SAS® Event Stream Processing 5.2: コネクタとアダプタ

コネクタとアダプタの使用

概要

Event Stream Processing エンジンウィンドウとの間でデータをストリーミングするには、コネクタまたはアダプタを使用できます。コネクタとアダプタは、パブリッシュ/サブスクライブ API を使用して、さまざまな通信ファブリック、ドライバ、およびクライアントとインターフェースします。コネクタは、イベントストリーム処理エンジンと同じプロセススペースでインスタンス化される C ++クラスです。XML モデルまたは C ++モデル内のコネクタを使用できます。アダプタは、通常、コネクタクラスから構築されたスタンドアロンの実行可能ファイルです。したがって、一部のアダプタは、対応するコネクタの実行可能なバージョンです。

構成ディレクトリ

SAS Event Stream Processing を配置すると、構成ファイルは特定の場所に置かれます。

表 1 デフォルトの構成ファイルの場所

オペレーティングシステム	場所
Linux	**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default**
Windows	**%ProgramData%\SAS\Viya\etc\SASEventStreamProcessingEngine\default**

Windows では、環境変数 ProgramData が定義されている場合、構成ディレクトリのパスの先頭、**\SAS**の前に置かれます。

2 つの環境変数を使用して、代替構成を設定することができます。

2

表 2 代替構成を設定するための環境変数

環境変数	説明
DFESP_CONFIG	デフォルトの場所を置換します。代替ログ、測定、および MAS ストアを使用するには、デフォルトの場所から指定フルパスにファイルをコピーする必要があります。ユーザーがこの変数を設定して実行した ESP サーバーは、代替場所で指定された構成値に従います。
DFESP_CONFIG_TAIL	デフォルトの場所に追加します。代替ログ、測定、および MAS ストアを使用するには、デフォルトの場所から指定サブディレクトリにファイルをコピーする必要があります。

注: 構成ファイルを編集するには、sas グループのメンバになる必要があります。

コネクタの概要

コネクタは何をしますか？

コネクタは、SAS Event Stream Processing パブリッシュ/サブスクライブ API を使用して、次のいずれかを実行します。

- イベントストリームをソースウィンドウにパブリッシュします。

  パブリッシュ操作では通常、次の処理を行います。

  - 指定されたソースからイベントデータを読み込みます
  - 実行中のイベントストリームプロセッサの特定のソースウィンドウにそのイベントデータをインジェクトします
- ウィンドウイベントストリームをサブスクライブします

  サブスクライブ操作は、実行中のイベントストリームプロセッサのウィンドウから指定されたターゲット(通常は継続的に)への出力イベントを書き込みます。

コネクタは同時にパブリッシュおよびサブスクライブしません。

$DFESP_HOME/lib/plugins にあるライブラリでコネクタを見つけることができます。Microsoft Windows プラットフォームでは、**%DFESP_HOME%\ bin \ plugins** にあります。

すべてのコネクタクラスは、コネクタライブラリに含まれる基本コネクタクラスから派生しています。基本コネクタライブラリには、初期化時にコネクタをロードするコネクタマネージャが含まれています。このライブラリは**$DFESP_HOME/lib/libdfxesp_connectors-Maj.Min** にあります。Maj.Min は配布のリリース番号を示します。

コネクタの例

C ++の例

C++のコネクタの例は、**$DFESP_HOME/examples/cxx** にあります。sample_connector ディレクトリには、dfESPconnector 基本クラスから派生したユーザー定義コネクタのソースコードが含まれています。また、サンプルコネクタを呼び出すサンプルコードも含まれています。コネクタの作成方法とコネクタマネージャによって読み込まれる方法の詳細については、“カスタムコネクタの作成と統合”を参照してください。

残りのコネクタの例は、既存のコネクタを呼び出すアプリケーションコードを実装しています。これらのコネクタは、初期化時にコネクタマネージャによってロードされます。これらの例は次のとおりです。

- db_connector_publisher

- db_connector_subscriber
- json_connector_publisher
- json_connector_subscriber
- socket_connector_publisher
- socket_connector_subscriber
- xml_connector_publisher
- xml_connector_subscriber

XML の例

XML モデルで指定されたコネクタの例は、**$DFESP_HOME/examples/xml** にあります。

- db_connector_publisher_xml
- db_connector_subscriber_xml
- url_connector_new
- url_connector_weather
- ws_connector_json
- ws_connector_xml
- xml_connector_publisher_xml
- xml_connector_subscriber_xml

コネクタの入手

C ++アプリケーションでコネクタの新しいインスタンスを取得するには、dfESPwindow::getConnector()メソッドを呼び出します。最初のパラメーターとしてコネクタの種類を渡し、オプションの 2 番目と 3 番目のパラメータとしてコネクタインスタンス名と"アクティブな"ブール値を渡します。

```
dfESPconnector *inputConn =
static_cast<dfESPconnector *>(input->getConnector("fs","inputConn", true));
```

パッケージ化されたコネクタの種類は次のとおりです。

- adapter
- bacnet (Linux のみ)
- db
- fs
- kafka
- modbus
- mq
- mqtt
- nurego
- opcua
- pylon
- project
- rmq

- sniffer
- smtp
- sol
- tdata
- tdlistener
- tibrv
- timer
- tva
- url
- uvc
- websocket
- pi (Windows のみ)

コネクタインスタンスが取得された後、その基本クラスのパブリックメソッドのいずれかを呼び出すことができます。これには、setParameter()が含まれます。これは、複数回呼び出して必須およびオプションのパラメータを設定できます。パラメータは、コネクタの開始前に設定する必要があります。

type パラメータは必須であり、すべてのコネクタに 共通です。それは pub または sub に設定する必要があります。

コネクタの種類に応じて、追加のコネクタ構成パラメータが必要です(このセクションの後半で説明します)。

オプションのプラグインの有効化とコネクタの除外

$DFESP_HOME/lib/plugins ディレクトリには、SAS Event Stream Processing でサポートされているプラグインオブジェクトの完全なセットが含まれています。ファイル名に "_cpi"を含むプラグインはコネクタです。

コネクタマネージャが起動すると、SAS Event Stream Processing は、**plugins** ディレクトリにあるすべてのコネクタを読み込みます。ただし、構成ディレクトリの **connectors.excluded** ファイルで指定されているものを除きます。

デフォルトでは、このファイルは SAS Event Stream Processing に同梱されていないサードパーティのライブラリを必要とするコネクタを指定します。リストされたコネクタは自動的にロードされないため、依存関係の欠落によるエラーは防止されます。

必要に応じて **connectors.excluded** を編集します。このファイルに有効なコネクタの種類をリストすることができます。

構成パラメータの設定

setParameter()メソッドを使用して、コネクタの必須およびオプションのパラメータを設定します。setParameter()は、必要な回数だけ使用できます。コネクタのパラメータを設定する必要があります。

type パラメータは必須であり、すべてのコネクタに 共通です。それは pub または sub に設定する必要があります。必要なその他のコネクタ構成パラメータは、コネクタの種類によって異なります。

ファイル内の構成パラメータの設定

構成ファイル内の構成パラメータは、完全または部分的に構成できます。パラメータのセットを指定し、そのセットにセクションラベルを付けます。次に、setParameter()を使用して、configfilesection パラメータをセクションラベルと同じに設定できます。これにより、セット全体が構成されます。いずれかのパラメータが冗長である場合は、setParameter()を使用して個別に構成するパラメータ値が優先されます。

パラメータのセットを構成すると、コネクタは構成ディレクトリ.の **connectors.config** にセクションラベルを見つけます。次に、そのセクションにリストされているパラメータを構成します。

次の行では、TestConfig の構成とラベル付けのための一連のコネクタパラメータを指定します

```
[testconfig]
type=pub
host=localhost
port=33340
project=sub_project
continuousquery=subscribeServer
window=tradesWindow
fstype=binary
fsname=./sorted_trades1M_256perblock.bin
```

セクションに必要な数のパラメータをリストし、ラベルを付けることができます。

オーケストレーションコネクタ

既定では、関連するプロジェクトが開始され、同時に実行されると、すべてのコネクタが 自動的に開始されます。コネクタのオーケストレーションが有効になると、コネクタのオーケストレーションが定義されているコネクタだけが開始されます。コネクタのオーケストレーションでは、コネクタの状態に応じて、プロジェクト内のコネクタの実行順序を定義できます。これにより、すべての 入力をオーケストレーションする自己完結型プロジェクトを作成できます。コネクタのオーケストレーションは、参照データをロードしたり、ストリーミングデータをインジェクトする前にバルクデータをウィンドウにインジェクトしたり、結合ウィンドウで役立ちます。

コネクタのオーケストレーションは、連続クエリの表現方法と同様に、有向グラフとして表すことができます。この場合、グラフのノードはコネクタのグループであり、エッジはグループの実行順序を示します。

コネクタは、通常、停止、実行、または終了という 3 つの状態のいずれかになります。無期限に(たとえば、メッセージバスから)パブリッシュできるサブスクライバコネクタとパブリッシャコネクタは、決して完了状態にはなりません。

コネクタの実行を別のコネクタの状態に依存させるには、両方のコネクタを異なるコネクタのグループに 定義する必要があります。グループには複数のコネクタを含めることができ、すべての 依存関係は個々のコネクタではなくグループによって 定義されます。

グループにコネクタを追加する場合は、対応するコネクタの状態も指定する必要があります。この状態は、グループ内のそのコネクタのターゲット状態を定義します。グループ内のすべてのコネクタがターゲット 状態に達すると、そのグループに依存する他のすべてのグループの 条件が満たされます。グループが満足されると、そのグループ内のすべてのコネクタが 実行状態になります。

G1、G2、G3、および G4 の 4 つのグループで 構成される次の構成を検討してください。

```
G1: {<connector_pub_A, FINISHED>, <connector_sub_B, RUNNING>}
G2: {<connector_pub_C, FINISHED>}
G3: {<connector_pub_D, RUNNING>}
G4: {<connector_sub_E, RUNNING>}
```

そして、次のオーケストレーションを考えてみましょう。

- G1 -> G3: G1 のすべてのコネクタが 目標状態になったら、G3 でコネクタを開始します。
- G2 -> G3: G2 のすべてのコネクタが 目標状態になったら、G3 でコネクタを開始します。これまでのオーケストレーションでは、G1 と G2 のコネクタがターゲット状態に達するまで G3 が起動しないことを意味します。
- G2 -> G4: G2 のすべてのコネクタが 目標状態になったら、G4 でコネクタを開始します。

G1 と G2 は他のグループに依存しないため、これらのグループ内のすべてのコネクタがすぐに 開始されます。構成により、次のオーケストレーションが発生します。

1. プロジェクトが開始されると、connector_pub_A、connector_sub_B、および connector_pub_C がただちに開始されます。
2. connector_pub_C が終了すると、connector_sub_E が開始されます。

3 connector_pub_D G3 のすべての条件が満たされた後、すなわち G1 と G2 の条件が満たされた後にのみ開始されます。したがって、connector_pub_A が終了し、connector_sub_B が実行されていて、connector_pub_C が終了したときにのみ開始されます。

コネクタのグループは、新しい dfESPconnectorGroup インスタンスへのポインタを返す newConnectorGroup()プロジェクトメソッドを呼び出すことによって定義されます。グループ名のみを渡すと、そのグループは他のグループに依存しません。逆に、グループインスタンスポインタのベクトルリストまたは変数リストを渡すこともできます。これは、新しいグループを実行するためにすべて満足しなければならないグループのリストを定義します。

グループを定義したら、dfESPconnectorGroup::addConnector() メソッドを呼び出してグループを追加できます。これは、コネクタインスタンス（dfESPwindow::getConnector()から返される）とそのターゲット状態をとります。

このオーケストレーションを定義する C++コードは次のとおりです。

```
dfESPconnectorGroup*G1= project->newConnectorGroup("G1");
G1-> addConnector(pub_A, dfESPabsConnector::state_FINISHED);
G1-> addConnector(sub_B, dfESPabsConnector::state_RUNNING);

dfESPconnectorGroup*G2= project->newConnectorGroup("G2");
G2-> addConnector(pub_C, dfESPabsConnector::state_FINISHED);

dfESPConnectorGroup*G3= project->newConnectorGroup("G3",2,G1,G2);
G3-> addConnector(pub_D, dfESPabsConnector::state_RUNNING);

dfESPConnectorGroup*G4= project->newConnectorGroup("G4",1,G2);
G4-> addConnector(sub_E, dfESPabsconnector::state_RUNNING);
```

対応する XML コードは次のとおりです。

```
<project-connectors>
    <connector-groups>
     <connector-group name='G1'>
      <connector-entry
       connector='contQuery_name/window_for_pub_A/pb_A'
       state='finished'/>
      <connector-entry
       connector='contQuery_name/window_for_sub_B/sub_B'
       state='running'/>
     </connector-group>
     <connector-group name='G2'>
      <connector-entry
       connector='contQuery_name/window_for_pub_C/pub_C'
       state='finished'/>
     </connector-group>
     <connector-group name='G3'>
      <connector-entry
        connector='contQuery_name/window_for_pub_D/pub_D'
        state='running'/>
     </connector-group>
     <connector-group name='G4'>
      <connector-entry
        connector='contQuery_name/window_for_sub_E/sub_E'
        state='running'/>
     </connector-group>
    </connector-groups>
    <edges>
     <edge source='G1' target='G3'/>
```

```
    <edge source='G2' target='G3'/>
    <edge source='G2' target='G4'/>
  </edges>
</project-connectors>
```

アダプタの概要

アダプタは、パブリッシュ/サブスクライブ API を使用して以下を行います。

- イベントストリームをエンジンにパブリッシュします
- エンジン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

多くのアダプタは、コネクタの実行可能バージョンです。したがって、ほとんどのアダプタの必須およびオプションのパラメータは、対応するコネクタのパラメータに直接マッピングされます。コネクタとは異なり、アダプタはネットワーク接続できます。

アダプタは、すべての SAS Event Stream Processing ライブラリに完全にアクセスできる必要があります。したがって、アダプタを使用するコンピュータシステムに SAS Event Stream Processing をインストールする必要があります。Event Stream Processing エンジンを実行するライセンスがない場合、そのシステムは製品ライセンスを使用すべきではありません。

アダプタは C ++、Java、Python で書かれています。

言語	アダプタ
C++	Bacnet パブリッシャ データベース イベントストリームプロセッサ ファイルとソケット Kafka Modbus MQTT OPC-UA PI Pylon パブリッシャ Rabbit MQ SMTP サブスクライバ Sniffer パブリッシャ Solace Systems TD リスナーサブスクライバ Teradata サブスクライバ Tervela データファブリック Tibco Rendezvous (RV) タイマパブリッシャ UVC パブリッシャ IBM WebSphere MQ

言語	アダプタ
Java **注:** Java アダプタは Java バージョン 8 を使用して構築されています。Java SE Runtime Environment 8 は、Java アダプタを実行しているすべてのプラットフォームで使用する必要があります。	SAS Cloud 分析サーバー Cassandra HDAT リーダー HDFS (Hadoop 分散ファイルシステム) Java Message Service (JMS) SAS LASR Analytic Server REST サブスクライバ SAS データセット Twitter パブリッシャ
Python	BoardReader Twitter GNIP

コネクタと同様に、アダプタはファイルから構成パラメータを取得できます。C ++アダプタは、構成ディレクトリ.内の構成ファイルを使用します。Java アダプタは **javaadapters.config** を使用します。

注: すべての Java アダプタパラメータは、Java アダプタ構成ファイルで完全に指定する必要があります。各 Java アダプタの **javaadapters.config** で指定する必要があるパラメータ名のリストについては、the table here.にリストされているアダプタの資料を参照してください。

アダプタを実行するときに、コマンド行で**-C** 引き数を使用して、構成ファイルにセクションラベルを指定することができます。詳細については、"ファイル内の構成パラメータの設定"を参照してください。

すべての C ++アダプタと Java アダプタは、ESP サーバーへの直接 TCP/IP 接続ではなく、Rabbit MQ サーバーを使用してパブリッシュまたはサブスクライブすることができます。各アダプタには、Rabbit MQ トランスポートの種類を使用するように指示する構成パラメータがあります。Rabbit MQ トランスポートの種類を使用する場合、アダプタはコードライブラリを使用して Rabbit MQ プロトコルを実装します。C ++で書かれたアダプタは rabbitmq-c ライブラリを使用します。Java で書かれたアダプタは、dfx-esp-rabbitmq-api.jar と rabbitmq-client.jar を使用します。Rabbit MQ Java クライアントライブラリ（http://www.rabbitmq.com/java-client.html）から rabbitmq-client.jar を入手する必要があります。rabbitmq-client.jar をシステムの **$DFESP_HOME/lib** にインストールします。

すべての C ++アダプタと Java アダプタは、ESP サーバーへの直接 TCP/IP 接続ではなく、Kafka クラスタを使用してパブリッシュまたはサブスクライブすることができます。各アダプタには、Kafka トランスポートの種類を使用するように指示する構成パラメータがあります。Kafka トランスポートの種類を使用する場合、アダプタはコードライブラリを使用して Kafka プロトコルを実装します。C ++で書かれたアダプタは、librdkafka ライブラリを使用します。Java で書かれたアダプタは、dfx-esp-kafka-api.jar および kafka-clients-*.jar を使用します。http://kafka.apache.org/downloads.html で kafka-clients - *.jar を入手する必要があります。システムの **$DFESP_HOME/lib** に kafka-clients-*.jar をインストールします。

