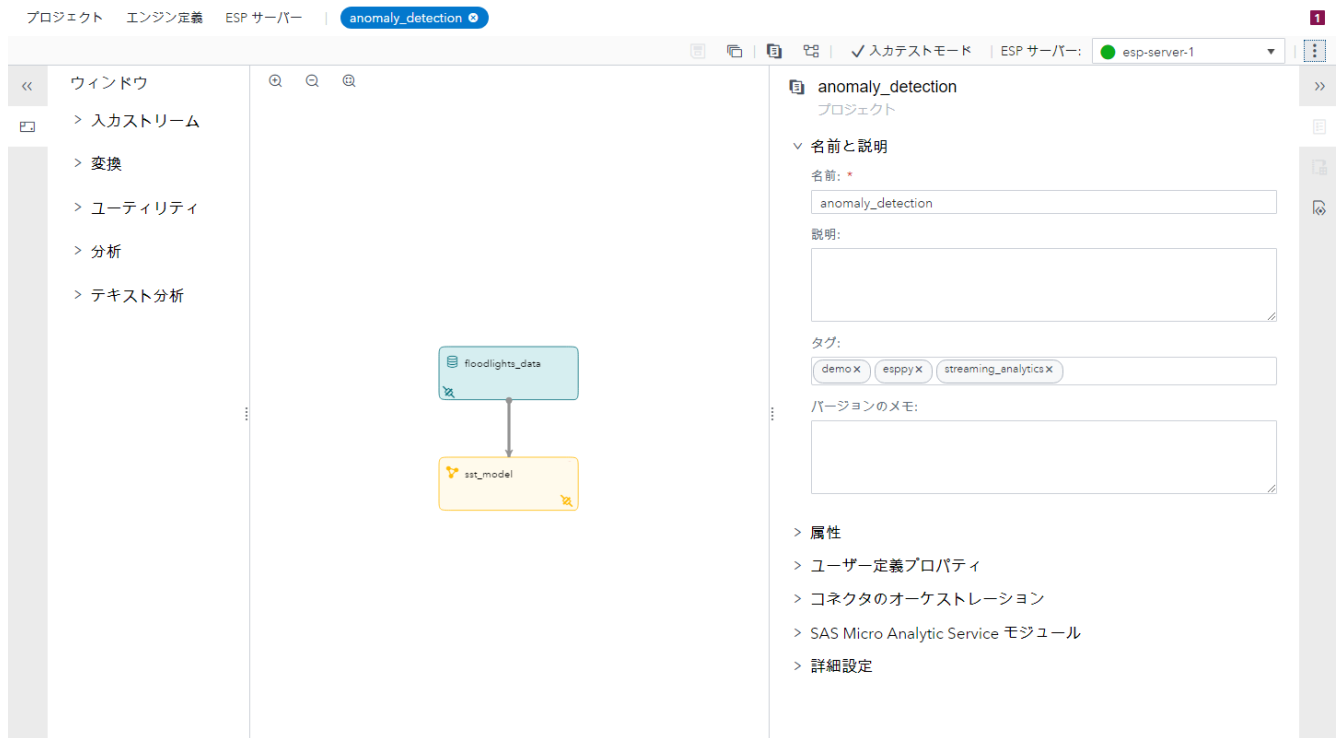
表 3 ESP Studio インターフェイスを介して利用可能なプロジェクト

プロジェクト	説明
anomaly_detection	異常検出は、残りのデータのパターンに対して異常と見なされるデータポイントを識別するために使用されます。このプロジェクトは、SAS Event Stream Processing を使用して、SAS キャンパスの駐車投光器のエネルギー消費を監視する方法を示します。
healthcare_false_positives	ヘルスケアでは、患者のステータスに関連するアラームが過剰であるため、誤検知(存在しない場合の条件の肯定フラグ)の問題が蔓延しています。アラートが意味をなさない場合、医師や看護師は患者を無視する可能性があります。アラートの数を最小限に抑えるために、このプロジェクトでは、患者のさまざまなバイタルサインの読み取り値と病歴が考慮されるパターンを定義します。しきい値は短期間の複数の読み取り値に対して定義され、しきい値を超えない限りアラームはトリガーされません。

プロジェクト	説明
manufacturing_vibration_analytics	機器の老朽化は製造においては不確定な問題であり、大きな損傷がある場合は修理に費用がかかる可能性があります。このプロジェクトでは、予防保守の必要性を予測するために、製造機器の振動イベントが監視されます。振動集団がしきい値を超えているかどうかを判断するために、各イベントにスコアを付けます。しきい値を超えている場合、異常として分類されます。期間内に十分な異常が発生した場合、SAS Event Stream Processing は、モーターの速度を下げることで機器を保護するように機能します。
project_definition_file	このプロジェクトは、CSV ファイルからの SAS Event Stream Processing ストリーミングデータの基本機能を示します。
project_definition_mqtt	このプロジェクトは、MQTT ブローカーからの SAS Event Stream Processing ストリーミングデータの基本機能を示します。
retail_smart_shelf	在庫と利益を最大化するために、最小限の待ち時間で顧客が何を購入しているかを感知することで、需要を満たすための供給を提供するより効果的な方法が保証されます。このプロジェクトでは、水筒を含む 4 種類の棚の重量センサーの読み取り値に基づいて、棚の占有率のシミュレーションデータを使用します。"ほぼ在庫なし"、"間違った商品"、"在庫不足"、"在庫なし"、"棚がすぐに空になる"という状態を知らせるアラートを提供します。
time_series_lag_monitoring	時系列ラグモニタリングでは、ターゲット時系列と 1 つ以上の追加時系列との間の相互相関を計算するアルゴリズムです。このプロジェクトは、計算ウィンドウを使用して時系列データのラグモニタリングを実行します。
utilities_pmu	<p>Phasor Measurement Units (PMU)は、以前のシステムが提供していたよりも高速かつ忠実に送電網の測定を行います。従来のシステムでは 3~4 秒ごとに読み取りを行っていましたが、1 秒あたり 30 回の測定レートで電力周波数、電圧、電流、フェーザー角度を測定します。</p> <p>このプロジェクトは、グリッドを安定させることを目的とし、送電網に影響を与えるイベントを検出して理解するプロセスを示しています。分析フローには次の 3 つの領域があります。</p> <ul style="list-style-type: none"> <li>■ イベント検出: "何かが起こりましたか"</li> <li>■ イベント識別: "何が起きましたか"</li> <li>■ イベントの資格: "どのくらい悪かったのですか"</li> </ul>