すべてのアダプタは、ESP サーバーへの直接 TCP/IP 接続ではなく、Solace ファブリックを使用してパブリッシュまたはサブスクライブすることができます。各アダプタには、Solace トランスポートの種類を使用するように指示する構成パラメータがあります。Solace トランスポートの種類を使用する場合、アダプタはコードライブラリを使用して Solace プロトコルを実装します。C ++で書かれたアダプタは、libsolclient ライブラリを使用します。Java で書かれたアダプタは、dfx-esp-solace-api.jar と sol-*.jar を使用します。Solace Systems から sol-*.jar を入手する必要があります。ご使用のシステムの**$DFESP_HOME/lib** に sol-*.jar をインストールしてください。

$DFESP_HOME/bin にアダプタがあります。

アダプタコネクタの使用

コネクタは、ESP サーバーと同じプロセススペースでインスタンス化される C++クラスであることに注意してください。ESP サーバーを使用して、プロジェクトでのコネクタ実行順序のオーケストレーションを行うことができます。(コネクタのオーケストレーションの詳細については、“オーケストレーションコネクタ”を参照してください。)

コネクタと異なり、アダプタはコマンドラインツールで直接管理するスタンドアロンプロセスです。アダプタは、ESP サーバーまたは ESP クライアントとオーケストレーションすることはできません。ESP サーバーを使用してアダプタ実行順序のオーケストレーションを管理することができないため、釣り合いの取れた配置を行うには不便です。

多くのアダプタには対応するコネクタがありますが、いくつかは対応していません。アダプタコネクタはアダプタを包むラッパーとして機能し、ESP サーバーがアダプタをコネクタとして処理できるようにします。アダプタコネクタを使用すると、コネクタのオーケストレーションの際にアダプタのオーケストレーションを行えます。アダプタコネクタを使用して、アダプタプロセスを開始および終了することもできます。これにより、オーケストレーションの必要性に関係なく、ESP サーバーでアダプタのライフサイクルを管理できます。

アダプタコネクタは、関連するアダプタが実行されている間、その状態を"実行中"に設定し、アダプタが実行されていないときには"終了"に設定します。コネクタが停止すると、関連する実行中のアダプタは終了します。

次の表では、アダプタコネクタの XML プロジェクト定義の必須パラメータとオプションパラメータについて説明します。

表 3 アダプタコネクタの必須パラメータ

パラメータ	説明
command	コマンド行からアダプタを実行するために使用されるコマンドとオプションを指定します。command パラメータを定義するときは、必要なアダプタパラメータをすべて指定する必要があります。 モードの指定が必要なアダプタには、-k パラメータを含めます。 ここで command パラメータの定義に-h を含めないでください。これは、url パラメータで指定された URL または生成されたデフォルトの URL によって上書きされます
type	アダプタのモードを指定します。モードは“pub”または“sub”で、実行中のアダプタのモードと一致する必要があります。

表 4 アダプタコネクタのオプションパラメータ

パラメータ	説明
url	アダプタ接続の URL を指定します。このパラメータが含まれていない場合、デフォルトの URL が生成されます。 ウィンドウの URL とともにアダプタパラメータを含めます。たとえば、Twitter アダプタを使用している場合、-u "dfESP://server.sas.com:9951/proj/cq/src"と書くことができます。

注: アダプタコネクタが起動すると、関連付けられたアダプタのサブミットされたコマンドがログに表示されます。アダプタコネクタのトラブルシューティングを行うには、ログに送信されたコマンドを探し、すべてのパラメータが正しいことを確認します。

Bacnet コネクタとアダプタの使用

Bacnet コネクタの使用

Bacnet コネクタは、パブリッシュ用に事前定義された Bacnet オブジェクトセットから、イベントストリーム処理ソースウィンドウへのデータをポーリングします。

注: Bacnet コネクタは、Linux プラットフォームでのみサポートされます。

Bacnet デバイスとオブジェクトのセットは、ローカルの JSON 構成ファイルで定義されています。JSON 構成ファイルへのパスは、必要なコネクタ構成パラメータ bacnetconfigfile で指定します。各オブジェクトのポーリング間隔は、構成ファイルの JSON Period キー値によって定義されます。

JSON 構成ファイルの形式は次のとおりです。

```
[
 {
  “RemoteEndPoint”: “<device ipaddr:port>”,
  “MaxAPDULengthAccepted”: <max apdu length>,
  “LocalPort”: <local port to use with this device>,
  "NetworkNumber": <bacnet network number (optional)>,
  "MAC": "<bacnet mac address (optional)>”,
  “BacnetPoints”: [
   {
    “Topic”: “<any string unique to this point>”,
    “ObjectIdentifier”: <bacnet object identifier>,
    “PropertyIdentifier”: <bacnet property identifier>,
    “Period”: <polling interval in seconds>
   },
   ...
  ]
 },
 ...
]
```

コネクタを外部デバイスとして登録し、Bacnet デバイスに直接アクセスするには、BACnet-IP ブロードキャスト管理デバイス(BBMD)の IP アドレスとポートを構成します。

複数の Bacnet コネクタインスタンスは、1 つのプロセス内に存在することはできません。複数の Bacnet コネクタインスタンスを個別のプロセスとして実行するには、複数の Bacnet アダプタを実行する必要があります。これらのアダプタは、個別のプロセスとして、またはイベントストリーム処理モデルで構成された複数のアダプタコネクタとして手動で実行できます。

Bacnet コネクタのソースウィンドウスキーマには、次のフィールドが必要です。

スキーマフィールド	説明
id:int64	コネクタによって制御されるキーフィールド
adapterid:string	コネクタによって制御されるキーフィールド
topic:string	JSON トピックキー値からコピーされました
timestamp:stamp	イベントがいつ作成されたかを示すタイムスタンプ

Bacnet オブジェクトから読み込まれた値を格納する追加のフィールドをいくつでも定義できます。ポーリング間隔に従って Bacnet オブジェクトを読み取ると、受信した値が値の種類と互換性のあるユーザ定義フィールドにコピーされ、対応する ESP イベントが構築されます。受信した値の種類に一致するフィールドがない場合、致命的なエラーが報告されます。

表 5 パブリッシャ Bacnet コネクタの必須パラメータ

パラメータ	説明
bacnetbbmdaddress	BBMD の IP アドレスを指定します。
bacnetbbmdport	BBMD のポートを指定します。
bacnetconfigfile	Bacnet デバイスとオブジェクトを含む JSON 構成ファイルを指定します。

表 6 パブリッシャ Bacnet コネクタのオプションパラメータ

パラメータ	説明
bacnetipport	コネクタが使用するローカルポートを指定します。デフォルトのポート番号は 47808 です。
blocksize	パブリッシュされたイベントブロックに含めるイベントの数を指定します。デフォルト値は 1 です。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
ignoretimeouts	Bacnet デバイスからのプロパティ読み取りがタイムアウトになると、警告が記録され、続行されます。デフォルトでは、エラーを記録して停止します。
publishwithupsert	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
maxevents	パブリッシュするイベントの最大数を指定します。
transactional	イベントブロックの種類をトランザクションに設定します。デフォルトのイベントブロックの種類は標準です。

Bacnet-SAS Event Stream Processing のデータ型マッピングは、次のとおりです。

Bacnet データ型	SAS Event Stream Processing データ型
Boolean	ESP_INT32 ESP_INT64 ESP_DOUBLE
Unsigned Int	ESP_INT32 ESP_INT64 ESP_DOUBLE

Bacnet データ型	SAS Event Stream Processing データ型
Signed Int	ESP_INT32 ESP_INT64 ESP_DOUBLE
Real	ESP_DOUBLE
Double	ESP_DOUBLE
Character String	ESP_UTF8STR
Date	ESP_DATETIME ESP_TIMESTAMP
Enumerated	ESP_INT32 ESP_INT64 ESP_DOUBLE
Octet String	ESP_BINARY

Bacnet アダプタの使用

Bacnet パブリッシャアダプタは、アダプタのみのオプションパラメータに加えて、Bacnet コネクタ,と同じ機能を提供します。

注: Bacnet アダプタのサポートは、Linux プラットフォームでのみ使用できます。

パブリッシャ:

dfesp_bacnet_adapter -B *bacnetbbmdaddress* <-b *blocksize*> <-C *configfilesection*> -c *bacnetconfigfile* <-E *tokenlocation*> <-e> <-g *gdconfig*> -h *url* <-I *bacnetipport*> <-i> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxevents*> -P *bacnetbbmdport* <-Q> <-R> <-V> <-y *logconfigfile*> <-Z *transportconfigfile*>

パラメータ	説明
-B bacnetbbmdaddress	BBMD の IP アドレスを指定します。
-b blocksize	ブロックサイズを設定します。デフォルト値は 1 です。
-C [configfilesection]	接続パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。
-c bacnetconfigfile	Bacnet デバイスとオブジェクト情報を含む JSON 構成ファイルを指定します。

パラメータ	説明
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e	イベントがトランザクションであることを指定します。
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP のパブリッシュとサブスクライブの標準 URL を **dfESP://host:port/project/continuousquery/window** の形式で指定します。
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-I bacnetipport	アダプタが使用するローカルポートを指定します。デフォルトのポート番号は 47808 です。
-i	Bacnet デバイスからのプロパティ 読み取りがタイムアウトになると、警告が記録され、続行されます。デフォルトでは、エラーを記録して停止します。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、必要なクライアント構成ファイルを使用します。これらのファイルは、C++C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されています。
-m maxevents	パブリッシュするイベントの最大数を指定します。
-P bacnetbbmdport	BBMD のポートを指定します。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-R	挿入の代わりに opcode=Upsert を使用してイベントを作成します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-y logconfigfile	ログ構成ファイルを指定します。

パラメータ	説明
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、*"Kafka によるパブリッシャアダプタのフェイルオーバー" (SAS Event Stream Processing: 高度なトピック)*を参照してください。

BoardReader パブリッシャアダプタの使用

BoardReader アプリケーションは、メッセージフィードを集約します。SAS Event Stream Processing が提供する BoardReader アダプタは、Python スクリプトの集まりです。これらのスクリプトは、C パブリッシュ/サブスクライブおよび JSON ライブラリを呼び出して BoardReader フィードからイベントブロックを作成し、ソースウィンドウにそれらを公開します。各スクリプトは、特定の BoardReader コンテンツソースへのアクセスを提供します。これらのコンテンツソースには、掲示板、ブログ、ニュース、YouTube、およびレビューが含まれます。

コンテンツソースへのアクセスを 構成するために必要なパラメータは、**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/boardreader-*-config.txt** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\boardreader-*-config.txt** (Windows)です。これらのファイルのサンプルバージョンは、SAS Event Stream Processing ディストリビューションに含まれています。

構成する必要があるパラメータは次のとおりです。

- customer_project_id
- customer_query_id
- key
- query

サンプルファイルに示されているように、他のパラメータはデフォルト値のままにすることができます。

アダプタは、構成された照会を独立したスレッドで実行し、それらのスレッドが完了するのを待ちます。各スレッドは応答を受け取ると、同じスレッドからイベントブロックをイベントストリーム処理サーバーにパブリッシュします。各クエリは、そのクエリの filter_date_from および filter_date_to パラメータごとにフィルタリングされたコンテンツソースからの応答を収集します。

アダプタがストリーミングモード（ request_mode パラメータのデフォルト設定）になると、すべてのスレッドが完了するのを待ちます。次に、request_interval パラメータと request_interval_unit パラメータで指定された間隔の後にプロセスを繰り返します。クエリの次の反復を実行する前に、各クエリの filter_date_from および filter_date_to パラメータが構成ファイルで更新されます。これにより、次回の呼び出しが中断されずにコンテンツソースから継続的にストリームされることが保証されます。

受信したすべての JSON レスポンスは、*"JSON メッセージングのパブリッシュ/サブスクライブ API サポート" (SAS Event Stream Processing: パブリッシュ/サブスクライブ API)*で説明した JSON 解析ルールを使用してイベントブロックに変換されます。対応するソースウィンドウには、アダプタパラメータ

esp-ignoremissingschemafields を指定しない限り、受信したすべての JSON キーに対応するフィールドが含まれている必要があります。Upsert を使用するようにアダプタを構成しないかぎり、イベントは挿入が演算コードで作成されます。必要に応じて、esp-matchsubstrings パラメータを使用して JSON フィルタリング関数を有効にすることができます。

ソースウィンドウでキーフィールドを割り当てます。キーフィールドとして JSON キーを使用することが現実的でない場合は、次のキーフィールドのいずれかまたは両方をソースウィンドウスキーマに追加できます。

```
eventindex*:int64
adapterindex*:string
```

eventindex キーフィールドは、入力 JSON から構築されたすべてのイベントに対して増分されます。adapterindex key フィールドには、アダプタのすべてのインスタンスで一意の定数 GUID 値が格納されます。これらの 2 つのキーフィールドを組み合わせることで、アダプタの複数のインスタンスがキーを複製せずに同じソースウィンドウにイベントをパブリッシュすることができます。

ソースウィンドウのスキーマには、次の 2 つのフィールドも含める必要があります。

- ProjectID:string
- Query:string

これらのフィールドの内容は、相関の目的で、customer_project_id および customer_query_id 構成パラメータの値を反映します。

使用法:

dfesp_*_boardreader -config-txt *configtxt* -esp-url *url* <-dev-mode-history-to-disk> <-dev-mode-resp-to-disk> <-email-from *emailfrom* > <-email-level *emaillevel* > <-email-smtphost *emailsmtphost* <-email-subject *emailsubject* >> <-email-to *emailto* > <-esp-dateformat *dateformat* > <-esp-logconfigfile *logconfigfile* > <-esp-loglevel *loglevel* > <-esp-ignoremissingschemafields> <-esp-matchsubstrings *substrings* > <-esp-publishwithupsert>

パラメータ	説明
-config-txt configtxt	BoardReader コンテンツソース構成パラメータを含む構成ディレクトリ内のテキストファイルの名前を指定します。構成ディレクトリ.内のサンプルファイルを参照してください。
-dev-mode-history-to-disk	追加のデバッグを提供します: work_dir パラメータを構成していない限り、構成ディレクトリの SQLite データベース形式のドキュメントをディスクに書き込みます。
-dev-mode-resp-to-disk	追加のデバッグを提供します。work_dir パラメータを構成していない限り、構成ディレクトリのディスクへの応答を書き込みます。
-email-from emailfrom	電子メールロギングメッセージの送信元アドレスを指定します。
-email-level emaillevel	電子メールのログレベルを指定します。許可される値は DEBUG、INFO、WARNING、ERROR、および CRITICAL です。デフォルト値は CRITICAL です。
-email-smtphost emailsmtphost	電子メールロギングメッセージの SMTP ホストを指定します。
-email-subject emailsubject	メールロギングメッセージの題名を指定します。
-email-to emailto	メールロギングメッセージの宛先アドレスを指定します。

パラメータ	説明
-esp-dateformat dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-esp-ignoremissingschemafields	スキーマで JSON キーが見つからない場合に無視して続行するように指定します。デフォルトでは、エラーが報告され、キーがないときにプロセスは終了します。
-esp-logconfigfile logconfigfile	ログ構成ファイルの名前を指定します。
-esp-loglevel loglevel	ロギングレベルを指定します。指定できる値は、TRACE、DEBUG、INFO、WARN、ERROR、または FATAL です。デフォルト値は INFO です。
-esp-matchsubstrings substrings	カンマ区切りの fieldname:value のペアのリストを指定します。ここで、fieldname 文字列フィールドに value の部分文字列を含まない入力イベントは削除されます。比較は大文字と小文字を区別しません。
-esp-publishwithupsert	opcode = Upsert でイベントをパブリッシュすることを指定します。デフォルトでは、イベントは opcode = Insert でパブリッシュされます。
-esp-url url	パブリッシャの標準 URL を "dfESP://host:port/project/continuousquery/window"の形式で指定します。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

SAS Cloud Analytic Services アダプタの使用

サブスクライバアダプタは、SAS Cloud Analytics Service Server 内のグローバルテーブルへのイベントストリーム処理イベントの追加をサポートします。このテーブルには、各 SAS Event Stream Processing イベントに関連付けられた演算コードを含む"_opcode"という名前の列が含まれています。

パブリッシャアダプタは、SAS Cloud Analytic Services サーバーのグローバルテーブルからすべての 行をフェッチすることをサポートしています。パブリッシャアダプタは、opcode=Insert（-u が構成されていない場合）を持つイベントとしてこれらの行をソースウィンドウにインジェクトします。

クライアントターゲットプラットフォームの **DFESP_JAVA_TRUSTSTORE** および **SSLCALISTLOC** 環境変数を定義する必要があります。SAS Cloud Analytic Services サーバーでは、クライアント接続を TLS 暗号化で保護する必要があるため、 **DFESP_JAVA_TRUSTSTORE** はトラストストアの.jks ファイルの場所を設定し、**SSLCALISTLOC** は必要な証明書を含む.pem ファイルの場所を設定します。

-p パラメータを使用し、かつ-E パラメータで暗号化を有効にしていない場合は、暗号化されていない形式で caspassword を指定する必要があります。パスワードの暗号化されたバージョンは、OpenSSL を使用して生成することができます。OpenSSL 実行可能ファイルは、SAS Event Stream Processing System の暗号化および認証オーバーレイインストールに含まれています。コンソールで OpenSSL を使用して暗号化されたパスワードを表示するには、次のコマンドを使用します。

echo "*caspassword*" | openssl enc -e -aes-256–cbc -a -salt -pass pass:"SASespCASadapterUsedByUser=*casusername*"

暗号化されたパスワードを使用して、-p パラメータに caspassword を指定し、-E パラメータで暗号化を有効にします。

サブスクライバの使用:

dfesp_cas_adapter -H *cashostport* -h *url* -k *sub* -t *castable* <-a *commitblocks*> <-b *buffersize*> <-C *configfilesection*> <-E> <-g *gdconfigfile*> <-i *cassessionid*> <-L *pubsublib*> <-l *severe* | *warning* | *info*> <-m *caslib*> <-Z *transportconfigfile*> <-n *casusername*> <-O *tokenlocation*> <-p *caspassword*> <-s *columns*> <-V>

注: サブスクライバの場合、SSAS Cloud Analytic Services 内の指定されたテーブルは存在する必要はありません。そのとき、その列は、イベントストリームプロセッサウィンドウ内の列のサブセットと一致する必要があります。既存のテーブルに行が追加されます。*cassessionid* パラメータを指定しないと、テーブルはグローバルテーブルにプロモートされます。

パブリッシャの使用:

dfesp_cas_adapter -H *cashostport* -h *url* -k *pub* -t *castable* <-b *blocksize*> <-C *configfilesection*> <-c> <-E> <-e> <-g *gdconfigfile*> <-i *cassessionid*> <-L *pubsublib*> <-l *severe* | *warning* | *info*> <-m *caslib*> <-Z *transportconfigfile*> <-n *casusername*> <-O *tokenlocation*> <-p *caspassword*> <-s *columns*> <-u> <-V> <-W *maxevents*>

注: パブリッシャの場合、SAS Cloud Analytic Services 内の指定されたテーブルが存在する必要があります。その列は、イベントストリームプロセッサのソースウィンドウ内の列のサブセットと一致する必要があります。さらに、*cassessionid* パラメータを指定しない場合、テーブルはグローバルでなければなりません。イベントストリームプロセッサソースウィンドウスキーマの 最初のフィールドは行番号用に予約されており、INT64 として定義する必要があります。この最初のフィールドは、イベントのキーフィールドとして機能します。

パラメータ	説明
-a commitblocks	一度に SAS Cloud Analytic Services テーブルにコミットするイベントブロックの数を指定します。デフォルト値は 8 です。このパラメータは、待機時間のスループットを調整します。値を大きくして、コミットされる前に SAS Cloud Analytic Services テーブルに追加されたイベントブロックの数を増やします。
-b blocksize	イベントブロックごとのイベント数を指定します。
-b buffersize	バッファーサイズを指定します。デフォルト値は 512 です。このバッファリングパラメータは、アダプタが SAS Cloud Analytic Services に送信する前にバッファリングするテーブル行の数を指定します。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-c	すべてのイベントがソースウィンドウにインジェクトされた後にプロジェクトを休止することを指定します。 **javaadapters.config** パラメータ名: quiesceproject
-E	*caspassword* が暗号化されることを指定します。 j**avaadapters.config** パラメータ名:pwdencrypted
-e	イベントブロックがトランザクションであることを指定します。デフォルトのイベントブロックの種類は標準です。 **javaadapters.config** パラメータ名: transactional

パラメータ	説明
-g gdconfigfile	クライアントの保証された配信構成ファイルを指定します。
-H cashostport	SAS Cloud Analytic Services のホスト名とポートを指定します。
-h url	dfESP 標準のパブリッシュおよびサブスクライブ URL を dfESP://*host*:*port*/*project*/*continuousquery*/*window* の形式で指定します。 サブスクライバに次の行を追加します。?snapshot=true | false です。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに以下を追加します。 ?collapse=true | false ?rmretdel=true | false
-i cassessionid	接続先のセッションの UUID を指定します。指定されていない場合、SAS Cloud Analytic Services アダプタは新しい SAS Cloud Analytic Services セッションを作成します。
-k sub | pub	サブスクライブまたはパブリッシュを指定します。
-L native | rabbitmq | solace | tervela | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに rabbitmq、solace、tervela、または kafka トランスポートを指定する場合は、必要なクライアント構成ファイルを使用します。これらのファイルは、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されています。
-l severe | warning | info	アプリケーションのログレベルを指定します。
-m caslib	SAS Cloud Analytic Services テーブルを含む SAS Cloud Analytic Services ライブラリを指定します。デフォルトは現在アクティブな SAS Cloud Analytic Services ライブラリです。
-n casusername	SAS Cloud Analytic Services の認証に使用されるユーザー名を指定します。
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-p caspassword	SAS Cloud Analytic Services セッションの認証に使用されるパスワードを指定します。*caspassword* が指定されておらず、*casusername* が指定されている場合、そのユーザーの **authoinfo/netrc** ファイルからパスワードが抽出されます。

パラメータ	説明
-s columns	SAS Cloud Analytic Services テーブルを読み書きする列のサブセットを指定します。フォーム *c1_name*、... *cn_name* を使用します。 **注:** SAS Cloud Analytic Services テーブルの列名がイベントストリーム処理ウィンドウのすべてのフィールド名と一致する場合は、-s パラメータを使用する必要はありません。イベントストリーム処理スキーマは、SAS Cloud Analytic Services スキーマと 1 対 1 で対応します(パブリッシャに必要な追加の予約キーフィールドとサブスクライバによって作成された演算コードフィールドは除きます)。
-t castable	SAS Cloud Analytic Services テーブルの名前を指定します。
-u	Insert の代わりに opcode = Upsert を使用してイベントを作成します。 **javaadapters.config** パラメータ名: publishwithupsert
-V	致命的なエラーが報告された場合は、アダプタを再起動します。 **javaadapters.config** パラメータ名: restartonerror
-W maxevents	パブリッシュするイベントの最大数を指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-L パラメータで指定されたトランスポートの種類によって異なります。 パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-L パラメータで指定されたトランスポートの種類によって異なります。 -L solace の場合、デフォルトは**./solace.cfg** です。 -L tervela の場合、デフォルトは**./client.config** です。 -L rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -L kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

注: SAS Cloud Analytic Services テーブルにイベントストリーム処理ウィンドウよりも少ない列が含まれている場合は、 -s を使用して、テーブルの読み取りまたは書き込みを行うイベントストリーム処理フィールドのサブセットを選択します。パブリッシャの場合、ウィンドウ内の一致しないフィールドにはヌル値がロードされます。サブスクライバの場合、ウィンドウ内の一致しないフィールドは、SAS Cloud Analytic Services テーブルに追加された変数リストから除外されます。

SAS Cloud Analytic Services と SAS Event Steam Processing のデータ型のマッピングは次のとおりです。

SAS Cloud Analytic Services の型	SAS Event Stream Processing データ型
INT32	INT32
INT64	INT64
DOUBLE	DOUBLE
DATE	DATETIME

SAS Cloud Analytic Services の型	SAS Event Stream Processing データ型
CHAR	UTF8STR または RUTF8STR
VARCHAR	UTF8STR または RUTF8STR
DECSEXT	MONEY
TIME	INT64
DATETIME	TIMESTAMP
VARBINARY	BINARY

注: SAS Cloud Analytic Services の DECQUAD 型と BINARY 型には、イベントストリームプロセッサのデータ型マッピングはありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

Cassandra アダプタの使用

Cassandra アダプタは、Java パブリッシャとサブスクライバの SAS Event Stream Processing クライアントをバンドルする dfx-esp-cassandra-adapter.jar で利用できます。サブスクライバクライアントは、SAS Event Stream Processing Engine イベントブロックを受信し、囲まれたイベントをターゲットの Cassandra キースペースおよびテーブルの挿入/更新/削除オペレーションに変換します。パブリッシャクライアントは、Cassandra 結果セットの行をイベントに変換し、Event Stream Processing エンジンのソースウィンドウにインジェクトします。

アダプタには Apache Cassandra の Datastax Java ドライバのバージョン 3.0.0 以上が必要です。これは https://github.com/datastax/java-driver からダウンロードする必要があります。このドライバは、Cassandra バージョン 1.2～3.0 以降をサポートしています。これは、接続中に使用するネイティブプロトコルのバージョンをネゴシエートします。クライアントターゲットプラットフォームは、環境変数 DFESP_DATASTAX_JARS を定義して、Datastax Java ドライバ JAR ファイルの場所を指定する必要があります。

アダプタは、データベースのパブリッシャとサブスクライバと 機能的に似ています。

サブスクライバの場合、サブスクライブされたすべてのイベントブロックは、インサートと更新と削除の別々のリストに分かれています。各リストは、一連の挿入、更新、および削除のバッチに追加されるプリペアドステートメントのセットに変換されます。次に、バッチは次の順序で実行されます。

1 バッチを挿入
2 バッチの更新
3 バッチを削除

デフォルトでは、バッチは、サブスクライブされた各イベントブロックごとに 1 回構築され、実行されます。*batchsize* および *batchsecs* パラメータを構成して、バッチ処理を変更できます。

パブリッシャの場合、ターゲットの Cassandra キースペースとテーブルに 対してユーザー構成の select 文が実行されます。結果セットの各行は、ソースウィンドウにインジェクトされた挿入イベント(または構成されている場合は Upsert)に変換されます。これらの操作の後、アダプタは終了します。

サブスクライバの使用:

dfesp_cassandra_subscriber -f *node* -k *keyspace* -t *table* -u *url* <-a *loadbalancingpolicy* > <-b *batchsize* > <-c *configfilesection* > <-g *gdconfigfile* > <-l native | rabbitmq | solace | tervela | kafka> <-m *compression* > <-n

username > <-O *tokenlocation* > <-o *loglevel* > <-p *password* > <-S> <-s *batchsecs* > <-y *retrypolicy* > <-x *port* > <-V> <-Z *transportconfigfile*>

パブリッシャ:

dfesp_cassandra_publisher -e *selectstatement* -f *node* -k *keyspace* -t *table* -u *url* <-a *loadbalancingpolicy* > <-b *blocksize* > <-c *configfilesection* > <-g *gdconfigfile* > <-l native | rabbitmq | solace | tervela | kafka> <-m *compression* > <-n *username* > <-O *tokenlocation* > <-o *loglevel* > <-p *password* > <-r> <-S> <-s> <-y *retrypolicy* > <-x *port* > <-Q> <-V> <-W *maxevents*> <-Z *transportconfigfile*>

パラメータ	説明
-a loadbalancingpolicy	カンマ区切りのラップされたポリシーを使用して、Cassandra ロードバランシングポリシーを指定します。デフォルトは"TokenAware,DCAwareRoundRobin"です。
-b batchsize	バッチの挿入、更新、および削除セットごとのステートメントの合計数を指定します。デフォルトはイベントブロックサイズです。
-b blocksize	イベントブロックごとのイベント数を指定します。
-c [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-e selectstatement	Cassandra テーブルから行を引き出すために使用する CQL ステートメントを指定します。
-f node	Cassandra クラスタノードの IP アドレスまたはホスト名を指定します。
-g gdconfigfile	クライアントの保証された構成ファイルを指定します。
-k keyspace	Cassandra のキースペースを 指定します。
-l native | rabbitmq | solace | tervela | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに rabbitmq、solace、tervela、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用してください。
-m compression	Cassandra 転送圧縮を指定します。有効な値は"NONE"、"LZ4"、"SNAPPY"です。デフォルトは"NONE"です。
-n username	Cassandra ホストにログオンするユーザー名を指定します。
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-o severe | warning | info	アプリケーションのログレベルを指定します。
-p password	指定されたユーザー名で使用するパスワードを指定します。

パラメータ	説明
-Q	すべてのイベントがソースウィンドウにインジェクトされた後、プロジェクトを休止します。 **javaadapters.config** パラメータ名: quiesceproject
-r	イベントブロックがトランザクションであることを指定します。デフォルトのイベントブロックの種類は標準です。 **javaadapters.config** パラメータ名: transactional
-S	デフォルトのプラットフォーム JSSE オプションを使用して、Cassandra 接続で SSL を有効にします。 **javaadapters.config** パラメータ名: ssl
-s batchsecs	未完了のバッチを保持する最大秒数を指定します。デフォルトは 0 です。
-s	Insert の代わりに opcode=Upsert を使用してイベントを作成します。 **javaadapters.config** パラメータ名: publishwithupsert
-t table	Cassandra テーブルを指定します。
-u url	dfESP 標準のパブリッシュおよびサブスクライブ URL を dfESP://*host*:*port*/*project*/*continuousquery*/*window* の形式で指定します。 サブスクライバに次の行を追加します。?snapshot=true | false です。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに以下を追加します。 ?collapse=true | false ?rmretdel=true | false
-V	致命的なエラーが報告された場合は、アダプタを再始動します。 **javaadapters.config** パラメータ名: restartonerror
-W maxevents	パブリッシュするイベントの最大数を指定します。
-x port	Cassandra クラスターノードのポートを指定します。デフォルト値は 9042 です。
-y retrypolicy	カンマで区切られたオプションのラップポリシーを使用して、Cassandra 再試行ポリシーを指定します。デフォルトは“Default”です。

パラメータ	説明
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、*“Kafka によるパブリッシャアダプタのフェイルオーバー” (SAS Event Stream Processing: 高度なトピック)*を参照してください。

データベースコネクタとアダプタの使用

データベースコネクタの使用

データベースコネクタの使用方法の概要

データベースコネクタは、パブリッシュ操作とサブスクライブ操作の両方をサポートします。コネクタ構成パラメータ connectstring には、データベースの種類を指定する driver=ステートメントが必要です。サポートされるドライバの値は次のとおりです。

- aster
- db2
- db2zos
- greenplum
- hive
- hawq
- impala
- mysql
- netezza
- odbc
- oracle
- postgres
- redshift
- saphana
- mssqlsvr
- sapiq
- teradata

- vertica

ODBCINI 環境変数、odbc.ini ファイル、または DSN 定義は必要ありません。対応するすべてのパラメータは、ドライバ指定に続いて connectstring パラメータで直接定義されます。ドライバ固有のパラメータの詳細については、*SAS Federation Server: Administrator's Guide* の Driver Reference for SAS Federation Server を参照してください。

一部のドライバでは、クライアントライブラリの追加インストールも必要です。この要件は、Driver Reference for SAS Federation Server にも記載されています。

driver=odbc を指定すると、標準の ODBC 準拠のドライバをインストールして使用できることに注意してください。この場合、ドライバは odbc.ini のような他のシステム要件を持つ可能性があります。Linux インストールで"driver=odbc"を使用する場合は、connectstring パラメータの指定時に必ず"dm_unicode=UCS2"を含めて、unixODBC Driver Manager の呼び出し時にドライバが UTF-16/UCS-2 を正しく使用するようにしてください。

Windows インストールの場合、一部のドライバでは DSN の定義が必要です。DSN を定義するには、オプションの ODBC コンポーネントがインストールされていることを確認します。次に、Windows ODBC データソース管理者を使用して DSN を構成できます。このアプリケーションは、**Windows のコントロールパネル**の**管理ツール**にあります。Windows 8 以降、このアイコンの名前は"ODBC データソース"です。64 ビットオペレーティングシステムでは、32 ビット版と 64 ビット版があります。

Windows 環境で DSN を作成するには、次の手順を実行します。

1 Windows の検索ウィンドウに odbc と入力します。デフォルトの ODBC データソースアドミニストレータが表示されます。

2 **システム DSN** タブを選択し、**追加**をクリックします。

3 リストから適切なドライバを選択し、**完了**をクリックします。

4 **ドライバセットアップ**ダイアログボックスに情報を入力します。

5 完了したら **OK** をクリックします。

この DSN は接続文字列で提供されます。

コネクタパブリッシャは、ユーザーによって構成された単一の SQL ステートメントを使用してデータベースから結果セットを取得します。追加の結果セットは、コネクタを停止して再始動することで取得できます。コネクタサブスクライバは、ウィンドウ出力イベントを、ユーザーが構成したデータベーステーブルに書き込みます。

必要に応じて、connectstring パラメータの pwd フィールドを暗号化されたパスワードで構成できます。暗号化されたパスワードのバージョンは、システムにインストールする必要がある OpenSSL を使用して生成することができます。SAS Event Stream Processing システムの暗号化と認証のオーバーレイをインストールすると、付属の OpenSSL 実行可能ファイルをインストールします。コンソールで次のコマンドを使用して、OpenSSL を起動し、暗号化されたパスワードを表示します。

```
echo "pwd in connectstring" | openssl enc -e -aes-256-cbc -a -salt
-pass pass:"SASespDBconnectorUsedByUser=uid in connectstring"
```

暗号化されたパスワードを connectstring パラメータにコピーし、pwdencrypted パラメータを有効にします。

データベースコネクタでは、次のパラメータを使用します。

注: Teradata プラットフォームの場合、個別にパッケージ化されたコネクタとアダプタでは、パフォーマンスを向上させるために Teradata Parallel Transporter が使用されます。

表 7 サブスクライバデータベースコネクタの必須パラメータ

パラメータ	説明
connectstring	ドライバの種類と他のドライバ固有のパラメータを指定します。

パラメータ	説明
desttablename	ターゲットテーブル名を指定します。
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。
type	サブスクライブを指定します。“sub”にしてください。

表 8 パブリッシャデータベースコネクタの必須パラメータ

パラメータ	説明
connectstring	ドライバの種類と他のドライバ固有のパラメータを指定します。
type	パブリッシュを指定します。“pub”にしてください。