- 2 プロジェクトをダブルクリックして SAS Event Stream Processing Studio Modeler で開き、モデルをデータフロー図として表示します。この図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できます。

図 10 異常検出プロジェクト



- 3 **ESP サーバー**をクリックして、インターフェイスに関連付けられている ESP サーバーにアクセスします。ESP トライアルサーバーが実行中(ステータスは緑)で、デフォルトに設定されていることに注意してください。

図 11 ESP トライアルサーバー

デフォルト	ステータス	サーバー	タイプ	タグ	ホスト	HTTP ポート
○	●	ESP Trial Server	ESP サーバー		localhost	5002

- 4 パネルの下部で、**energy\_solar\_farm** プロジェクトが実行されている(開始済み)ことに注意してください。

名前	タグ	ステータス
● energy_solar_farm		開始済み

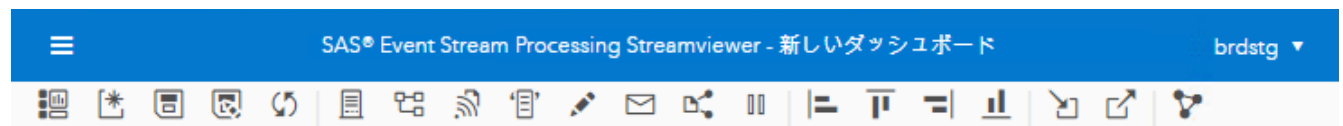
**重要** このプロジェクトをロード、アンロード、実行、または停止しないでください。これは、事前にロードされた SAS ESP Streamviewer ダッシュボードにイベントをストリーミングしています。

この環境をナビゲートし、その環境内でプロジェクトを使用する方法を示すビデオは、[SAS Event Stream Processing の学習とサポートのページ](#)にあります。SAS Event Stream Processing Studio を使用する方法の詳細については、[SAS Event Stream Processing: SAS Event Stream Processing Studio の使用](#)を参照してください。

# SAS Event Stream Processing Streamviewer での作業

- 1 イベントストリーム処理モデルを介してストリーミングするイベントを可視化できるクライアントである SAS Event Stream Processing Streamviewer を操作するには、このインターフェイスをクリックします。トライアル環境には、ユーザーが調査して使用する事前に作成されたダッシュボードが用意されています。
- 2 トライアル環境で SAS ESP Streamviewer を開くと、上部に次のツールバーが表示されます。

図 12 ESP Streamviewer ツールバー





 をクリックし、利用可能なビルド済みのダッシュボードを表示します。

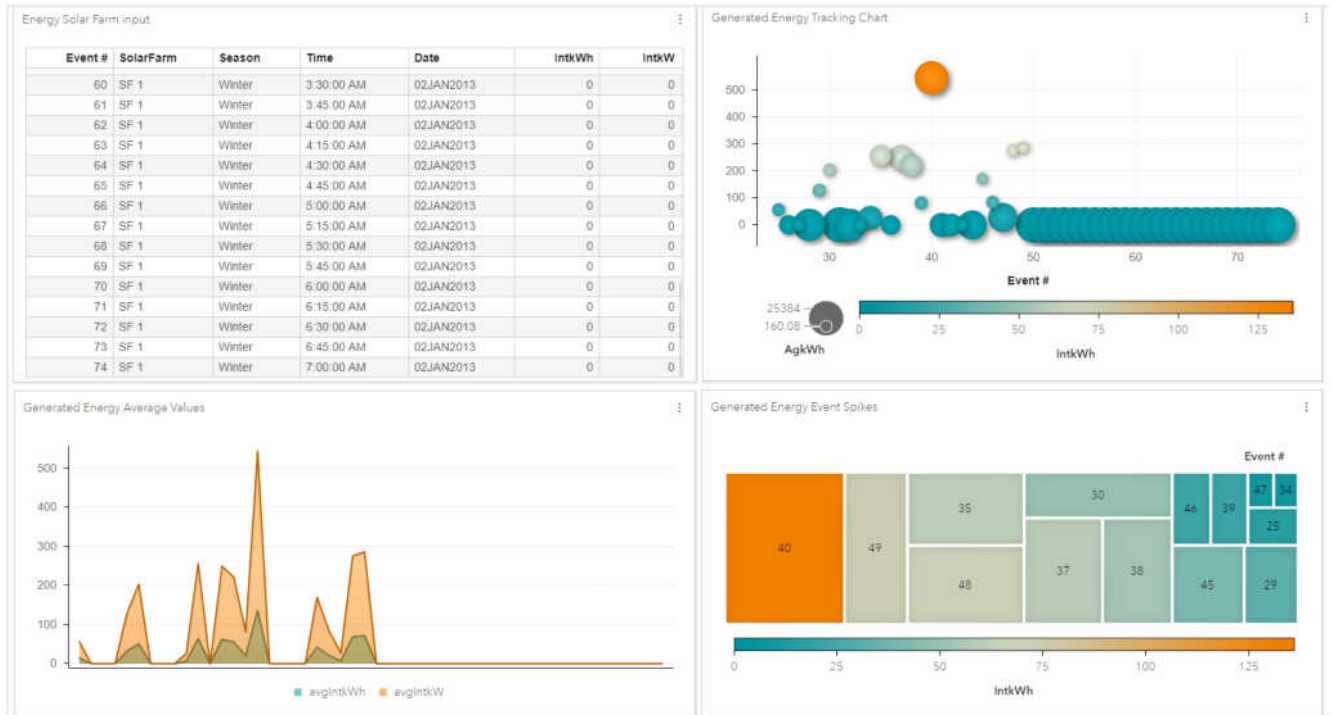
表 4 ビルド済みのダッシュボード

ダッシュボード	関連プロジェクト
エネルギー — ソーラーファーム	事前にロードされたソーラーファームプロジェクト
ユーティリティ — スマートグリッド	SAS ESP Studio インターフェイスで、まず <code>utilities_pmu</code> をテストモードで実行する必要があります。
ヘルスケア — 誤検知	SAS ESP Studio インターフェイスで、まず <code>healthcare_false_positives</code> をテストモードで実行する必要があります。
製造 — モーター振動	SAS ESP Studio インターフェイスで、まず <code>manufacturing_vibration_analytics</code> をテストモードで実行する必要があります。
ユーティリティ — スマートキャンパス	SAS ESP Studio インターフェイスで、まず <code>runanomaly_detection</code> をテストモードで実行する必要があります。