表 9 サブスクライバデータベースコネクタのオプションパラメータ

パラメータ	説明
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
commitrows	バッファする出力行の最小数を指定します。
commitsecs	未完了のコミットバッファに保持する最大秒数を指定します。
ignoresqlerrors	以前の挿入、更新、または削除でエラーが発生した場合でも、コネクタが引き続き挿入、更新、および削除をデータベーステーブルに書き込むことを可能にします。
maxcolbinding	サポートされている文字列の幅の最大値を指定します。デフォルト値は 4096 です。
pwdencrypted	connectstring の pwd フィールドが暗号化されることを指定します。
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバによって受信されたイベントブロックからすべての 削除イベントを削除することを指定します。

表 10 パブリッシャデータベースコネクタのオプションパラメータ

パラメータ	説明
blocksize	パブリッシュされたイベントブロックに含めるイベントの数を指定します。デフォルト値は 1 です。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
greenplumlogminer	Greenplum ログマイナーモードを使用可能にします。"ログマイナーモードの使用"。
logminerdbname	Greenplum ログマイナーモードの queries_history テーブルを含む gpperfmon データベースを指定します。dd-mmm-yyy hh:mm:ss という形式を使用します。
logminerschemaowner	Oracle または Greenplum ログマイナーモードを使用する場合のスキーマ所有者を指定します。 詳細については、"ログマイナーモードの使用"を参照してください。
logminerstartdatetime	Oracle または Greenplum ログマイナーモードを使用する場合の開始日時を指定します。 詳細については、"ログマイナーモードの使用"を参照してください。
logminertablename	Oracle または Greenplum ログマイナーモードを使用する場合のテーブル名を指定します。 詳細については、"ログマイナーモードの使用"を参照してください。
maxcolbinding	サポートされている文字列の幅の最大値を指定します。デフォルト値は 4096 です。
maxevents	パブリッシュするイベントの最大数を指定します。
oraclelogminer	Oracle ログマイナーモードを使用可能にします。"ログマイナーモードの使用"。
publishwithupsert	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
pwdencrypted	connectstring の pwd フィールドが暗号化されることを指定します。
selectstatement	ソースデータベースで実行される SQL ステートメントを指定します。oraclelogminer および greenplumlogminer が有効でない場合に必要です。
transactional	イベントブロックの種類をトランザクションに設定します。デフォルトのイベントブロックの種類は標準です。

ソースまたはターゲットのデータベーステーブルの 列数とそのデータ型は、関係するイベントストリームプロセッサウィンドウのスキーマと互換性がなければなりません。

サブスクライバイベントストリームプロセッサから SQL データ型へのマッピング

サブスクライバイベントストリームプロセッサのデータ型	SQL データ型
ESP_UTF8STR	SQL_CHAR、SQL_VARCHAR、SQL_LONGVARCHAR、SQL_WCHAR、SQL_WVARCHAR、SQL_WLONGVARCHAR、SQL_BINARY、SQL_VARBINARY、SQL_LONGVARBINARY、SQL_GUID
ESP_INT32	SQL_INTEGER、SQL_BIGINT、SQL_DECIMAL、SQL_BIT、SQL_TINYINT、SQL_SMALLINT
ESP_INT64	SQL_BIGINT、SQL_DECIMAL、SQL_BIT、SQL_TINYINT、SQL_SMALLINT
ESP_DOUBLE	SQL_DOUBLE、SQL_FLOAT、SQL_REAL、SQL_NUMERIC、SQL_DECIMAL
ESP_MONEY	SQL_DOUBLE (ESP_DOUBLE に変換)、SQL_FLOAT、SQL_REAL、SQL_NUMERIC (SQL_NUMERIC に変換)、SQL_DECIMAL (SQL_NUMERIC に変換)
ESP_DATETIME	SQL_TYPE_DATE (年/月/日のみ設定)、SQL_TYPE_TIME (時/分/秒のみ設定)、SQL_TYPE_TIMESTAMP (小数点以下の秒数を 0 に設定)
ESP_TIMESTAMP	SQL_TYPE_TIMESTAMP

Publisher SQL から Event Stream Processor データ型のマッピング

パブリッシャ SQL データ型	イベントストリームプロセッサのデータ型
SQL_CHAR	ESP_UTF8STR
SQL_VARCHAR	ESP_UTF8STR
SQL_LONGVARCHAR	ESP_UTF8STR
SQL_WCHAR	ESP_UTF8STR
SQL_WVARCHAR	ESP_UTF8STR
SQL_WLONGVARCHAR	ESP_UTF8STR
SQL_BIT	ESP_INT32、ESP_INT64
SQL_TINYINT	ESP_INT32、ESP_INT64
SQL_SMALLINT	ESP_INT32、ESP_INT64
SQL_INTEGER	ESP_INT32、ESP_INT64
SQL_BIGINT	ESP_INT64
SQL_DOUBLE	ESP_DOUBLE、ESP_MONEY (ESP_DOUBLE からのアップキャスト)
SQL_FLOAT	ESP_DOUBLE、ESP_MONEY (ESP_DOUBLE からのアップキャスト)

パブリッシャ SQL データ型	イベントストリームプロセッサのデータ型
SQL_REAL	ESP_DOUBLE
SQL_TYPE_DATE	ESP_DATETIME (年/月/日のみ設定)
SQL_TYPE_TIME	ESP_DATETIME (時/分/秒のみ設定)
SQL_TYPE_TIMESTAMP	ESP_TIMESTAMP、ESP_DATETIME
SQL_DECIMAL	ESP_INT32 (スケール= 0 で精度が<= 10 の場合のみ)、ESP_INT64 (スケール= 0 で精度が<= 20 である場合のみ)、ESP_DOUBLE
SQL_NUMERIC	ESP_INT32 (スケール= 0 で精度は<= 10)、ESP_INT64 (スケール= 0 の場合のみ精度は<= 20)、ESP_DOUBLE、ESP_MONEY (SQL_NUMERIC から変換した場合)
SQL_BINARY	ESP_UTF8STR
SQL_VARBINARY	ESP_UTF8STR
SQL_LONGVARBINARY	ESP_UTF8STR

ログマイナーモードの使用

概要

データベースコネクタは、Oracle または Greenplum の 2 つの特別なログマイナーモードのいずれかで実行できます。これらのモードはパブリッシャにのみ適用されますこれらは、Oracle Log Miner データベースまたは Greenplum **gpperfmon** データベースの **queries_history** テーブルから行をフェッチするように設計されています。行がフェッチされた後、行ごとのイベントが、固定の、よく知られたフィールドのサブセットを持つソースウィンドウにパブリッシュされます。

適切なパラメータ oraclelogminer または greenplumlogminer を有効にして、ログマイナーモードを構成します。

Oracle の要件については、Oracle の管理ガイド(http://docs.oracle.com/cd/B28359_01/server.111/b28319/logminer.htm)を参照してください。要約すると、データベースは補助ログを有効にしてアーカイブログモードにする必要があります。ALTER DATABASE ADD SUPPLEMENTAL LOG DATA です。ログを読み取るユーザーには、SELECT_CATALOG_ROLE と SELECT ANY TRANSACTION 権限を付与します。

Oracle ログマイナーモードの使用

logminerschemaowner パラメータを使用して構成するユーザー名には、Oracle Log Miner データベースに対して SELECT_CATALOG_ROLE、EXECUTE_CATALOG_ROLE、SELECT_ANY_TRANSACTION という次の権限が必要です。コネクタは、ユーザーが指定した開始時刻を指定して、選択操作を繰り返し実行してログからレコードを引き出します。後続の各選択では、最後に受信した SQL_REDO 応答のタイムスタンプに等しい開始時間が指定されます。

具体的には、コネクタは次のステートメントで初期開始時刻と終了時刻を定義します。

Declare startdate date := to_date('logminerstartdatetime','DD-MON-YYYY HH24:MI:SS');begin execute immediate 'ALTER SESSION SET NLS_DATE_FORMAT = ' || chr(39) || 'DD-MON-YYYY HH24:MI:SS' || chr(39);dbms_logmnr.start_logmnr(OPTIONS=> DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG +DBMS_LOGMNR.CONTINUOUS_MINE,STARTTIME=>startdate,ENDTIME=>SYSDATE-(1/86400));end;

コネクタは、以下のステートメントを使用して繰り返し照会を実行します。

Select OPERATION, TIMESTAMP, replace (SQL_REDO,chr(0),' ') SQL_REDO, CSF from v$logmnr_contents where SEG_OWNER='logminerschemaowner' and SEG_NAME='logminertablename'and OPERATION in (‘INSERT’,’UPDATE’,’DELETE’);

コネクタは、挿入/更新/削除操作を含む SQL_REDO 応答を処理します。後続の各問合せの開始日の値は、現行の応答の sql_timestamp 列の値に 1 秒を足した値に設定されます。

ステートメントはイベントに変換され、ソースウィンドウにインジェクトされます。スキーマのスキーマは次のフィールドで始まる必要があります。

"index*:int64,ts:stamp,sql_operation:string,sql_timestamp:stamp,schema:string,tablename:string,sql_where:string"

コネクタはこれらのフィールドを次のように完了します。

- index: ユニークな増分値
- ts: SQL_REDO を受信したタイムスタンプ
- sql_operation: 挿入/更新/削除
- sql_timestamp: SQL_REDO 応答のタイムスタンプ
- schema: コネクタに構成された logminerschemaowner
- tablename: コネクタに構成された logminertablename
- sql_ where: **ROWID** 値を除いた更新および削除ステートメントの WHERE 句の内容。挿入の場合はヌルです。

スキーマの残りの部分には、SQL_REDO の挿入/更新/削除応答で返された列の名前の文字列フィールドが含まれている必要があります。これらの列の値は、対応するソースウィンドウのスキーマフィールドにコピーされます。

更新イベントと削除イベントでは、WHERE 句の値が最初にコピーされます。更新イベントでは、WHERE 句からコピーされた値を上書きして set 句の値を次にコピーします。

Greenplum ログマイナーモードの使用

このモードでは、コネクタは最初に **cq_logminerstartdatetime** という名前のワークテーブルが存在するかどうかを検査します。テーブルが存在する場合は切り捨てられます。それ以外の場合は、単一の列(タイムゾーンのない **last_ctime timestamp(0)**)で作成されます。コネクタは、logminerstartdatetime の値を含む行をテーブルに追加します。次に、コネクタは、0 以外の行数が戻されるまで、 **queries_history** テーブルに対して 1 秒間隔で次の SQL コードを繰り返し実行します。

SELECT COUNT(1) FROM queries_history WHERE ctime > (select max(last_ctime) FROM “cq__logminerstartdatetime”) AND db = ‘logminerdbname’ AND username= ‘logminerschemaowner’ AND query_text LIKE ‘%logminertablename%’;

0 以外の行数が返されると、ワークテーブルは次のように更新されます。

INSERT INTO “cq__logminerstartdatetime” SELECT max(ctime) FROM queries_history WHERE ctime > (SELECT max(last_ctime) FROM “cq__logminerstartdatetime”);

次に、行は次のように **queries_history** テーブルからフェッチされます。

SELECT ctime, query_text FROM queries_history WHERE ctime > (SELECT max(last_ctime) FROM “cq__logminerstartdatetime” WHERE last_ctime < (SELECT max(last_ctime) FROM “cq__logminerstartdatetime”)) AND db = ‘logminerdbname’ AND username = ‘logminerschemaowner’ AND query_text LIKE ‘%logminertablename%’;

すべての行がフェッチされると、コネクタはループバックして、**queries_history** テーブル内の別のゼロ以外の行数の検索を開始します。フェッチされた行はイベントに変換され、ソースウィンドウにインジェクトされます。

ソースウィンドウのスキーマ要件およびフィールドの内容は、Oracle ログマイナーモードでの SQL_REDO レスポンス処理と同じです。

データベースアダプタの使用

関数の説明と構成の要件については、“データベースコネクタの使用”を参照してください。

サブスクライバの使用:

dfesp_db_adapter -c *connectstring* -h *url* -k sub -t *tablename* <-C [*configfilesection*]> <-E *tokenlocation* > <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-N *maxcolbinding* > <-q> <-V> <-X> <-x *commitrows* > <-y *logconfigfile* > <-Z *transportconfigfile*> <-z *commitsecs* >

パブリッシャ:

dfesp_db_adapter -c *connectstring* -h *url* -k pub <-B *logminerdbname* > <-b *blocksize* > <-C [*configfilesection*]> <-D *logminerstartdatetime* > <-E *tokenlocation* > <-e> <-G> <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxevents*> <-N *maxcolbinding* > <-O> <-Q> <-R> <-S *logminerschemaowner* > <-s *selectstatement* > <-T *logminertablename* > <-V> <-X> <-y *logconfigfile* > <-Z *transportconfigfile*>

パラメータ	説明
-B logminerdbname	queries_history テーブルを含む gpperfmon データベースを指定します。
-b blocksize	ブロックサイズを指定します。デフォルト値は 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-c connectstring	ドライバの種類と他のドライバ固有のパラメータを指定します。
-D logminerstartdatetime	Oracle または Greenplum ログマイナーモードの開始日時を指定します。次の形式を使用します。"dd-mmm-yyyy hh:mm:ss"
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e	イベントがトランザクションであることを指定します。
-G	Greenplum ログマイナーモードを有効にします。
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP パブリッシュおよびサブスクライブ標準 URL を "dfESP://host:port/project/continuousquery/window"の形式で指定します。 サブスクライバの場合は?snapshot=true |false を追加します。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに?rmretdel=true | false を追加します。

パラメータ	説明
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-k	サブスクライバ使用の場合は“sub”を、パブリッシャ使用の場合は“pub”を指定します。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用してください。
-m maxevents	パブリッシュするイベントの最大数を指定します。
-N maxcolbinding	サポートされている文字列の幅の最大値を指定します。デフォルト値は 4096 です。
-O	Oracle ログマイナーモードを使用可能にします。
-Q	パブリッシャの場合、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトを休止します。
-q	SQL 挿入、更新、または削除の実行時にエラーを無視します。
-R	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-S logminerschemaowner	Oracle または Greenplum ログマイナーモードのスキーマ所有者を指定します。
-s selectstatement	パブリッシャソースデータベースの SQL ステートメントを指定します。
-T logminertablename	Oracle または Greenplum ログマイナーモードのテーブル名を指定します。
-t tablename	サブスクライバターゲットデータベーステーブルを 指定します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-X	connectstring の pwd フィールドが暗号化されることを指定します。
-x commitrows	バッファする最小行数を指定します。
-y logconfigfile	ログ構成ファイルを指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

パラメータ	説明
-z commitsecs	未完了のコミットバッファに保持する最大秒数を指定します。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

イベントストリームプロセッサーアダプターの 使用

イベントストリームプロセッサアダプタを使用すると、ウィンドウをサブスクライブして、それを別のソースウィンドウにパブリッシュすることができます。この操作は、単一のイベントストリームプロセッサ内で行うことができます。ネットワーク上の異なるマシン上で実行される 2 つのイベントストリームプロセッサで発生する可能性が高くなります。

対応するイベントストリームプロセッサコネクタはありません。

使用法:

dfesp_esp_adapter –p *url* -s *url*

パラメータ	説明
-p url	フォームに標準 URL をパブリッシュすることを指定します。 dfESP://host:port/project/contquery/window
-s url	フォームに標準 URL をサブスクライブすることを指定します "dfESP://host:port/project/contquery/window?snapshot=true |false ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて次の行を追加します。 ?collapse=true | false"

eventblock のサイズとトランザクションの性質は、サブスクライブされたウィンドウから継承されます。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

ファイルとソケットコネクタとアダプタの使用

ファイルコネクタとソケットコネクタの 使用

ファイルコネクタとソケットコネクタの概要

ファイルコネクタとソケットコネクタは、次のデータ型をストリームするファイルまたはソケット接続のパブリッシュ操作とサブスクライブ操作の両方をサポートします。

- **バイナリ**

- **cef** (ArcSight Common Event Format、パブリッシュ操作のみサポート)
- **csv**
- **json**
- **sashdat** (サブスクライブ操作のみをサポートし、クライアントタイプのソケットコネクタとしてのみサポート)
- **syslog** (パブリッシュ操作のみサポート)
- **xml**

コネクタのファイルまたはソケットの性質は、構成された fsname の形式で指定されます。*host*: *port* 形式の名前は、ソケットコネクタです。それ以外の場合は、ファイルコネクタです。

コネクタがソケット接続を実装する場合、コネクタはパブリッシャかサブスクライバかにかかわらず、クライアントまたはサーバーとして動作します。**fsname** に *host* と *port* の両方を指定すると、コネクタはクライアントを実装します。構成されたホストとポートは、接続のサーバー側を実装するネットワークピアを指定します。しかし、*host* がブランクの場合(つまり、**fsname** が ":*port*"の形式である場合)、接続は逆になりますコネクタはサーバーを実装し、ネットワークピアはクライアントです。

ファイルコネクタとソケットコネクタを指定する場合は、次のパラメータを使用します。

表 11 ファイルコネクタとソケットコネクタの必須パラメータ

パラメータ	説明
fsname	パブリッシャの入力ファイル、サブスクライバーの出力ファイル、またはソケット接続を *host*:*port* の形式で指定します。クライアントの代わりにサーバーを実装するには、*host* ブランクのままにします。
fstype	binary | csv | xml | json | syslog | hdat | cef
type	パブリッシュするかサブスクライブするかを指定します。