- 3 ダッシュボードの 1 つをクリックし、 をクリックしてダッシュボードを開きます。

たとえば、これはビルド済みのソーラーファームダッシュボードです。

図 13 ソーラーファームダッシュボード



イベントは関連するモデルを介してストリーミングされるため、テーブルとグラフは動的に更新されます。

この環境をナビゲートし、その環境内でダッシュボードを使用する方法を示すビデオは、[SAS Event Stream Processing の学習とサポートのページ](#)にあります。SAS Event Stream Processing Streamviewer を使用する方法の詳細については、[SAS Event Stream Processing: SAS Event Stream Processing Streamviewer の使用](#)を参照してください。

## イベントについて

### イベントとは

イベントは、イベントストリームの個々のレコードです。これは、イベントストリーム処理の基本ブロックです。イベントは、メタデータとフィールドデータで構成されます。

イベントのメタデータは、次のもので構成されます。

- オペコード(演算コード)
- (イベントが、保持ポリシー管理からの標準、部分更新、または保持生成イベントかどうかを示す) フラグのセット
- 待機時間測定に使用できる 4 つのマイクロ秒のタイムスタンプのセット

表 5 SAS Event Stream Processing でサポートされる演算コード

演算コード	説明
削除(D)	ウィンドウからイベントデータを削除します。
挿入(I)	ウィンドウにイベントデータを追加する
更新(U)	ウィンドウ内のイベントデータを変更する
アップサート(P)	キーフィールドがすでに存在する場合は、イベントデータを更新します。それ以外の場合は、イベントデータをウィンドウに追加します。
安全な削除 (SD)	イベントが存在しない場合、ERROR を生成せずにウィンドウからイベントデータを削除します。

イベントの1つまたは複数のフィールドをプライマリーキーとして指定する必要があります。キーフィールドは、演算コードのサポートを可能にします。

イベントオブジェクト内のデータは、スキーマオブジェクトに記述された通りの内部形式で格納されます。すべてのキー値は連続しており、イベントの先頭にパックされています。イベントオブジェクトは、作成されたキーに基づいて内部ハッシュ値を保持します。さらに、dfESPeventcomp namespace には、同じ基礎となるスキーマを使用して作成されたイベントを、簡単に比較するための関数があります。

パブリッシュするときに、イベントに更新または挿入の演算コードが必要かどうかかわからないときは、アップサートを使用します。イベントがインジェクトされているソースウィンドウは、挿入または更新として処理されるかどうかを決定します。ソースウィンドウは、正しいイベントと演算コードをモデル内の次の接続ウィンドウセットまたはサブスクライバに伝播します。

## イベントのデータ型

イベントは、次のデータ型をサポートします。

- INT32
- INT64
- DOUBLE
- STRING
- DATE (秒単位の粒度)
- STAMP (マイクロ秒単位までの粒度)
- MONEY (192 ビット固定小数)
- BINARY (バイナリラージオブジェクトまたは BLOB)
- RUTF8STR (参照カウント文字列または rstring)
- ARRAY (32 ビット整数、64 ビット整数、double)

基本データ型(INT32、INT64、DOUBLE、STRING、DATE、STAMP、MONEY)の場合、データはインラインでイベントに格納されます。インラインストレージは、高速なインデックス作成とシリアル化を可能にします。





表 6 イベントブロックのタイプ

イベントブロック	説明
トランザクション	プロジェクトを通じた処理はアトミックです。イベントブロック内の1つのイベントが失敗した場合(たとえば、存在しないイベントの削除など)、イベントブロック内のすべてのイベントが失敗します。失敗したイベントは記録され、オプションの不良レコードファイルに格納されます。これはさらに処理を行うことができます。
標準	プロジェクトを通じた処理はアトミックではありません。イベントは効率のためにパッケージされていますが、ソースウィンドウにインジェクトされると個別に処理されます。

一意のトランザクション ID は、変換されたイベントブロックがエンジンモデルを通過する際に伝播されます。この持続性により、イベントストリームサブスクライバは、サブスクリプションされたイベントのグループを、ID を介して特定のパブリッシュされたイベントのグループに戻すことができます。

通常、イベントブロックのサイズは、プロジェクトで使用されるコネクタまたはアダプタの `blocksize` パラメータで設定します。エンジンのパフォーマンスを最適化するには、次をお勧めします。

- 小規模イベント(>=1 および <=6 フィールド、blob を含まない)の場合、`blocksize` を 256 に設定します。
- 中規模イベント(>=7 および <=12 フィールド、blob を含まない)の場合は、32 に設定します。
- 大規模イベント(>=13 フィールド、blob を含まない)の場合は、4 に設定します。
- 巨大イベント(blob を含む)の場合は、1 に設定します。

`blocksize` を 16000 を超えて設定する大義名分はほぼありません。その値は、小規模イベントが非常に速く到着する場合のみ実用的です。イベントに画像の blob データが含まれる場合は必ず、`blocksize` を 1 に設定します。

## エンジンについて

エンジンはイベントストリーム処理モデルの最上位のコンテナです。ESP サーバーは、プロジェクトを実行するインスタンス化されたエンジンです。複数のプロジェクトを実行するか、複数の連続クエリを実行するかは、処理のニーズに応じて決まります。複数のプロジェクトを動的に導入、破棄、停止、開始することができます。異なるユースケースや ESP サーバーで異なるスレッドモデルを取得するために、複数のプロジェクトを使用できます。次のモデルを使用できます。

- より高いレベルの決定性のための単一スレッドモデル
- より高いレベルの並列性のためのマルチスレッドモデル

モジュール式のメカニズムとして連続クエリを使用できるため、実装するクエリの数は、ウィンドウがどのように区画化されているかによって異なります。連続クエリでは、必要な数のウィンドウをインスタンス化して定義できます。任意のウィンドウが1つまたは複数のウィンドウにデータを流すことができます。連続クエリでは、ループバック条件は許可されません。プロジェクトコネクタを使用して、連続クエリ全体をループバックすることができます。

イベントストリームは、次のいずれかを使用してソースウィンドウにパブリッシュまたはインジェクトする必要があります。

- パブリッシュ/サブスクライブ API

- コネクタ
- アダプタ
- HTTP クライアント
- SAS Event Stream Processing Studio
- SAS Event Stream Processing Streamviewer

連続クエリでは、使用可能なすべてのウィンドウタイプを使用してデータフローモデルを定義できません。

---

## プロジェクトについて

プロジェクトは、1つ以上の連続クエリを保持するコンテナを指定し、ユーザ定義のサイズのスレッドプールによってサポートされます。プロジェクトの増分計算の決定性のレベルは、デフォルトでは完全な並行性です。これを変更する場合は、project 要素の use-tagged-token 属性は、プロジェクトがタグ付きトークンデータフローセマンティクスを使用できるようにします。パブリッシュ/サブスクライブのスケラビリティのためのオプションのポートを指定することもできます。

データフローモデルは常に計算的に決定性です。プロジェクトがマルチスレッド化されている場合、異なるプロジェクト実行間で異なる時間に中間計算が行われる可能性があります。したがって、プロジェクトがすべての増分計算を監視する場合、インクリメンタル計算の統一が常に同じであっても、増分は実行によって変化する可能性があります。

---

**注:** 決定性レベルまたはエンジンで使用されるスレッドの数にかかわらず、各ウィンドウは常にすべてのデータを順番に処理します。したがって、ウィンドウによって受信されたデータは、決して並べ替えられず、順番以外では処理されません。

---

---

## 連続クエリについて

連続クエリは、1つ以上の有向グラフのウィンドウを保持し、ウィンドウ間の接続を指定できるコンテナを指定します。連続クエリ内のウィンドウは、データを変換または分析したり、パターンを検出したり、計算を実行することができます。クエリコンテナは、大規模プロジェクトに対して機能的なモジュール性を提供します。通常、各コンテナは単一の有向グラフを保持します。

連続クエリの処理は、次の手順に従います。

- 1 1つまたは複数のイベントを含むイベントブロック(アトミックプロパティあり/なし)がソースウィンドウにインジェクトされます。
- 2 イベントブロックは、ソースウィンドウに直接結合されている任意の派生ウィンドウに流れます。トランザクションプロパティが設定されている場合、1つ以上のイベントのイベントブロックは、結合された各派生ウィンドウへの途中でアトミックに処理されます。つまり、すべてのイベントを完全に実行する必要があります。

トランザクションプロパティを持つイベントブロック内のイベントが失敗すると、そのイベントブロック内のすべてのイベントが失敗します。失敗したイベントが記録されます。この機能を有効

にすると、レビュー、修正、および再パブリッシュするために、不正レコードファイルに書き込まれます。

- 3 派生ウィンドウは、イベントを派生ウィンドウのプロパティに基づいてゼロまたは複数の新規作成イベントに変換します。新規作成イベントが派生ウィンドウによって計算された後、イベントはモデルの更に先に流れ、新規作成イベントが潜在的に計算される場所である、接続された派生ウィンドウの次のレベルに到達します。
- 4 このプロセスは、次のいずれかが発生したときに、特定のイベントブロックのモデルのアクティブパスごとに終了します。
  - 生成されたイベントを渡すことができる、接続された派生ウィンドウはこれ以上存在しません。
  - パスに沿った派生ウィンドウが、そのイベントブロックに対して結果として発生したイベントはゼロです。したがって、結合された派生ウィンドウの次のセットには何も渡されません。

## ウィンドウについて

連続クエリには、1つ以上の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。すべてのイベントストリームをパブリッシュするかソースウィンドウにインジェクトして、連続クエリを入力する必要があります。イベントストリームは、他のウィンドウタイプにパブリッシュまたはインジェクトすることはできません。

Windows はエッジによって接続されており、それは関連する方向を持っています。

SAS Event Stream Processing では、次の派生ウィンドウタイプがサポートされています。

表7 派生ウィンドウタイプ

ウィンドウタイプ	説明
計算ウィンドウ	入力イベントストリームフィールドの計算操作を通じて入力イベントの出力イベントへの1対1変換を可能にします。
集計ウィンドウ	キー以外のフィールドが計算される点で計算ウィンドウに似ています。集計ウィンドウは、グループごとの条件のキーフィールドを使用します。すべての一意キーフィールドの組み合わせが、集計ウィンドウ内で独自のグループを形成します。同じキーの組み合わせを持つすべてのイベントは、同じグループの一部です。
計算ウィンドウ	さまざまなアルゴリズムを使用してデータイベントを変換します。
コピーウィンドウ	親ウィンドウのコピーを作成します。コピーを作成すると、新しいイベント状態保持ポリシーを設定することができます。保持ポリシーは、ソースウィンドウおよびコピーウィンドウでのみ設定できます。  コピーウィンドウのイベント状態の保持は、ウィンドウが挿入専用ではなく、ウィンドウインデックスが pi_EMPTY に設定されていない場合にのみ設定できます。その後のすべての関連ウィンドウは、保持管理の影響を受けます。イベントは、ウィンドウ保持ポリシーを超えると削除されます。
カウンタウィンドウ	モデルを通してストリーミングされているイベントの数とそれらが処理されているレートを確認できます。