表 12 サブスクライバのファイルコネクタとソケットコネクタの必須パラメータ

パラメータ	説明
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。

表 13 サブスクライバのファイルコネクタおよびソケットコネクタのオプションパラメータ

パラメータ	説明
collapse	サブスクライバの出力をパブリッシュ可能にするために、UPDATE_BLOCK イベントを更新イベントに変換します。

パラメータ	説明
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。
hdatcashostport	CAS サーバーのホストとポートを指定します。**hdat** コネクタにのみ適用されます。
hdatcaspassword	CAS サーバーのパスワードを指定します。**hdat** コネクタにのみ適用されます。
hdatcasusername	CAS サーバーのユーザー名を指定します。**hdat** コネクタにのみ適用されます。
hdatfilename	HDFS (Hadoop 分散ファイルシステム)に書き込む Objective Analysis Package Data (HDAT)ファイルの名前を指定します。fsname パラメータで指定された名前ノードを参照するときに、HDFS ブラウザに表示されるフルパスを含めます。.sashdat 拡張子は含めないでください。hdat コネクタにのみ適用され、 **hdat** コネクタに必要です。
hdatlasrhostport	SAS LASR Analytic Server のホストとポートを指定します。**hdat** コネクタにのみ適用されます。
hdatlasrkey	**tklasrkey.sh** へのパスを指定します。デフォルトでは、このファイルは **$DFESP_HOME/bin** にあります。**hdat** コネクタにのみ適用されます。
hdatmaxdatanodes	データノード接続の最大数を指定します。デフォルト値は、ネームノードによって認識されるライブデータノードの総数です。hdatnumthreads<= 1 の場合、このパラメータは無視されます。**hdat** コネクタにのみ適用されます。
hdatmaxstringlength	Objective Analysis Package Data (HDAT)ファイルの文字列フィールドの固定サイズをバイト単位で指定します。8 の倍数を指定する必要があります。文字列はスペースで 埋められます。**hdat** コネクタにのみ適用され、 hdat コネクタに必要です。 列ごとに一意の文字列長を指定するには、スペースを 使用せずにコンマ区切りの値の文字列を構成します。すべての値は 8 の倍数でなければならず、値の数はサブスクライブされたウィンドウスキーマの文字列フィールドの数と等しくなければなりません。
hdatnumthreads	データノードソケット接続へのマルチスレッド書き込みに使用されるスレッドプールのサイズを指定します。0 または 1 の値は、書き込みがシングルスレッドであり、名前ノード接続のみを使用することを示します。**hdat** コネクタにのみ適用され、 hdat コネクタに必要です。

パラメータ	説明
hdfsblocksize	Objective Analysis Package Data (HDAT)ファイルの書き込みに使用されるブロックサイズを MB 単位で指定します。**hdat** コネクタにのみ適用され、hdat コネクタに必要です。
hdfsnumreplicas	Objective Analysis Package Data (HDAT)ファイルを作成して作成された HDFS (Hadoop 分散ファイルシステム)レプリカの数を指定します。デフォルト値は 1 です。**hdat** コネクタにのみ適用されます。
header	false | true | full CSV サブスクライバの場合、カンマで区切られたフィールドを示すヘッダー行を書き込むように指定します。ヘッダー行に opcode, flags,を含めるには、値を full に設定します。
maxfilesize	サブスクライバ出力ファイルの最大サイズをバイト単位で指定します。到達すると、新しい出力ファイルが開きます。構成すると、タイムスタンプがすべての出力 filename に追加されます。このパラメータは、fstype が **hdat** 以外のソケットコネクタには適用されません。
periodicity	サブスクライバ出力ファイルが閉じられ、新しい出力ファイルが開かれる間隔を秒単位で指定します。構成すると、ファイルが開かれたときのすべての出力 filename にタイムスタンプが付加されます。このパラメータは、fstype が **hdat** 以外のソケットコネクタには適用されません。
rate	待機時間モードを有効にすると、生成された出力ファイルにこの指定レートが表示されます。
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバによって受信されたイベントブロックからすべての 削除イベントを削除することを指定します。
unbufferedoutputstreams	ファイルまたはソケットに書き込むときにバッファリングされていないストリームを作成することを指定します。デフォルトでは、ストリームはバッファされます。

表 14 パブリッシャファイルコネクタとソケットコネクタのオプションパラメータ

パラメータ	説明
addcsvflags	入力 CSV イベントに挿入するイベントの種類を指定します(カンマ)。有効な値は normal と partialupdate です。
addcsvopcode	CSV イベントを入力するための演算コードとカンマを先頭に付けます。演算コードは、publishwithupsert が有効になっていない場合は挿入です。
blocksize	パブリッシュされたイベントブロックに含めるイベントの数を指定します。デフォルト値は 1 です。
cefsyslogprefix	fstype=cef の場合、CEF イベントに syslog 接頭辞が含まれることを指定します。

パラメータ	説明
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config**(Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config**(Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
growinginputfile	増加している入力ファイルからパブリッシャが読むことができます。有効にすると、コネクタが停止するか、サーバーが接続を切断するまで、パブリッシャは入力ファイルから無期限に読み取ります。このパラメータは、ソケットコネクタには適用されません。
header	パブリッシュ操作を開始する前にスキップする入力行の数を指定します。デフォルト値は 0 です。このパラメータは、csv および syslog パブリッシュコネクタにのみ適用されます。
ignorecsvparseerrors	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、発行が続行されるように指定します。
maxevents	パブリッシュするイベントの最大数を指定します。
noautogenfield	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。
prebuffer	イベントブロックがインジェクションを実行する前にイベントブロックベクトルにバッファリングされるかどうかを制御します。デフォルト値は“false”です。有効でないか、growinginputfile ソケットコネクタ用です。
publishwithupsert	opcode=Insert の代わりに opcode=Upsert を使用してイベントをビルドします。
rate	要求された送信レートを 1 秒あたりのイベント数で指定します。
repeatcount	パブリッシュ操作を繰り返す回数を指定します。有限数のイベントをパブリッシュするパブリッシャに適用されます。デフォルト値は 0 です。
transactional	イベントブロックの種類をトランザクションに設定します。デフォルトのイベントブロックの種類は標準です。

CSV ファイルとソケットコネクタのデータ形式

CSV パブリッシャは、各行をイベントに変換します。各行の最初の 2 つの値は、演算コードフラグとイベントフラグであると予想されます。CSV ファイルの任意の行に演算コードとイベントフラグが含まれていない場合は、addcsvopcode パラメータと addcsvflags パラメータを使用します。CVS ファイルの行に、キーとして使用でき

る一意の値を持つ列が含まれていない場合は、autogen-key と noautogenfield の両方のコネクタプロパティに true を指定する必要があります。

XML ファイルとソケットコネクタのデータ形式

次の XML エレメントが有効です。

- <data>
- <events>
- <transaction>
- <event>
- <opcode>
- <flags>

さらに、イベントデータフィールド名に対応するエレメントは、イベント内に含まれるときに有効です。必須エレメントは<events>と<event>です。<events>には 1 つ以上の<event>エレメントが含まれています。これはイベントブロックを定義します。<transaction>エレメントに含まれるイベントは、type = transactional のイベントブロックに含まれます。それ以外の場合、イベントは type = normal のイベントブロックに含まれます。

単一の<data>エレメントは、常に出力 XML のイベントブロックの完全なセットをラップします。入力 XML の終了<data>エレメントは、ストリーミングされたイベントブロックの終了を示します。イベントは、<opcode>エレメントまたは<flags>エレメントで特に指定されていない限り、opcode=INSERT および flags=NORMAL の入力 XML から作成されます。

入力データの opcode エレメントの有効なデータは、次のとおりです。

opcode **タグ値**	**演算コード**
i	Insert
u	Update
p	Upsert
d	Delete
s	Safe delete

入力データの flags エレメントに有効なデータは、次のとおりです。

- **n**: イベントは標準です
- **p**: イベントは部分的な更新です

サブスクライバは、出力データに演算コードを挿入、更新、および削除するだけを書き込みます。

イベントのキー以外のフィールドは入力データには必要なく、フィールドにデータがない場合は value = NULL になります。サブスクライバは常にすべてのイベントデータフィールドを 書き込みます。

イベントフィールドには、partialupdate XML 属性を含めることができます。その値が **true** で、イベントに部分更新を指定する flags エレメントが含まれている場合、イベントは部分更新フラグで構築されます。フィールドは、ヌル値を持つ部分的な更新としてフラグが立てられます。

JSON ファイルとソケットコネクタのデータ形式

JSON パブリッシャは、入力 JSON テキストが次の形式に準拠している必要があります。

[[*event_block_n*],[*event_block_n+1*],[*event_block_n+2*],....]

最も外側の括弧は、イベントブロックの配列を定義します。角括弧の各ネストされたペアは、イベントブロックに対応するイベントの配列を定義します。この配列をイベントブロックとの間で変換するセマンティクスを"JSON メッセージングのパブリッシュ/サブスクライブ API サポート" (*SAS Event Stream Processing: パブリッシュ/サブスクライブ API*)参照してください。

JSON サブスクライバは、JSON パブリッシャが期待するのと同じ形式で JSON テキストを書き込みます。スキーマフィールドに加えて、サブスクライバは、常に値がイベント演算コードである opcode フィールドを含みます。出力 JSON の各イベントブロックはカンマと改行で区切られています。

Syslog ファイルとソケットコネクタノート

syslog ファイルとソケットコネクタは、パブリッシャ操作でのみサポートされています。Syslog イベントを収集し、ESP イベントブロックに変換し、実行中の ESP モデルの 1 つ以上のソースウィンドウにインジェクトします。

入力された syslog イベントは、syslog デーモンによって書き込まれたファイルまたは名前付きパイプから読み込まれます。デーモンによってフィルタにかけられた Syslog イベントは、コネクタによって 認識されません。名前付きパイプから読み取るときは、次の条件を満たす必要があります。

- コネクタは、デーモンと同じプロセススペースで 実行されている必要があります。
- パイプが存在する必要があります。
- デーモンは、名前付きパイプに書き込むように構成する必要があります。

growinginputfile パラメータを指定すると、ファイルまたは名前付きパイプからデータを読み込むことができます。

コネクタは、一度に 1 行ずつテキストデータを読み取り、区切り文字としてスペースを 使用して行をフィールドに解析します。

ウィンドウスキーマの最初のフィールドは、INT32 型のキーフィールドでなければなりません。対応するウィンドウスキーマの他のフィールドは、ESP_UTF8STR または ESP_DATETIME 型でなければなりません。スキーマフィールドの数は、syslog ファイルで解析されるフィールドの数を超えてはいけません。

サンプルのスキーマは次のとおりです。

```
<schema>
  <fields>
    <field name='ID' type='int32' key='true'/>
    <field name='udate' type='date'/>
    <field name='hostname' type='string'/>
   <field name='message' type='string'/>
 </fields>
</schema>
```

HDAT サブスクライブソケットコネクタノート

このコネクタは、Objective Analysis Package Data (HDAT)ファイルを SASHDAT 形式で書き込みます。このソケットコネクタは、fsname パラメータで指定された名前ノードに接続します。デフォルトの LASR ネームノードのポートは 15452 です。このポートを構成できます。接続先の適切なポートについては、SAS Hadoop プラグインのプロパティを参照してください。

SAS Hadoop プラグインは、サービスからの put を許可するように構成する必要があります。プロパティ com.sas.lasr.hadoop.service.allow.put=true を構成します。デフォルトでは、これらの put は有効になっています。このプロパティが名前ノードに加えてデータノードにも構成されていることを確認してください。

マルチスレッドで実行されている場合（hdatnumthreads パラメータ）を使用して、コネクタは、名前ノードによって戻されるすべてのデータノードへのソケット 接続をオープンします。その後、hdfsblocksize パラメータで指定されたブロックサイズを使用して、サブスクライバイベントデータをすべてのデータノードに Objective

Analysis Package Data (HDAT)ファイルパートとして書き込みます。これらのファイル部分は、ネームノードがコネクタによって指示されたときに、単一のファイルに連結されます。

コネクタは、停止時にファイル部分を自動的に連結します。また、オプションの periodicity および maxfilesize パラメータで指定されているように、定期的に停止することもあります。

デフォルトでは、ネームノードとデータノードへのソケット接続には、デフォルトのタイムアウト時間が 5 分以内に設定されています。このタイムアウトにより、サーバーは、ソケット上で 5 分間使用しないと、イベントストリーム処理コネクタを切断します。SAS Hadoop プラグインの構成プロパティ com.sas.lasr.hadoop.socket.timeout に値 0 を構成することで、タイムアウトを無効にすることをお勧めします。

SASHDAT 形式は静的行データで構成されているため、コネクタは削除イベントを無視します。更新イベントは挿入イベントと同じ扱いになります。

hdatlasrhostport または hdatcashostport パラメータを指定すると、HDAT ファイルが書き込まれ、ファイルが LASR または CAS インメモリテーブルにロードされます。periodicity パラメータまたは maxfilesize パラメータを指定すると、HDAT ファイルの各書き込みの直後にそのファイルのロードが続きます。

Common Event Format (CEF)ファイルとソケットコネクタ

このモードでは、データを ArcSight Common Event Format (CEF)でソースウィンドウにパブリッシュします。デフォルトでは、syslog 接頭辞を含まない CEF イベントを処理します。イベントにその接頭辞が含まれている場合は、パラメータ cefsyslogprefix を構成します。

ソースウィンドウのスキーマは、次のフィールドで開始する必要があります。

index*:int32,version:int32,devicevendor:string,deviceproduct:string,deviceversion:string,signatureid:string,name:string,severity:string

ただし、cefsyslogprefix が構成されている場合、スキーマは次のように開始する必要があります。

index*:int32,datetime:date,hostname:string,version:int32,devicevendor:string,deviceproduct:string,deviceversion:string,signatureid:string,name:string,severity:string

インデックスはコネクタによって 追加され、イベントキーフィールドとして 機能します。残りのフィールドは、syslog 接頭辞と必須の CEF ヘッダーフィールドです。

スキーマの残りの部分には、CEF 拡張の追加フィールドを含めることができますが、必須ではありません。それらを含めると、スキーマのフィールド名は CEF のキーと値のペアのキーと一致する必要があります。スキーマのフィールド型は、キーと値のペアの値の型と一致する必要があります。キーが一致しない場合、フィールドはヌルに設定されます。これにより、異なる拡張フィールドを持つ CEF イベントを単一のソースウィンドウにインジェクトすることができます。

CEF キー名に英数字またはアンダースコア以外の文字が含まれている場合は、対応するソースウィンドウのスキーマフィールド名のアンダースコアに 置き換える必要があります。

syslog 接頭辞の日付を適切に解析するには、dateformat パラメータを適宜構成します。たとえば、

"Jan 18 11:07:53" set dateformat to "%b %d %H:%M:%S”

ファイルとソケットアダプタの使用

ファイルとソケットアダプタは、次のデータ型をストリームするファイルまたはソケット接続のパブリッシュおよびサブスクライブ操作をサポートします。

- dfESP バイナリ
- CSV
- XML
- JSON
- syslog (パブリッシャのみ)

- HDAT (サブスクライバのみ)
- CEF (ArcSight Common Event Format)(パブリッシャのみ)

サブスクライバの使用:

dfesp_fs_adapter -f *fsname* - h *url* -k sub -t binary | csv | xml | json | hdat <-a *aggrsize* > <-B> <-C [*configfilesection*]> <-c *period* > <-d *dateformat* > <-E *tokenlocation* > <-g *gdconfig* > <-H *hdatlasrhostport* > <-j trace | debug | info | warn | error | fatal | off> <-K *hdatalasrkey* > <-l native | solace | tervela | rabbitmq | kafka><-M true | false | full> <-N *latencyblksize* > <-n> <-o *hdat_filename* > <-P *hdatcashostport*> <-q *hdat_max_data_nodes* > <-r *rate* > <-s *maxfilesize* > <-U *hdatcasusername*> <-u *hdfs_blocksize* > <-V> <-v *hdfs_numreplicas* > <-W *hdatcaspassword*> <-w *hdat_numthreads* > <-y *log_configfile* > <-z *hdat_max_stringlength* > <-Z *transportconfigfile*> <-3 *doubleprecision*>

パブリッシャの使用:

dfesp_fs_adapter -f *fsname* -h *url* -k pub -t binary | csv | xml | json | syslog | cef < -A> <-b *blocksize* ><-C [*configfilesection*]> < -D *csvfielddelimiter* > <-d *dateformat* > <-E *tokenlocation* > <-e> <-F *eventtype* > <-G *rampupsecs* > <-g *gdconfig* > <-i> <-I> <-J *repeatcount*> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxevents* > <-n > <-O> <-p > <-Q> <-R> <-r *rate* > <-S><-V> <-x *header* > <-y *log_configfile* > <-Z *transportconfigfile*>

パラメータ	説明
-A	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。
-a aggrsize	待機時間モードで、集計ブロックサイズの統計を指定します。カンマ区切りの値で値を追跡して、待機時間計算で無視する開始および終了集計ブロックの数を指定できます。
-B	ファイルまたはソケットに書き込むときにバッファリングされていないストリームを作成することを指定します。デフォルトでは、ストリームはバッファされます。
-b blocksize	ブロックサイズを設定します。デフォルト値は 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-c period	出力ファイルの時間(秒)を指定します。アクティブなファイルが閉じられ、ファイル名にタイムスタンプが付加された新しいアクティブなファイルが開きます。
-D csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。