ウィンドウタイプ	説明
フィルタウィンドウ	登録されたブールフィルタ関数または式を使用します。この関数または式は、フィルタウィンドウにどの入力イベントが許可されるかを決定します。
関数ウィンドウ	さまざまな種類の関数を使用してイベントデータを操作または変換できます。関数ウィンドウ内のフィールドは階層構造にすることができ、Web 分析などの応用に役立ちます。
ジオフェンスウィンドウ	ウィンドウを作成して、イベントストリームの場所が関心領域の内側にあるのか、関心領域に近いのかを判断できます。
結合ウィンドウ	2つの入力ウィンドウと結合タイプをとります。1対多、多対1、多対多の等価結合をサポートします。内部結合と外部結合の両方がサポートされています。
モデルリーダーウィンドウ	分析ストアファイルまたは非バイナリファイルの組み合わせとして SAS Event Stream Processing に取り込まれたモデルを読み取ります。
モデルスーパーバイザウィンドウ	モデルリーダーウィンドウから受信したモデルを管理します。
通知ウィンドウ	メール、ショートメッセージまたはマルチメディアメッセージを使用して通知が送信できます。任意の数の配布チャンネルを作成して通知を送信することができます。通知ウィンドウは、関数ウィンドウと同じ基本言語と関数を使用します。
オブジェクトトラッキングウィンドウ	リアルタイムでマルチオブジェクトトラッキング(MOT)を実行できます。
パターンウィンドウ	関心があるイベント(EOI)の検出を有効にします。このウィンドウタイプで定義されたパターンは、宣言された関心があるイベントを論理的に接続する式です。  パターンウィンドウを定義するには、関心があるイベントを定義し、次にこれらの関心があるイベントを演算子を使用して接続する必要があります。サポートされる演算子は、"AND"、"OR"、"FBY"、"NOT"、"NOTOCCUR"、"IS"です。演算子はオプションの一時的条件を受け入れることができます。
プロシジャウィンドウ	任意の数の入力ウィンドウや各入力ウィンドウの入力ハンドラ関数の指定(すなわち、イベントストリーム)を可能にします。
状態の削除ウィンドウ	モデルのステートフル部分からモデルのステートレス部分への移行を容易にします。
スコアウィンドウ	SAS Event Stream Processing とともにパッケージ化されたオンライン分析アルゴリズム、または <b>オフラインモデル</b> のアルゴリズムを使用したスコアイベント。
テキストカテゴリウィンドウ	受信イベントのテキストフィールドを分類できます。テキストカテゴリウィンドウは挿入専用です。テキストフィールドは、スコアを持つ0個以上のカテゴリを生成することができます。  このオブジェクトにより、SAS Contextual Analysis のライセンスを取得したユーザーは、MCO ファイルを使用してテキストカテゴリウィンドウを初期化できます。

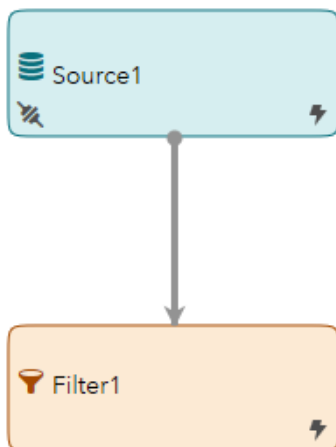
ウィンドウタイプ	説明
テキストコンテキストウィンドウ	<p>非構造化文字列フィールドから分類された用語を抽象化できます。</p> <p>このオブジェクトを使用すると、SAS Contextual Analysis のライセンスを取得したユーザーは Liti ファイルを使用してテキストコンテキストウィンドウを初期化できます。このウィンドウタイプを使用して、イベント入力からの文字列フィールドを分析し、分類された用語を検索します。その用語から生成されたイベントは、他のウィンドウタイプで分析できます。たとえば、パターンウィンドウは、テキストコンテキストウィンドウの後に、関心のあるツイートパターンを探ることができます。</p>
テキストセンチメントウィンドウ	<p>指定された受信テキストフィールド内のテキストのセンチメントとその出現確率を決定します。センチメント値は“正”、“中立”または“負”です。確率は 0 と 1 の間の値です。テキストセンチメントウィンドウは挿入専用です。</p> <p>このオブジェクトを使用すると、SAS Sentiment Analysis のライセンスを取得したユーザーは、SAM ファイルを使用してテキストセンチメントウィンドウを初期化できます。</p>
テキストトピックウィンドウ	<p>イベントで SAS Text Miner 分析を実行します。テキストトピックウィンドウは、ドキュメントから文字列フィールドとしてテキストを受け取り、処理します。テキストマイニング分析モデルは、分析ストアファイルを使用してテキストトピックウィンドウに入ります。</p>
学習ウィンドウ	<p>ストリーミングイベントデータに基づいてオンラインモデルのアルゴリズムパラメータを調整します。</p>
転置ウィンドウ	<p>イベントの行を列として、または列を行として交換できます。</p>
和結合ウィンドウ	<p>SQL 和結合(Union)操作のように、同じスキーマを持つ複数のイベントストリームを単一のストリームに結合します。</p>

## 連続クエリを介したイベントのストリーミング

### 概要

次の例は、イベントが連続クエリのウィンドウを介してストリーミングされるとどうなるかを示しています。

図 14 ソースウィンドウとフィルタウィンドウを使用した連続クエリ



このクエリでは、ソースウィンドウがイベントを単一のフィルタウィンドウにパブリッシュします。ソースウィンドウには、次のスキーマが含まれています。

ID\*: int32, symbol: string, quantity: int32, price: double

このスキーマは 4 つのフィールドで構成されます。

表 8 ソースウィンドウスキーマ

フィールド	種類
ID	32 ビット整数
シンボル	文字列
数量	32 ビット整数
価格	倍精度浮動小数点

スキーマのキーフィールドは、挿入、更新、削除、またはアップサートなどの操作のイベントを識別します。キーフィールドは一意である必要があります。イベントストリームをデータベース、キーフィールドをルックアップキーと考えることができます。

このスキーマでは、ID フィールドには、このフィールドがウィンドウのキーの一部であることを示す \*指定子があります。ID フィールドは完全にキーを形成するので、この指定子を持つ他のフィールドは、このスキーマにはありません。

ソースウィンドウの XML コードは次のとおりです。

```
<window-source name="Source1" pubsub="true">
  <schema>
    <fields>
      <field type="int32" name="ID" key="true" />
      <field type="string" name="symbol" />
      <field type="int32" name="quantity" />
      <field type="double" name="price" />
    </fields>
  </schema>
</window-source>
```

```

</schema>
<connectors>
  <connector class="fs" name="New_Connector_1">
    <properties>
      <property name="type">pub</property>
      <property name="fsname">events.csv</property>
      <property name="fstype">csv</property>
      <property name="transactional">>true</property>
      <property name="blocksize">1</property>
    </properties>
  </connector>
</connectors>
</window-source>

```

ファイルおよびソケットコネクタは、ソースウィンドウに処理されるイベントを含む CSV ファイルを現在のディレクトリにパブリッシュします。

フィルタウィンドウは `quantity > 1000` という式を適用します。したがって、イベントの数量フィールドが 1000 の値を超えた場合にのみ、イベントが渡されます。

フィルタウィンドウの XML コードは次のとおりです。

```

<window-filter name="Filter1" pubsub="true">
  <expression>quantity > 1000</expression>
</window-filter>

```

ソースウィンドウとフィルタウィンドウを接続するエッジの XML は次のとおりです。

```

<edges>
  <edge target="Filter1" source="Source1" />
</edges>

```

使用可能なフィルタ条件の詳細については、「[式の概要](#)」 (*SAS Event Stream Processing: ソースウィンドウと派生ウィンドウの使用*) を参照してください。連続クエリのコーディングに使用する XML 言語の詳細については、「[SAS Event Stream Processing: イベントストリーム処理モデルの XML 言語リファレンス](#)」を参照してください。

次のセクションでは、このモデルを介して 5 つのイベントがストリーミングされるときに何が起こるかについての詳細情報を提供します。

---

## 最初のイベントの処理

クエリを介した最初のイベントストリーミングが次のようになっているとします。

```
e1: [i,n,10,IBM,2000,164.1]
```

- 1 ソースウィンドウは、入力イベントとして e1 を受け取ります。イベントを保存し、それをフィルタウィンドウに渡します。
- 2 フィルタウィンドウは、最初のフィールドの "i" で指定されたように、入力イベントとして e1 を受け取ります。これ以降のすべてのイベントの 2 番目のフィールドは、「標準」を指定します。
- 3 **Quantity** フィールドの値は 2000 です。フィルタ式が `quantity > 1000` の場合、フィルタウィンドウは入力を格納します。通常、フィルタウィンドウは e1 を前方に渡します。ただし、フィルタウィンドウには従属ウィンドウがないため、イベントの追加データフローはありません。

ウィンドウの内容は次のようになります。

表 9 最初のイベント後のソースウィンドウの内容

ID	シンボル	数量	価格
10	IBM	2000	164.10

表 10 最初のイベント後のフィルタウィンドウの内容

ID	シンボル	数量	価格
10	IBM	2000	164.10

## 2 番目のイベントの処理

2 番目のイベントは次のとおりです。

e2: [p,n,20,MSFT,1000,114.22]

- 1 ソースウィンドウは、アップサートイベントとして e2 を受け取ります。ウィンドウにキー(ID)が 20 のストアイベントがあるかどうかをチェックします。
- 2 20 の ID は格納されていないので、ソースウィンドウは新しいイベント e2a: [I, 20, "MSFT", 1000, 114.22] を作成します。この新しいイベントを保存し、それをフィルタウィンドウに渡します。
- 3 フィルタウィンドウは e2a を入力イベントとして受け取ります。
- 4 e2 の **Quantity** フィールドの値は 1000 で、スキーマのフィルタ式で設定された条件を満たしません。したがって、このイベントは格納されず、従属ウィンドウに渡されません。

ウィンドウの内容は次のようになります。

表 11 2 番目のイベント後のソースウィンドウの内容

ID	シンボル	数量	価格
10	IBM	2000	164.10
20	MSFT	1000	114.22

表 12 2 番目のイベント後のフィルタウィンドウの内容

ID	シンボル	数量	価格
10	IBM	2000	164.10



## 3 番目のイベントの処理

3 番目のイベントは次のとおりです。

```
e3: [d,n,10,,,,]
```

注: 削除イベントの場合は、キーフィールドのみを指定する必要があります。この例では、ID フィールドのみがキーであることに注意してください。

- 1 ソースウィンドウは e3 を削除イベントとして受け取ります。
- 2 ソースウィンドウは、同じキーで格納されているイベントを検索します。削除演算コードは、ソースウィンドウからイベントを削除します。
- 3 ソースウィンドウは、「削除」演算コードが指定された状態で、検出されたレコードをフィルタウィンドウに渡します。この場合、フィルタウィンドウに渡されるレコードは次のとおりです。

```
e3a: [d,n,10,IBM,2000,164.1]
```

- 4 フィルタウィンドウは e3a を入力イベントとして受け取ります。
- 5 e3a の **Quantity** フィールドの値は 2000 になります。以前に格納されたこの古いイベントはフィルタを通過するため、削除されます。

ウィンドウの内容は次のようになります。

表 13 3 番目のイベント後のソースウィンドウの内容

ID	シンボル	数量	価格
20	MSFT	1000	114.22

フィルタウィンドウは空です。

## 4 番目のイベントの処理

4 番目のイベントは次のとおりです。

```
e4: [u,n,20,MSFT,3000,114.25]
```

- 1 ソースウィンドウは e4 を更新イベントとして受け取ります。
- 2 ソースウィンドウは、同じキーで保存されたイベントを検索し、変更します。
- 3 ソースウィンドウは、次で構成される更新ブロックを構築します。
  - 更新された値が更新ブロックとしてマークされた新しいレコード
  - 更新された古いレコード
- 4 ブロックは、削除イベントとしてマークされます。フィルタウィンドウに渡される新しいイベント更新ブロックは次のようになります。

e4a: [ub,n,20,MSFT,3000,114.22] , [d,n,20,MSFT,1000,114.25]

注: 派生ウィンドウはしばしばイベントの現在および前の状態を必要とするため、古いレコードと新しいレコードの両方が供給されます。更新プログラムによって発生した増分変更を計算するには、これらの状態が必要です。

- 5 フィルタウィンドウは e4a を入力イベントとして受け取ります。
- 6 e4a > 1000 の **Quantity** フィールドの値ですが、以前は 1000 以下でした。入力は以前のフィルタ条件を通過しませんでした。今は通過します。入力がフィルタウィンドウに存在しないため、フィルタウィンドウは次の形式の挿入イベントを生成します。

e4b: [i,n,20,MSFT,3000,114.25]

- 7 挿入イベントが格納されます。フィルタウィンドウは e4b を通過させます。ただし、従属ウィンドウがないため、この入力は通過しません。このイベントのデータフローはこれ以上ありません。

ウィンドウの内容は次のようになります。

表 14 4 番目のイベント後のソースウィンドウの内容

ID	シンボル	数量	価格
20	MSFT	3000	114.25

表 15 4 番目のイベント後のフィルタウィンドウの内容

ID	シンボル	数量	価格
20	MSFT	3000	114.25

## 5 番目のイベントの処理

5 番目のイベントは次のとおりです。

e5: [i,n,30,APPL,2000,225.06]