パラメータ	説明
-e	イベントがトランザクションであることを指定します。
-F eventtype	入力 CSV イベントに挿入するイベントの種類をカンマで指定します。有効な値は"normal"と"partialupdate"です。
-f fsname	サブスクライバ出力ファイル、パブリッシャ入力ファイル、またはソケット"host:port"を指定します。サーバーを実装するには、*host* をブランクのままにしておきます。
-G rampupsecs	ゼロから、-r で要求する送信レートまで直線的に増加する秒数を指定します。デフォルトは 0 です。
-g gdconfig	保証された配信構成ファイルを指定します。
-H hdatlasrhostport	LASR 分析サーバーのホストとポートを指定します。
-h url	dfESP のパブリッシュおよびサブスクライブの標準 URL を dfESP://host:port/project/continuousquery/window の形式で指定します。 サブスクライバに次の行を追加します。?snapshot=true | false です。必要に応じて、サブスクライバに以下を追加します。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 ■ ?collapse=true | false ■ ?rmretdel=true | false
-I	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。
-i	増大するファイルから読み込みます。
-J repeatcount	パブリッシュ操作を繰り返す回数を指定します。有限数のイベントをパブリッシュするパブリッシャに適用されます。デフォルト値は 0 です。
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-K hdatlasrkey	tklasrkey.sh へのパスを指定します。デフォルトでは、このファイルは **$DFESP_HOME/bin** にあります。
-k	サブスクライバ使用の場合は sub を、パブリッシャ使用の場合は pub を指定します。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、必要なクライアント構成ファイルを使用します。これらのファイルは、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されています。
-M true | false | full	CSV サブスクライバの場合、コンマ区切りのフィールド名を示すヘッダー行を書き込みます。ヘッダー行に "opcode、flags"を含めるには "full"に設定します。

パラメータ	説明
-m maxevents	パブリッシュするイベントの最大数を指定します。
-N latencyblksize	待機時間モードでタイムスタンプを格納するために割り当てるメモリのブロックサイズ(イベント数)を指定します。追加ブロックは必要に応じて割り当てられます。-n を指定する場合は必須です。
-n	待機時間モードを指定します。
-O	CSV イベントを読み取るための演算コードとカンマを先頭に付けます。演算コードは、-R が有効になっていない場合は挿入されます。
-o hdat_filename	HDFS (Hadoop 分散ファイルシステム)に書き込むファイル名を指定します。
-P hdatcashostport	CAS サーバーのホストとポートを指定します。
-p	イベントブロックをパブリッシュする前にすべてのイベントブロックをバッファリングします。
-Q	パブリッシャの場合、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトを休止します。
-q hdat_max_datanodes	データノードの最大数を指定します。デフォルト値は、ライブデータノードの数です。
-R	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-r rate	パブリッシャの要求された送信レートを 1 秒あたりのイベント数で指定します。サブスクライバの場合、待機時間モードが有効な場合にファイルを書き込むレートを書き込みます。
-S	CEF イベントに syslog 接頭辞が含まれることを指定します。デフォルトでは、それらはしません。
-s maxfilesize	出力ファイルのボリュームをバイト単位で指定します。アクティブなファイルが閉じられ、新しいアクティブなファイルがファイル名にタイムスタンプを付けて開かれます。
-t binary | csv | xml | json | syslog | hdat | cef	ファイルシステムの種類を指定します。syslog 値と cef 値はパブリッシャにのみ有効です。hdat の値は、サブスクライバに対してのみ有効です。
-U hdatcasusername	CAS サーバーのユーザー名を指定します。
-u hdfs_blocksize	Hadoop 分散ファイルシステム(HDFS)ブロックサイズを MB で指定します。
-V	致命的なエラーが報告された場合にアダプタを再始動するように指定します。
-v hdfs_numreplicas	HDFS (Hadoop 分散ファイルシステム)の複製数を指定します。デフォルト値は 1 です。
-W hdatcaspassword	CAS サーバーのパスワードを指定します。
-w hdat_numthreads	アダプタのスレッドプールサイズを指定します。値<= 1 は、データノードが使用されないことを意味します。

パラメータ	説明
-x header	無視するヘッダー行の数を指定します。
-y logconfigfile	ログ構成ファイルを指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。
-z hdat_max_stringlength	HDAT レコードの文字列フィールドに使用する固定サイズを指定します。値は 8 の倍数でなければなりません。
-3 doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

HDAT リーダーアダプタの使用

HDAT リーダーアダプタは、Java パブリッシャの SAS Event Stream Processing クライアントをバンドルする dfx-esp-hdatreader-adapter.jar にあります。アダプタは、HDAT ファイルの各行を ESP イベントに変換し、イベントブロックをエンジンのソースウィンドウにインジェクトします。-s パラメータを指定しない限り、各イベントは挿入演算コードで作成されます。

ターゲットソースウィンドウスキーマのフィールド数は、HDAT 行データの列数と一致する必要があります。また、すべての HDAT 列型は、UTF8STR 型の ESP フィールドに対応する列を除き、数値でなければなりません。この場合、列には文字データが含まれている必要があります。

ソース HDFS (Hadoop 分散ファイルシステム)とファイルシステム内のファイル名は、必要なパラメータとしてアダプタに渡されます。クライアントターゲットプラットフォームは、環境変数 **DFESP_HDFS_JARS** を定義する必要があります。これは、Hadoop JAR ファイルの場所を指定します。

hadoop-common-*.jar、hadoop-hdfs-*.jar、*common hadoop JARs* の順に JAR ファイルをリストします。

Linux の場合:

```
$ export DFESP_HDFS_JARS=/usr/local/hadoop-2.5.0/share/hadoop/common/hadoop-common-2.5.0.jar:
/usr/local/hadoop-2.5.0/share/hadoop/hdfs/hadoop-hdfs-2.5.0.jar:
/usr/local/hadoop-2.5.0/share/hadoop/common/lib/*
```

使用法:

dfesp_hdat_publisher -f *hdfs* -i *inputfile* -u *url* <-b *blocksize* > <-c [*configfilesection*]> <-g *gdconfigfile* > <-l native | solace | tervela | rabbitmq | kafka> <-O *tokenlocation* > <-o severe | warning | info> <-Q> <-s> <-t> <-V> <-W *maxevents*> <-Z *transportconfigfile*>

パラメータ	説明
-b blocksize	イベントブロックごとのイベント数を指定します。
-c [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-f hdfs	ターゲットファイルシステムを"hdfs://host:port"の形式で指定します。core-site.xml のプロパティ fs.defaultFS に通常構成されているファイルシステムを指定します。
-g gdconfigfile	クライアントの保証された配信構成ファイルを指定します。
-i inputfile	入力された CSV ファイルを "/path/filename.csv"の形式で指定します。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、*"Java クライアント用の代替トランスポートライブラリの使用" (SAS Event Stream Processing: パブリッシュ/サブスクライブ API)*で指定された必要なクライアント構成ファイルを使用します。
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-o severe | warning | info	アプリケーションのログレベルを指定します。
-Q	すべてのイベントがソースウィンドウにインジェクトされた後、プロジェクトを休止します。 **javaadapters.config** パラメータ名: quiesceproject
-s	Insert の代わりに opcode=Upsert を使用してイベントを作成します。 **javaadapters.config** パラメータ名: publishwithupsert
-t	イベントブロックがトランザクションであることを指定します。デフォルトのイベントブロックの種類は標準です。 **javaadapters.config** パラメータ名: transactional
-u url	パブリッシュする標準 URL を "dfESP://host:port/project/continuousquery/window"の形式で指定します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。 **javaadapters.config** パラメータ名: restartonerror
-W maxevents	パブリッシュするイベントの最大数を指定します。

パラメータ	説明
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、"Kafka によるパブリッシャアダプタのフェイルオーバー" (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

HDFS (Hadoop 分散ファイルシステム)アダプタの使用

HDFS アダプタは、Java パブリッシャとサブスクライバの SAS Event Stream Processing クライアントをバンドルする dfx-esp-hdfs-adapter.jar にあります。サブスクライバクライアントは、イベントブロックを受信し、イベントを CSV 形式で HDFS ファイルに書き込む。パブリッシャクライアントは、HDFS ファイルから CSV 形式のイベントを読み取り、エンジンのソースウィンドウにイベントブロックをインジェクトします。

ターゲット HDFS およびファイルシステム内のファイル名は、両方とも必要なパラメータとしてアダプタに渡されます。

サブスクライバクライアントを使用すると、HDFS ブロックサイズとレプリカ数の値を指定できます。オプションの periodicity または maxfilesize パラメータを使用して、定期的に HDFS ファイルを書き込むようにサブスクライバクライアントを構成できます。このように構成すると、書き込まれた各ファイルのファイル名にタイムスタンプが追加されます。

増大するファイルから読み込むようにパブリッシャクライアントを構成できます。この場合、パブリッシャは無期限に実行され、HDFS ファイルサイズが増加するたびにイベントブロックをパブリッシュします。

クライアントターゲットプラットフォームの DFESP_HDFS_JARS 環境変数を定義する必要があります。この変数は、Hadoop JAR ファイルと core-site.xml ファイルの場所を指定します。

hadoop-common-*.jar、hadoop-hdfs-*.jar、*common hadoop JARs* の順に JAR ファイルをリストします。

たとえば、Linux では、コマンドラインから次のコマンドを実行します。

```
$ export DFESP_HDFS_JARS=/usr/local/hadoop-2.5.0/share/hadoop/common/hadoop-common-2.5.0.jar:
/usr/local/hadoop-2.5.0/share/hadoop/hdfs/hadoop-hdfs-2.5.0.jar:
/usr/local/hadoop-2.5.0/share/hadoop/common/lib/*:/usr/local/hadoop-2.5.0/conf
```

サブスクライバの使用:

dfesp_hdfs_subscriber -f *hdfs* -t *outputfile* -u *url* <-b *hdfsblocksize* > <-c [*configfilesection*]> <-d *dateformat* > <- g *gdconfigfile* > <-j *jaasconf*> <-k *krb5conf*> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxfilesize* > <-n *hdfsnumreplicas* > <-O *tokenlocation* > <-o severe | warning | info> <-P *opaquestringfield*> <-p *periodicity* > <-V> <-Z *transportconfigfile*>

パラメータ	説明
-b hdfsblocksize	HDFS ブロックサイズを MB で指定します。デフォルト値は 64 です。

パラメータ	説明
-c [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-f hdfs	ターゲットファイルシステムを"hdfs://host:port"の形式で指定します。ファイル core-site.xml のプロパティ fs.defaultFS で通常構成されているファイルシステムを指定します。
-g gdconfigfile	クライアントの保証された配信構成ファイルを指定します。
-j jaasconf	ローカルファイルシステム上の JAAS 構成の場所を指定します。このファイルは、Hadoop グリッドへの Kerberos 認証に使用されます。デフォルトでは、認証はありません。
-k krb5conf	ローカルファイルシステム上の Kerberos 5 構成ファイルの場所を指定します。このファイルは、Hadoop グリッドへの Kerberos 認証に必要です。デフォルト値は**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/krb5.conf** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\krb5.conf** (Windows)です。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、*"Java クライアント用の代替トランスポートライブラリの使用" (SAS Event Stream Processing: パブリッシュ/サブスクライブ API)*で指定された必要なクライアント構成ファイルを使用します。
-m maxfilesize	出力ファイルの周期性をバイト単位で指定します。
-n hdfsnumreplicas	レプリカの HDFS 数を指定します。デフォルト値は 1 です。
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-o severe | warning | info	アプリケーションのログレベルを指定します。
-p periodicity	出力ファイルの周期性を秒単位で指定します。
-P opaquestringfield	出力行を抽出するサブスクリプションウィンドウフィールドを指定します。
-t outputfile	出力 CSV ファイルを "/path/filename.csv"の形式で指定します。

パラメータ	説明
-u url	dfESP サブスクライブの標準 URL を dfESP://host:port/project/continuousquery/window?snapshot=true | false の形式で指定します。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて次の行を追加します。 ?collapse=true | false ?rmretdel=true | false
-V	致命的なエラーが報告された場合は、アダプタを再始動します。 **javaadapters.config** パラメータ名: restartonerror
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

パブリッシャ:

dfesp_hdfs_publisher -f *hdfs* –i *inputfile* -u *url* <-b *blocksize* > <-C> <-c [*configfilesection*]> <-d *dateformat* > <-e> <-F *eventtype*> <- g *gdconfigfile* > <-j *jaasconf*> <-k *krb5conf*> <-l native \| solace \| tervela \| rabbitmq \| kafka> < -m *csvfielddelimiter* > <-n > <-O *tokenlocation* > <-o severe \| warning \| info> <-Q> <-q> <-s> <-t> <-V> <-W *maxevents*> <-Z *transportconfigfile*>

パラメータ	説明
-b blocksize	イベントブロックごとのイベント数を指定します。
-C	CSV イベントを読み取るための演算コードとカンマを先頭に付けます。演算コードは、-s が有効になっていない限り Insert です。 **javaadapters.config** パラメータ名: addcsvopcode
-c [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。

パラメータ	説明
-e	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。 **javaadapters.config** パラメータ名: ignorecsvparseerrors
-F eventtype	入力 CSV イベントに挿入するイベントの種類をカンマで指定します。有効な値は "normal"と"partialupdate"です。
-f hdfs	ターゲットファイルシステムを"hdfs://host:port"の形式で指定します。
-g gdconfigfile	クライアントの保証された配信構成定ファイルを指定します。
-i inputfile	入力された CSV ファイルを "/path/filename.csv"の形式で指定します。
-j jaasconf	ローカルファイルシステム上の JAAS 構成の場所を指定します。このファイルは、Hadoop グリッドへの Kerberos 認証に使用されます。デフォルトでは、認証はありません。
-k krb5conf	ローカルファイルシステム上の Kerberos 5 構成ファイルの場所を指定します。このファイルは、Hadoop グリッドへの Kerberos 認証に必要です。デフォルト値は**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/krb5.conf** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\krb5.conf** (Windows)です。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、*"Java クライアント用の代替トランスポートライブラリの使用" (SAS Event Stream Processing: パブリッシュ/サブスクライブ API)*で指定された必要なクライアント構成ファイルを使用します。
-m csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。
-n	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。 **javaadapters.config** パラメータ名: noautogenfield
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-o severe | warning | info	アプリケーションのログレベルを指定します。
-Q	すべてのイベントがソースウィンドウにインジェクトされた 後、プロジェクトを休止します。 **javaadapters.config** パラメータ名: quiesceproject
-q	ファイルから読み取られるすべての行を不透明な文字列として扱うように指定します。ソースウィンドウスキーマは、 "index:int64,message:string" とみなされます。 **javaadapters.config** パラメータ名: opaquestring

パラメータ	説明
-s	opcode=Insert の代わりに opcode=Upsert を使用してイベントを作成します。 **javaadapters.config** パラメータ名: publishwithupsert
-t	イベントブロックがトランザクションであることを指定します。デフォルトのイベントブロックの種類は標準です。 **javaadapters.config** パラメータ名: transactional
-u url	パブリッシュする標準 URL を "dfESP://host:port/project/continuousquery/window"の形式で指定します。
-V	致命的なエラーが報告された場合は、アダプタを再始動します。 **javaadapters.config** パラメータ名: restartonerror
-W maxevents	パブリッシュするイベントの最大数を指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、*“Kafka によるパブリッシャアダプタのフェイルオーバー” (SAS Event Stream Processing: 高度なトピック)*を参照してください。

Java Message Service (JMS)アダプタの使用

Java Message Service (JMS)アダプタは、Java パブリッシャおよびサブスクライバクライアントをバンドルする dfx-esp-jms-adapter.jar にあります。どちらも JMS クライアントです。サブスクライバクライアントは、イベントブロックを受信し、JMS メッセージプロデューサです。パブリッシャクライアントは JMS メッセージコンシューマであり、イベントブロックをエンジンのソースウィンドウにインジェクトします。

サブスクライバクライアントには、イベントを格納するために使用される JMS メッセージの種類を定義するコマンドラインパラメータが必要です。パブリッシャクライアントは、次の JMS メッセージの種類を使用します。

- BytesMessage - イベントストリーム処理イベントブロック
- TextMessage - CSV または XML 形式のイベントストリーム処理イベント
- MapMessage - フィールド名と値が対応する MapMessage フィールドにマップされた Event Stream Processing イベント

TextMessage 形式の JMS メッセージには、サードパーティ製の形式でエンコードされた XML 文書を含めることができます。dfx-esp-jms-native.jar の代わりにクラスパス内の対応する JAR ファイルを使用するか、JMS アダプタスクリプトで-x スイッチを使用できます。現在、dfx-esp-jms-axeda.jar のみがサポートされています。

-E が構成されていない場合、JMS パスワードは暗号化されていない形式で渡されなければなりません。暗号化されたパスワードは、システムにインストールする必要がある OpenSSL を使用して生成することができます。SAS Event Stream Processing システムの暗号化と認証のオーバーレイをインストールするときは、付属の OpenSSL 実行可能ファイルをインストールします。コンソールで OpenSSL を使用して暗号化されたパスワードを表示するには、次のコマンドを使用します。

echo "jmspassword" | openssl enc -e -aes-256–cbc -a -salt -pass pass:"SASespJMSadapterUsedByUser=jmsuserid"

暗号化されたパスワードを *jmspassword* パラメータに指定し、-E パラメータで暗号化を有効にします。

別の XML 形式で実行する場合は、環境変数 DFESP_JAXB_JARS に JAXB JAR ファイルを指定する必要があります。https://jaxb.java.net から JAXB をダウンロードできます。

クライアントターゲットプラットフォームは、実行中の JMS ブローカ(または JMS サーバー)に接続する必要があります。環境変数 DFESP_JMS_JARS には、JMS ブローカ JAR ファイルの場所を指定する必要がありますまた、クライアントには jndi.properties ファイルが必要です。このファイルは、DFESP_JMS_PROPERTIES 環境変数で指定する必要があります。このプロパティファイルは、ブローカーに連絡して JMS 接続を作成するために必要な接続ファクトリーと、宛先 JMS トピックまたはキューを指定します。