- 1 ソースウィンドウは e5 を入力イベントとして受け取り、格納し、e1 をフィルタウィンドウに渡します。
- 2 フィルタウィンドウは e5 を入力イベントとして受け取ります。**Quantity** フィールドの値が 1000 を超えるため、フィルターウィンドウに入力が保管されます。フィルタウィンドウには従属ウィンドウがないため、それ以上のデータフローはありません。

ウィンドウの内容は次のようになります。

表 16 5 番目のイベント後のソースウィンドウの内容

ID	シンボル	数量	価格
20	MSFT	3000	114.25
30	APPL	2000	225.06


表 17 5 番目のイベント後のフィルタウィンドウの内容

ID	シンボル	数量	価格
20	MSFT	3000	114.25
30	APPL	2000	225.06

## コード例

この例を実装する XML コードは、`xml` ディレクトリに `filter_exp.xml` として [オンライン](#) で使用できます。例には、これらの 5 つのイベントをストリーミングするデータを含む CSV ファイルが含まれています。

## イベントストリーム処理プロジェクトの実行

他の SAS Viya 製品を使用している場合は、 をクリックし、**ストリーミングプロジェクトの設計** を選択して、SAS Event Stream Processing Studio 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のリストの存在は、システムがどのようにインストールされたか、およびアクセス許可に依存します。

独立したアプリケーションとして SAS Event Stream Processing を配置した場合は、Web ブラウザを使用して次の URL を開きます。

`https://name-of-ingress-host:port/SASEventStreamProcessingStudio`

`nam-of-ingress-host:port` には、クラスタ Ingress に設定されているホスト名とポートを指定します。

**注:** SAS にサインインウィンドウは、配置でユーザーが SAS Event Stream Processing Studio にログインできるように構成されている場合にのみ表示されます。配置がこのように構成されていない場合は、アプリケーションにアクセスするためにユーザー ID とパスワードを入力する必要はありません。

SAS Event Stream Processing Studio を開いたら、次の手順に従ってください。

- 1 モデラーでプロジェクトをデザインするか、プロジェクトコードを含む XML ファイルをアップロードします。ESP Studio Moder の詳細については、[SAS Event Stream Processing: SAS Event Stream Processing Studio の使用](#)を参照してください。プロジェクトの XML コードエレメントの詳細については、[SAS Event Stream Processing: イベントストリーム処理モデルのXML 言語リファレンス](#)を参照してください。
- 2 プロジェクト内のイベントストリーム処理モデルをテストし、検証します。
- 3 SAS Viya で SAS Event Stream Processing を配置した場合は、プロジェクトをパブリッシュします。プロジェクトをパブリッシュすると、SAS Event Stream Manager に表示されます。  
独立したアプリケーションとして SAS Event Stream Processing を配置した場合は、プロジェクトをローカルのファイルシステムにダウンロードします。その後、SAS Event Stream Manager UI を使用して、ローカルファイルシステムからプロジェクトをアップロードします。
- 4 SAS Event Stream Manager を使用して、ESP サーバーにプロジェクトを配置し、SAS Event Stream Processing 環境を管理します。詳細については、["SAS Event Stream Manager について"](#)を参照してください。
- 5 次のいずれかの方法で、1 つ以上のイベントストリームをエンジンにパブリッシュします。
  - コネクタ(インプロセスクラス)を介して、コネクタの単一インスタンスは、イベントストリーム処理モデルの単一ウィンドウに対して読み取りまたは書き込みを行います。詳細は、["コネクタとアダプタについて"](#) ([SAS Event Stream Processing: コネクタとアダプタ](#))を参照してください。
  - Java、C、または Python パブリッシュ/サブスクリブ API を使用します。  
パブリッシュ/サブスクリブ API の詳細については、["パブリッシュ/サブスクリブ API の概要"](#) ([SAS Event Stream Processing: パブリッシュ/サブスクリブ API リファレンス](#))を参照してください。
- 6 コネクタ、パブリッシュ/サブスクリブ API、SAS Event Stream Processing Studio、または SAS Event Stream Processing Streamviewer を使用して、連続クエリ内の関連するウィンドウイベントストリームをサブスクリブします。  
SAS Event Stream Processing Streamviewer の詳細については、[SAS Event Stream Processing: SAS Event Stream Processing Streamviewer の使用](#)を参照してください。

---

## SAS Event Stream Manager について

---

### SAS Event Stream Manager について

SAS Event Stream Manager は、SAS Event Stream Processing 環境を管理できるようにする Web ベースのクライアントです。

SAS Event Stream Manager を使用して、次のタスクを実行できます。

- SAS Event Stream Processing プロジェクトをプロダクション環境とテスト環境に配置
- 配置、ESP サーバ、実行中のプロジェクトの健全性を監視
- 配置の管理と変更の管理

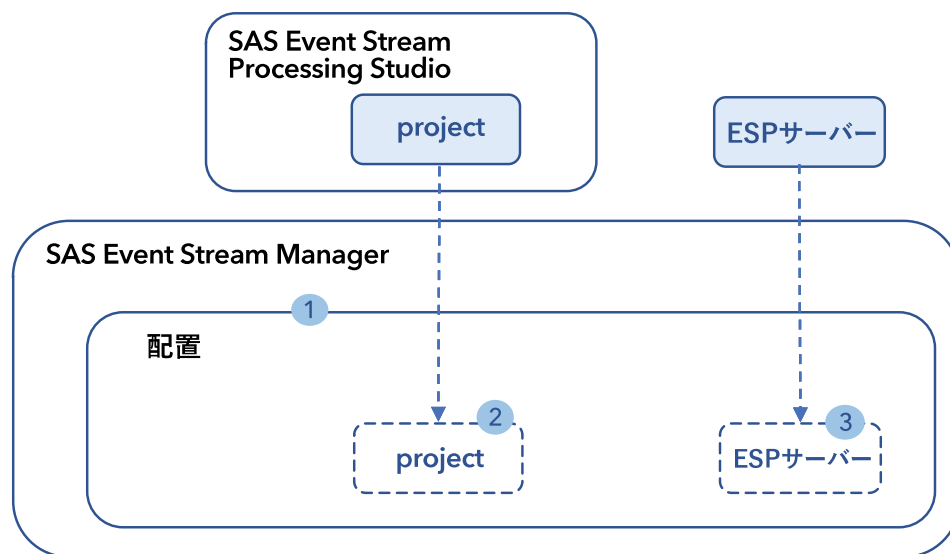
- SAS Event Stream Processing 測定サーバーを監視する

SAS Event Stream Processing Studio を使用して、SAS Event Stream Manager を使用して ESP サーバーに配置するプロジェクトを作成できます。

## SAS Event Stream Manager を使用してのプロジェクト配置の概要

SAS Event Stream Manager で実行できる主なタスクの 1 つは、SAS Event Stream Processing プロジェクトを SAS Event Stream Processing 環境に配置することです。このトピックでは、これがどのように機能するかをハイレベルで説明します。

図 15 SAS Event Stream Manager での配置



- 1 配置が作成されます。
- 2 SAS Event Stream Processing Studio で作成およびパブリッシュされたプロジェクトは、SAS Event Stream Manager に表示され配置で使用できます。
- 3 ESP サーバーが配置に追加されます。プロジェクトが kubernetes クラスタ内に配置されている場合、ESP サーバーはオンデマンドでクラスタ内に作成され、関連する配置に追加されます。

SAS Event Stream Processing 環境に存在する任意の ESP サーバーを追加できます。ESP サーバーが配置に追加された後、ESP サーバーは配置から参照されます。

プロジェクトは、SAS Event Stream Manager のユーザーインターフェイスで動的に配置するか、ジョブテンプレートを使用して配置することができます。

- SAS Event Stream Manager ユーザーインターフェイスのコントロールは、プロジェクトをすばやくロードして開始する方法を提供します。ただし、ジョブテンプレートで使用できるすべてのオプションを利用することはできません。

図 16 ジョブテンプレートなしでプロジェクトを展開するためのプロセスフロー



- ジョブテンプレートを使用すると、プロジェクトの読み込みや開始などのタスクのオプションを指定できます。たとえば、指定したフィルターに一致する ESP サーバー上のプロジェクトを停止したり、プレースホルダーの置換を実行するためにプロジェクトに変数を渡したりできます。ジョブテンプレートは、SAS Event Stream Manager 内で提供されるテキストエディターを使用して作成できます。

図 17 ジョブテンプレートを使用してプロジェクトを展開するためのプロセスフロー



注: Kubernetes クラスタで動作する ESP サーバにジョブ テンプレートを配置することはできません。

詳細は、[SAS Event Stream Processing: SAS Event Stream Manager の使用](#)を参照してください。

## プロダクションアセット

特定のアセットをプロダクションアセットとしてマークして、テスト用のアセットが誤ってプロダクションに使用される(つまり、ライブ環境で使用される)のを防ぐことができます。次のアセットは、プロダクションアセットとしてマークできます。

- 配置
- プロジェクト
- ジョブテンプレート

これらのアセットは、作成時にプロダクションアセットとしてマークできます。後でプロダクションステータスを別のステータスに変更することもできます。たとえば、プロダクション配置を非プロダクション配置に変更したり、非プロダクション配置をプロダクション配置に変更したりできます。

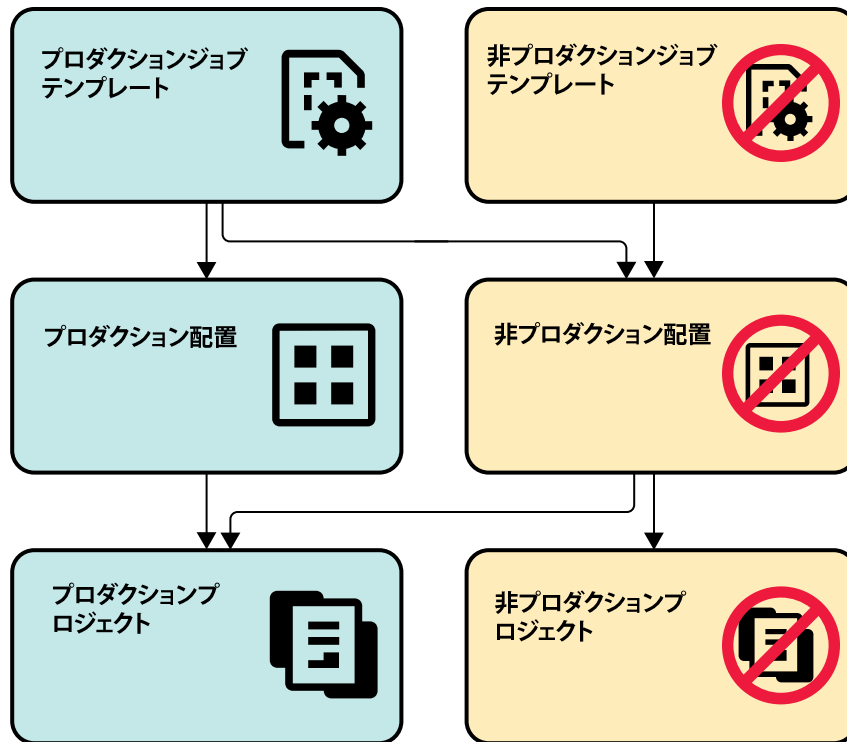
非プロダクションジョブテンプレートは、非プロダクション配置に対してのみ配置できます。このような状況では、SAS Event Stream Manager を使用して、配置するプロダクションプロジェクトまたは非プロダクションプロジェクトを選択できます。この選択により、現在のプロダクションプロジェクトをテストしたり、非プロダクションプロジェクトをテストして、将来的にプロダクションプロジェクトとしてマークするのに適しているかどうかを評価できます。

プロダクション配置に対して、または非プロダクション配置に対してプロダクションジョブテンプレートを配置できます。

- プロダクション配置に対してプロダクションジョブテンプレートを配置する場合、SAS Event Stream Manager では、配置するプロダクションプロジェクトのみを選択できます。
- 非プロダクション配置に対してプロダクションジョブテンプレートを配置する場合、SAS Event Stream Manager では、配置するプロダクションプロジェクトまたは非プロダクションプロジェクトのいずれかを選択できます。

次の図は、この情報を示しています。

図 18 ジョブテンプレートを配置するためのオプション



## 例

SAS Event Stream Processing の手順を追った例は、[SAS Event Stream Processing の製品サポートページ](#)から入手できます。

SAS Event Stream Processing コード例は[オンライン](#)で入手できます。例は、次のプログラミング言語で書かれています。

表 18 プログラミング言語別の例

プログラミング言語	サブディレクトリ
XML	xml
Python	python
Java	java