アダプタによって参照されるデフォルトの JNDI 名をオーバーライドして、接続ファクトリとキューまたはトピックを見つけることができます。これを行うには、jndidestname および jndifactname アダプタパラメータを構成します。宛先で資格情報が必要な場合は、アダプタのコマンド行でオプションのパラメータとして指定できます。

jndi.properties ファイルのサンプルが構成ディレクトリ.に含まれています。

サブスクライバの使用:

dfesp_jms_subscriber -m BytesMessage | TextMessage | MapMessage -u *url* <-a *jndifactname* > <-c [*configfilesection*]> <-d *dateformat* > <-E> <-f *protofile* > <- g *gdconfigfile* > <-i *jmsuserid* > <-j *jndidestname* > <-L> <-l native | solace | tervela | rabbitmq | kafka> <-M> <-O *tokenlocation* > <-o severe | warning | info> <-P *opaquestringfield*> <-p *jmspassword* > <-r *protomsg* ><-V><-x native | axeda> <-Z *transportconfigfile*>

パラメータ	説明
-a jndifactname	JMS 接続ファクトリを参照する JNDI 名を指定します。デフォルト値は "ConnectionFactory"です。このオプションは、**jndi.properties の**設定を上書きします。
-c [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-E	JMS パスワードが暗号化されることを指定します。 **javaadapters.config** パラメータ名: pwdencrypted
-f protofile	Google プロトコルバッファーのサポートに使用されるメッセージを含む**.proto** ファイルを指定します。

パラメータ	説明
-g gdconfigfile	クライアントの保証された配信構成ファイルを指定します。
-i jmsuserid	JMS 宛先のユーザー ID を指定します。
-j jndidestname	JMS 宛先を検索する JNDI 名を指定します。デフォルト値は "jmsDestination"です。このオプションは、**jndi.properties** の設定を上書きします。
-L	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクショナルイベントブロックごとに 1 つのメッセージと、非トランザクションイベントブロック内のイベントごとに 1 つのメッセージです。マルチイベントメッセージでは、CSV 行は "line.separator"システムプロパティで 区切られます。 **javaadapters.config** パラメータ名: csvmsgpereventblock
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、*"Java クライアント用の代替トランスポートライブラリの使用" (SAS Event Stream Processing: パブリッシュ/サブスクライブ API)*で指定された必要なクライアント構成ファイルを使用します。
-M	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクショナルイベントブロックごとに 1 つのメッセージと、非トランザクションイベントブロック内のイベントごとに 1 つのメッセージです。マルチイベントメッセージでは、CSV 行は "line.separator"システムプロパティで 区切られます。 **javaadapters.config** パラメータ名: csvmsgperevent
-m BytesMessage | TextMessage | MapMessage	JMS メッセージの種類を指定します。
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-o severe | warning | info	アプリケーションのログレベルを指定します。
-P opaquestringfield	JMS TextMessage を抽出するサブスクリプションウィンドウフィールドを指定します。
-p jmspassword	JMS 宛先のパスワードを指定します。
-r protomsg	protofile パラメータで指定された**.proto** ファイル内のメッセージ自体を指定します。

パラメータ	説明
-u url	dfESP サブスクライブ標準 URL を dfESP://*host*:*port*/*project*/*continuousquery*/*window*?snapshot=true | false 形式で指定します。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて次の行を追加します。 ?collapse=true | false ?rmretdel=true | false
-V	致命的なエラーが報告された場合は、アダプタを再起動します。 **javaadapters.config** パラメータ名: restartonerror
-x native | axeda	TextMessage の XML 形式を指定します。メッセージが CSV 形式の場合は native を指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

パブリッシャの使用:

dfesp_jms_publisher -u *url* <-a *jndifactname* > <-b *blocksize* > <-C> <-c [*configfilesection*]> <-d *dateformat* > <-E> <-e> <-F *eventtype*> <-f *protofile* > <- g *gdconfigfile* > <-i *jmsuserid* > <-j *jndidestname* > <-l native \| solace \| tervela \| rabbitmq \| kafka> < -m *csvfielddelimiter* > <-n > <-O *tokenlocation* > <-o severe \| warning \| info> <-p *jmspassword* > <-q> <-r *protomsg* > <-s> <-t> <-V> <-x native \| axeda> <-W *maxevents*> <-Z *transportconfigfile*>

パラメータ	説明
-a jndifactname	JMS 接続ファクトリを参照する JNDI 名を指定します。デフォルト値は "ConnectionFactory"です。このオプションは、**jndi.properties** の設定を上書きします。
-b blocksize	イベントブロックごとのイベント数を指定します。
-C	演算コードとコンマの前に CSV イベントを入力するか、 "eventOpcode"フィールドがない MapMessage の種類のメッセージを入力します。演算コードは、-s が有効になっていない限り挿入です。 **javaadapters.config** パラメータ名: addcsvopcode

パラメータ	説明
-c [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-E	JMS パスワードが暗号化されることを指定します。 **javaadapters.config** パラメータ名: pwdencrypted
-e	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄されエラーが記録され、パブリッシュが続行されるように指定します。 **javaadapters.config** パラメータ名: ignorecsvparseerrors
-F eventtype	CSV イベントの入力にカンマを使用して挿入するイベントの種類、または"eventFlags"フィールドがない MapMessage の種類の受信メッセージを指定します。有効な値は "normal"と"partialupdate"です。
-f protofile	Google プロトコルバッファーのサポートに使用されるメッセージを含む**.proto** ファイルを指定します。
-g gdconfigfile	クライアントの保証された配信構成ファイルを指定します。
-i jmsuserid	JMS 宛先のユーザー ID を指定します。
-j jndidestname	JMS 宛先を検索する JNDI 名を指定します。デフォルト値は"jmsDestination"です。このオプションは、**jndi.properties** の設定を上書きします。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、*"Java クライアント用の代替トランスポートライブラリの使用" (SAS Event Stream Processing: パブリッシュ/サブスクライブ API)*で指定された必要なクライアント構成ファイルを使用します。
-o severe | warning | info	アプリケーションのログレベルを指定します。
-m csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。
-n	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。 **javaadapters.config** パラメータ名: noautogenfield
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-p jmspassword	JMS 宛先のパスワードを指定します。

パラメータ	説明
-q	受信した JMS 文字列メッセージを不透明な文字列として扱うことを指定します。ソースウィンドウのスキーマは、 "index:int64,message:string"とみなされます。 **javaadapters.config** パラメータ名: opaquestring
-r protomsg	protofile パラメータで指定された**.proto** ファイル内のメッセージ自体を指定します。
-s	Insert の代わりに opcode=Upsert を使用してイベントを作成します。 **Javaadapters.config パラメータ名:**パラメータ名: publishwithupsert
-t	イベントブロックがトランザクションであることを指定します。デフォルトのイベントブロックの種類は標準です。 **javaadapters.config** パラメータ名: transactional
-u url	パブリッシュする標準 URL を "dfESP://host:port/project/continuousquery/window"の形式で指定します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。 **javaadapters.config** パラメータ名: restartonerror
-W maxevents	パブリッシュするイベントの最大数を指定します。
-x native | axeda	TextMessage の XML 形式を指定します。メッセージが CSV 形式の場合は native を指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

Kafka コネクタとアダプタの使用

Kafka 0.9 と MapR ストリームに Kafka コネクタを使用する

Kafka コネクタは、パブリッシュおよびサブスクライブ操作のために Kafka ブローカと通信します。Apache Kafka バージョン 0.9 以上が必要です。コネクタによって 提供されるバス接続により、エンジンが個々のパブリッシュ/サブスクライブ接続を管理する必要がなくなります。コネクタは、単一のエンジンに対する同時パブリッシュ/サブスクライブ接続の大容量を実現します。

注: Kafka コネクタは、パブリッシュとサブスクライブ用に MapR コンバージドデータプラットフォームで動作することが保証されています。

Kafka サブスクライバコネクタは、サブスクライブされたイベントブロックを Kafka トピックの固定パーティションにパブリッシュします。Kafka パブリッシャコネクタは、Kafka トピックの固定パーティションからイベントブロックを読み取り、イベントブロックをソースウィンドウにインジェクトします。トピックとパーティションは、必要なコネクタ構成パラメータです。トピックのすべてのパーティションを 読み書きするには、パーティションパラメータを-1 に設定します。トピック内のすべてのパーティションを 読み書きする場合はフェイルオーバーがサポートされないため、単一のコネクタインスタンスに対してパーティションパラメータを -1 に設定することはお勧めしません。よりよい方法は、トピックの追加パーティションを読み書きするためにコネクタの追加インスタンスを実行することです。

Kafka パーティションからの消費を開始する最初のオフセットはオプションのパラメータです。デフォルト値は、使用可能な最小のオフセットです。使用可能な最小のオフセットを使用すると、トピックのメッセージストアから状態を完全に再構築できます。Kafka クラスタ管理者は、サーバーの log.retention プロパティがそのオフセットに対して適切に設定されていることを確認する必要があります。初期オフセットの他の可能な値は、最大のオフセットまたは特定のオフセット値です。

Kafka クラスタを介して送信されるイベントブロックは、バイナリ、CSV、Google の protobufs、または JSON メッセージとしてエンコードできます。コネクタはバイナリ形式との変換を行います。パラメータを使用してメッセージ形式を構成することができます。

Kafka コネクタは、ホットフェイルオーバー操作をサポートしています。ホットフェイルオーバーが有効になっている場合、コネクタは Apache Zookeeper を使用して、フェイルオーバーグループ内の他のコネクタの存在をモニターします。ホットフェイルオーバー操作が有効になっていなくても、Zookeeper C クライアントライブラリをインストールする必要があります。Zookeeper バージョン 3.4.8 を使用してコネクタをテストしました。

Kafka クライアントランタイムライブラリは、実行中のコネクタインスタンスをホストするプラットフォームにインストールする必要があります。コネクタは、LibrdKafka C および C ++ライブラリを使用します。これらのライブラリは、https://github.com/edenhill/librdkafka からダウンロードできます。SAS Event Stream Processing バージョン 4.3 以上でバージョン 0.9.3 を使用してください。SAS Event Stream Processing バージョン 4.2 以下の場合は、バージョン 0.9.2 を使用してください。ランタイム環境では、それらのライブラリへのパスを定義する必要があります(たとえば、Linux プラットフォームでは **LD_LIBRARY_PATH** を指定します)。パスには、C LibrdKafka ライブラリと C++ LibrdKafka ライブラリの両方が含まれている必要があります。パスには、Zookeeper ライブラリも含まれている必要があります。これは http://www.apache.org/dyn/closer.cgi/zookeeper/からダウンロードできます。LibrdKafka ライブラリと Zookeeper ライブラリがコネクタが動作するマシン上にビルドされていることを確認して、ビルド時に使用されていたものと同じシステムライブラリを実行時に使用できるようにします。./configure ステップで SSL サポート付きのライブラリを構築するには、オプション--LDFLAGS=-L$DFESP_HOME/lib を使用して、LibrdKafka が SAS Event Stream Processing と同じライブラリに対して構築されるようにします。

以下で説明する関連するコネクタパラメータを除き、すべての LibrdKafka 構成パラメータはコネクタによってデフォルト値のままです。ユーザーは、以下のコネクタ kafkaglobalconfig および kafkatopicconfig パラメータを使用して、librdkafka パラメータをデフォルト値から変更できます。関心のある待機時間関連のパラメータは、https://github.com/edenhill/librdkafka/wiki/How-to-decrease-message-latency に記載されています。SASL をサポートするように LibrdKafka を構成する手順については、https://github.com/edenhill/librdkafka/wiki/Using-SASL-with-librdkafka を参照してください。

対応するイベントストリーム処理のパブリッシュ/サブスクライブクライアントプラグインライブラリは、C および Java パブリッシュ/サブスクライブクライアントで使用できます。これらのライブラリーは、パブリッシュ/サブスクライブクライアントアプリケーションを処理する標準イベントストリームが、Kafka クラスタを介して Event Stream Processing Server とイベントブロックを交換することを可能にします。メッセージは、直接 TCP 接続の代わりにクラスタを介して交換されます。次の場合を除き、パブリッシュ/サブスクライブクライアントコードに変更は必要ありません。

- Kafka プラグインライブラリを指定する C_dfESPpubsubSetPubsubLib()を追加呼び出します(C クライアントの場合)
- クラスパスの前に dfx-esp-kafka-api.jar を追加します(Java クライアントの場合)

ファイル **kafka.cfg** は現在の作業ディレクトリになければなりません。このファイルは、クライアントの Kafka 構成パラメータを指定します。

クライアントプラグインライブラリは、固定トピック名を使用します。これらの名前は、ユーザーが渡したイベントストリーム処理 URL から作成されます。URL は、モデルのプロジェクト、クエリおよびウィンドウの名前を参照します。Kafka は特殊文字のサポートが限られているため、プロジェクト/クエリ/ウィンドウ名には英数字とアンダースコア、ハイフン、ドットのみを使用してください。この要件を満たすために、クライアントプラグインは渡された URL を変更して ':'を '_'に、 '/'を '。'に置き換えます。トピック名を作成するときに使用します。対応する Kafka コネクタの構成されたトピックは、クライアントによって 作成されたトピック名と一致する必要があります。

ホットフェイルオーバー操作用に構成されている場合、Zookeeper は、フェイルオーバーグループ内の他の Event Stream Processing Server で実行されている他のコネクタとコネクタのアクティブ /スタンバイ状態を調整します。したがって、アクティブコネクタに 障害が発生すると、スタンバイコネクタがアクティブになります。ホットフェイルオーバーグループ内のすべての Event Stream Processing Server は、スイッチオーバが正常に行われるように、次の条件を満たす必要があります。

- 同じモデルを実行している必要があります。
- Kafka のパブリッシャコネクタは、同じオフセットで同じ Kafka パーティションから使用を開始する必要があります。Kafka はメッセージの順序をパーティション内でのみ保証します。関連するすべてのサーバーで同一の状態を保証するには、同じオフセットを使用する必要があります。さらに、モデルによってインバウンドイベントブロックに追加されたメッセージ ID は、すべてのパブリッシャコネクタで 同期される必要があります。これらのメッセージ ID にも一貫した順序が必要です。初期オフセットは、パブリッシャコネクタに必要なパラメータで、largest、smallest または整数に設定できます。コネクタは、すべてのパブリッシャが GUID ベースの一意のグループ ID を持つ別のコンシューマグループに各コンシューマを割り当てることによって、パーティション上のすべてのメッセージを 確実に受信するようにします。
- Zookeeper 構成では、tickTime の値を 500 ミリ秒以下に指定する必要があります。これにより、Zookeeper クライアントとして機能するサブスクライバコネクタは、1 秒以内に障害が発生したクライアントを検出し、プロセスの切り替えを開始できます。

新しいサブスクライバコネクタがアクティブになると 、アウトバウンドメッセージフローは同期したままになります。同期は、スタンバイコネクタによるメッセージのバッファリングと、再開されたフローの Kafka クラスタとの調整のために発生します。具体的には、新しいアクティブなサブスクライバコネクタは 、書き込まれている Kafka パーティションの現在のオフセットを取得し、現在のオフセット-1 でメッセージを消費します。アクティブなサブスクライバコネクタは、次の処理を行います。

- このメッセージからメッセージ ID を抽出します
- 大きな ID を含むすべてのバッファリングされたメッセージをパーティションに 書き込みます
- サブスクライブしたウィンドウからメッセージの書き込みを再開します

メッセージバッファのサイズは、サブスクライバコネクタに必要なパラメータです。

Kafka コネクタは、SAS Event Stream Processing の Kafka クライアントプラグインライブラリを使用する新しく接続されたパブリッシュ/サブスクライブクライアントへのカスタムスナップショットの送信をサポートしていません。Kafka はメッセージストアであり、クライアントコンシューマの初期パーティションオフセットはクライアントプラグインライブラリで構成可能であるため、必要はありません。

コネクタが開始すると、トピック urlhostport.M (ここで、urlhostport はコネクタ構成パラメータ)がサブスクライブされます。これにより、コネクタは、その host:port の組み合わせに関連付けられたエンジン内のウィンドウをパブリッシュまたはサブスクライブする Kafka プラグインを使用して、クライアントからメタデータ要求を受信できます。urlhostport パラメータで ':'を '_'に置き換える必要があることに注意してください。メタデータ応答は、以下のいくつかの組み合わせで構成されます。

- コネクタに関連付けられたウィンドウのプロジェクト名
- コネクタに関連付けられたウィンドウのクエリ名
- コネクタに関連付けられたウィンドウのウィンドウ名
- ウィンドウのシリアライズされたスキーマ

デフォルトでは、すべての Kafka サブスクライバコネクタとクライアントトランスポートには、ブローカからのメッセージ確認が必要です。肯定応答を使用すると、ブローカの接続障害、トピックリーダーの変更、およびブローカによって通知されたプロデューサエラーを適切に処理できます。

表 15 Kafka サブスクライバコネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
kafkahostport	以下の形式で 1 つ以上の Kafka ブローカを指定します。 host:port,host:port,... 。
kafkatopic	Kafka のトピックを指定します。
kafkapartition	Kafka パーティションを指定します。トピック内のすべてのパーティションを 使用するには、-1 を指定します。ホットフェイルオーバーが有効な場合、この値はサポートされません。
kafkatype	**binary**、**CSV**、**JSON**、またはサブスクライブされたウィンドウスキーマ内の文字列フィールドの名前を指定します。**opaquestring** の場合、ソースウィンドウスキーマは index:int64,message:string とみなされます。
urlhostport	起動時にサブスクライブされるメタデータトピックの host:port フィールドを指定します。この組み合わせは、メタデータ要求をフィールドします。Kafka のトピックの命名要件を満たすには、 ':'を '_'に置き換えます。
numbufferedmsgs	スタンバイサブスクライバコネクタによってバッファされるメッセージの 最大数を指定します。超過すると、最も古いメッセージは破棄されます。コネクタがアクティブになると、バッファがフラッシュされ、メッセージ ID シーケンスを維持するためにバッファに入れられたメッセージが必要に応じてクラスタに送信されます。
snapshot	スナップショットデータを送信するかどうかを指定します。サポートされている唯一の値は false です。このサブスクライバによって送信されたメッセージを読み取っているサブスクライバクライアントは、適切な初期オフセットを Kafka パーティションに構成することによって、スナップショットを制御できます。

表 16 Kafka パブリッシャコネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
kafkahostport	以下の形式で 1 つ以上の Kafka ブローカを指定します。 host:port,host:port,... 。
kafkatopic	Kafka のトピックを指定します。
kafkapartition	Kafka パーティションを指定します。トピック内のすべてのパーティションを 使用するには、-1 を指定します。ホットフェイルオーバーが有効な場合、この値はサポートされません。
kafkatype	**binary**、**CSV**、**JSON**、または **opaquestring** を指定します。

パラメータ	説明
urlhostport	起動時にサブスクライブされるメタデータトピックの host:port フィールドを指定します。この組み合わせは、メタデータ要求をフィールドします。Kafka のトピックの命名要件を満たすには、 ':'を '_'に置き換えます。

表 17 Kafka サブスクライバコネクタのオプションパラメータ

パラメータ	説明
collapse	サブスクライバ出力をパブリッシュ可能にするために UPDATE_BLOCK イベントの変換を有効にします。
rmretdel	ウィンドウ保持ポリシによって導入されたサブスクライバによって受信されたイベントブロックからすべての 削除イベントを削除することを指定します。
hotfailover	ホットフェイルオーバーモードを有効にします。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
protofile	イベントブロックを protobuf メッセージに変換するために使用される Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
csvincludeschema	kafkatype=CSV の場合、出力 CSV データにウィンドウのシリアライズされたスキーマをいつ付加するかを指定します。有効な値は、never、once、および pereventblock です。デフォルト値は“never”です。
useclientmsgid	フェイルオーバー操作を実行し、イベントブロックからメッセージ ID を抽出するときに、エンジンが生成したメッセージ ID ではなく、クライアントが生成したメッセージ ID を使用します。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
zookeeperhostport	Zookeeper サーバーを host:port の形式で指定します。
kafkaglobalconfig	librdkafka グローバル構成値を構成するための key=value 文字列のセミコロン区切りリストを指定します。
kafkatopicconfig	librdkafka トピック構成値を構成するための key=value 文字列のセミコロン区切りリストを指定します。

パラメータ	説明
csvmsgperevent	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
csvmsgpereventblock	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

表 18 Kafka パブリッシャコネクタのオプションパラメータ

パラメータ	説明
transactional	kafkatype=CSV の場合、イベントブロックの種類をトランザクションに設定します。デフォルトのイベントブロックの種類は標準です。
blocksize	kafkatype=CSV の場合、公開されたイベントブロックに含めるイベント数を指定します。デフォルト値は 1 です。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
ignorecsvparseerrors	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。
protofile	イベントブロックを protobuf メッセージに変換するために使用される Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。
noautogenfield	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。
publishwithupsert	opcode = Insert の代わりに opcode = Upsert でイベントを構築することを指定します。

パラメータ	説明
kafkainitialoffset	Kafka のトピックとパーティションからのメッセージの使用を開始するオフセットを指定します。有効な値は、smallest、largest または整数です。デフォルト値は smallest です。
addcsvopcode	CSV イベントを書き込むための演算コードとカンマを先頭に付けます。演算コードは、publishwithupsert が有効になっていない場合は挿入です。
addcsvflags	入力 CSV イベントに挿入するイベントの種類を指定します(カンマ)。有効な値は normal と partialupdate です。
kafkaglobalconfig	librdkafka グローバル構成値を構成するための key=value 文字列のセミコロン区切りリストを指定します。
kafkatopicconfig	librdkafka トピック構成値を構成するための key=value 文字列のセミコロン区切りリストを指定します。
useclientmsgid	ソースウィンドウがディスクからディスクにリストアされている 場合、復元されたウィンドウに最大のメッセージ ID よりも小さいメッセージ ID を含む受信したバイナリイベントブロックを無視します。
maxevents	パブリッシュするイベントの最大数を指定します。

Kafka 0.9 と MapR ストリーム用の Kafka アダプタの使用

Kafka アダプタは、Kafka クラスタ上のパブリッシュおよびサブスクライブ操作をサポートします。アダプタを使用するには、LibrdKafka C、C ++、および Apache Zookeeper ライブラリをインストールする必要があります。詳細は、Apache Kafka documentation を参照してください。

注: Kafka アダプタは、パブリッシュおよびサブスクライブ用に MapR コンバージドデータプラットフォームで動作することが保証されています。

サブスクライバの使用:

dfesp_kafka_adapter -h *url* -k sub -n *numbufferedmsgs* -o *urlhostport* -p *kafkapartition* -s *kafkahostport* -t *kafkatopic* -z binary | csv | json <-C configfilesection> <-d *dateformat* > <-E tokenlocation> <-f *protofile* > <-G *kafkaglobalconfig*> <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal | off> <-L> <-l native | solace | tervela | rabbitmq | kafka> <-M> <-m *protomsg* > <-T *kafkatopicconfig*> <-V> <-w never | once | pereventblock> <-y logconfigfile> <-Z *transportconfigfile*> <-3 *doubleprecision*>

パブリッシャ:

dfesp_kafka_adapter -h *url* -k pub -o *urlhostport* -p *kafkapartition* -s *kafkahostport* -t *kafkatopic* -z binary | csv | json | opaquestring <-A> <-b *blocksize* > <-C *configfilesection* > <-D *csvfielddelimiter* > <-d dateformat> <-E *tokenlocation* > <-e > <-F *eventtype*> <-f protofile> <-G *kafkaglobalconfig*> <-g *gdconfig* > <-I> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *protomsg* > <-O> <-P *kafkainitialoffset* > <-Q> <-R> <-T *kafkatopicconfig*> <-V> <-Y *maxevents*> <-y logconfigfile> <-Z *transportconfigfile*>

パラメータ	説明
-A	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。

パラメータ	説明
-b blocksize	イベントブロックサイズを指定します。デフォルトは 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-D csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-E tokenlocation	OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。このトークンは、パブリッシュ/サブスクライブサーバーによる認証に必要です。
-e	イベントがトランザクションであることを指定します。
-F eventtype	入力 CSV イベントに挿入するイベントの種類をカンマで指定します。有効な値は "normal"と "partialupdate"です。
-f protofile	Google プロトコルバッファーのサポートに使用される**.proto** ファイルを指定します。
-G kafkaglobalconfig	librdkafka グローバル構成値を構成するためのセミコロンで区切られた "key=value"文字列のリストを指定します。
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP のパブリッシュとサブスクライブの標準 URL を dfESP://host:port/project/continuousquery/window の形式で指定します。 サブスクライバに次のものを追加します。 ?snapshot=false 。 **注:** ?snapshot=true は無効です。 必要に応じて、サブスクライバに以下を追加します。 ?collapse=true | false ?rmretdel=true
-I	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。

パラメータ	説明
-j trace | debug | info | warn | error | fatal | off	ロギングレベルを指定します。デフォルトは“warn”です。
-k sub | pub	サブスクライバの場合は sub、パブリッシャの場合は pub を指定します。
-L	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
-l native | solace | tervela | rabbitmq | kafka	パブリッシュ/サブスクライブトランスポートを指定します。デフォルトは native です。
-M	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
-m protomsg	protofile パラメータで指定された**.proto** ファイル内のメッセージ自体を指定します。
-n numbufferedmsgs	スタンバイサブスクライバコネクタによってバッファされるメッセージの最大数を指定します。
-O	CSV イベントを書き込むための演算コードとカンマを先頭に付けます。演算コードは、-R が有効になっていない場合は挿入されます。
-o urlhostport	コネクタがサブスクライブするメタデータトピック内の host:port フィールドを指定します(':'を '_'で置き換えます)。
-P kafkainitialoffset	Kafka パーティションからの消費を開始するオフセットを指定します。有効な値は、"最小"、"最大"、または整数です。デフォルトは"最小"です。
-p kafkapartition	Kafka パーティションを指定します。トピック内のすべてのパーティションを使用するには、-1 を指定します。ホットフェイルオーバーが有効な場合、この値はサポートされません。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-R	opcode=Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-s kafkahostport	以下の形式で 1 つ以上の Kafka ブローカを指定します。"host:port,host:port,..." 。
-T kafkatopicconfig	librdkafka トピック構成値を構成するためのセミコロン区切りの "key=value"文字列のリストを指定します。
-t kafkatopic	Kafka のトピックを指定します。

パラメータ	説明
-w never | once | pereventblock	rmqtype=CSV の場合、ウィンドウのシリアライズされたスキーマに出力 CSV データをいつ付加するかを指定します。デフォルト値は“never”です。
-V	致命的なエラーが報告された場合にアダプタを再起動することを指定します。
-Y maxevents	パブリッシュするイベントの最大数を指定します。
-y logconfigfile	ロギング構成ファイルを指定します。デフォルトでは、何もありません。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。
-z binary | csv | json | opaquestring	メッセージの形式を指定します。opaquestring はパブリッシャにのみ有効です。opaquestring の場合、ソースウィンドウスキーマは "index:int64,message:string"とみなされます。サブスクライバの場合、サブスクライブされたウィンドウスキーマ内の文字列フィールドの名前もサポートされます。
-3 doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

SAS LASR Analytic Server アダプタの使用

SAS LASR Analytic Server アダプタは、SAS LASR Analytic Server でのパブリッシュおよびサブスクライブ操作をサポートします。SAS LASR Analytic サーバーを認証するには、次のいずれかを実行します。

- パスワードのない SSH をサーバーに呼び出すスクリプトへのパスを使用してアダプタを構成します
- SAS LASR Analytic Server 認証サービスにアクセスするようにアダプタを構成します

注: SAS LASR Analytic Server 認証サービスにアクセスするようにアダプタを構成しない場合:

- UNIX 環境でこのアダプタを実行するには、SSH クライアントをインストールする必要があります。
- Microsoft Windows 環境でこのアダプタを実行するには、SAS Event Stream Processing システムの暗号化と認証のオーバーレイをインストールする必要があります。
- SAS LASR Analytic Server が SSH サポートまたは SAS LASR Analytic Server 認証サービスをサポートしない SMP モードで実行されている場合、LASR アダプタは SAS LASR Analytic Server と同じコンピュータシステ

ムで実行されている必要があります。アダプタを起動するときは、SAS LASR Analytic Server にアクセスする権限を持つユーザとしてログインする必要があります。

- アダプタを実行しているユーザは、SAS LASR Analytic Server を起動したユーザと同じでなければなりません。

注: LASR テーブルの数値列にマップされた日時とタイムスタンプの値は、SAS 出力形式(1960 年 1 月 1 日以降の秒数)から自動的に変換されます。

注: LASR テーブルの列名は 32 文字を超えることはできません。ウィンドウスキーマがこの制限に準拠していることを確認してください。

注: ビジュアルアナリティクスでサブスクライバアダプタによって 書き込まれたデータを視覚化するには、ビジュアルアナライザが作成された後に LASR テーブルを登録する必要があります。

サブスクライバは、com.sas.lasr.LASRPreparedAppender クラスのインスタンスを実装する LASR テーブルに行を書き込みます。アダプタがシャットダウンされるとアペンダーがフラッシュされるため、その時に LASR テーブルを追加行で更新することができます。アダプタが実行されている間、アペンダが SAS LASR Analytic Server に行を送信するタイミングを制御する 2 つの構成パラメータがあります。それは blocksize と autocommit です。これらのデフォルト値は blocksize=0 と autocommit=0 (無効)です。したがって、デフォルトでは、すべての行はリアルタイムで SAS LASR Analytic Server に送信されますが、アペンダがアダプタによって閉じられるまで、テーブルにコミットされません。サブスクライブされたウィンドウが挿入のみのイベントを生成している(更新も削除もない)場合、アペンダは、アダプタがクローズされるまで閉じられません。あるいは、autocommit を構成して、秒数または行数に基づいて LASR テーブルに行を定期的にコミットすることもできます。

SAS LASR Analytic Server アダプタが LASR 認証サービスにアクセスするように構成されている場合に-N パラメータで暗号化が有効になっていない場合、authpassword を-Z パラメータで暗号化されていない形式で指定する必要があります。パスワードの暗号化されたバージョンは、OpenSSL を使用して生成することができます。OpenSSL 実行可能ファイルは、SAS Event Stream Processing System の暗号化および認証オーバーレイインストールに含まれています。コンソールで OpenSSL を使用して暗号化されたパスワードを表示するには、次のコマンドを使用します。

echo "*authpassword*" | openssl enc -e -aes-256–cbc -a -salt -pass pass:"SASespLASRadapterUsedByUser=*authusername*"

暗号化されたパスワードを使用して、-Z パラメータに authpassword を指定し、-N パラメータで暗号化を有効にします。

サブスクライバの使用:

dfesp_lasr_adapter -H *lasrurl* -h *url1<, url2,...urlN*> -k sub -t *lasr_library_server_tag.table_name* -X *tklasrkey* <-A *autocommit* > <-a *obstoanalyze* > <-b *blocksize* > <-C [*configfilesection*]> <-d *dateformat* > <-g *gdconfigfile* > <-L native | solace | tervela | rabbitmq | kafka> <-l *loglevel* > <-N> <-n> <-O *tokenlocation* > <-S><-s *schema*> <-U *authurl* > <-V> <-Y *authusername* > <-y *transportconfigfile*> <-Z *authpassword* > <-z *bufsize* >

パブリッシャ:

dfesp_lasr_adapter -H *lasrurl* -h *url1<, url2,...urlN*> -k pub -t *table* -X *tklasrkey* <-b *blocksize* > <-C [*configfilesection*]> <-c> <-d *dateformat* > <-E> <-e> <-g *gdconfigfile* > <-L native | solace | tervela | rabbitmq | kafka> <-l *loglevel* > <-N> <-O *tokenlocation* > < -Q *action* | -q *action*> <-S> <-T *stimeout* > <-U *authurl* <-u> > <-V> <-W *maxevents*> <-Y *authusername* > <-y *transportconfigfile*> <-Z *authpassword* > <-z *bufsize* >

パラメータ	説明
-A autocommit	サーバーにコミットするオブザベーション数を指定します。値<0 は、自動コミット操作の間隔を秒単位で指定します。値> 0 は、到達した後に自動コミット操作を強制するレコードの数を指定します。0 の値は、自動コミット操作を指定しません。デフォルト値は 0 です。

パラメータ	説明
-a obstoanalyze	出力 SAS LASR Analytic Server 構造を定義するために分析するオブザベーション数を指定します。デフォルト値は 4096 です。-s を指定すると、このオプションは無視されます。
-b blocksize	サブスクライバの場合は、サーバーにフラッシュするバッファサイズ(オブザベーション数)を指定します。パブリッシャの場合は、ソースウィンドウにパブリッシュするイベントブロックサイズを指定します。デフォルトは 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-c	すべてのイベントがソースウィンドウにインジェクトされた後、プロジェクトを休止します。 **javaadapters.config** パラメータ名: quiesceproject
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-E	パブリッシャの場合、テーブルからデータをフェッチするときにキャッシュを有効にします。デフォルトでは、キャッシングは無効になっています。 **javaadapters.config** パラメータ名: fetchallcached
-e	イベントブロックがトランザクションであることを指定します。デフォルトでは、イベントブロックは正常です。 **javaadapters.config** パラメータ名: transactional
-g gdconfigfile	保証された配信構成ファイルを指定します。
-H lasrurl	SAS LASR Analytic Server URL を "host:port"の形式で指定します。
-h url	パブリッシュおよびサブスクライブ標準 URL を dfESP://host:port/project/continuousquery/window の形式で指定します。 サブスクライバに次のものを追加する必要があります。 ?snapshot=true | false 。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに以下を追加します。 ?collapse=true | false?rmretdel=true | false 。 **注:** 複数の URL を指定できます。
-k sub | pub	サブスクライブまたはパブリッシュを指定します。

パラメータ	説明
-L native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、"Java クライアント用の代替トランスポートライブラリの使用" (*SAS Event Stream Processing: パブリッシュ/サブスクライブ API*)で指定された必要なクライアント構成ファイルを使用します。
-l loglevel	Java 標準のログレベルを指定します。有効な値は SEVERE | WARNING | INFO です。デフォルト値は WARNING です。
-N	認証サービスのパスワードが暗号化されていることを指定します。 **javaadapters.config** パラメータ名: pwdencrypted
-n	新しい LASR テーブルを作成します。デフォルトでは、アダプタは LASR テーブルがすでに存在するものと見なします。 **javaadapters.config** パラメータ名: newtable
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-Q action	パブリッシャの場合、LASR テーブルからキャッシュ結果を取得するアクションを指定します。たとえば、-q "fetch field1 / to=100 where field2=2"を指定します。
-q action	パブリッシャの場合は、LASR テーブルから結果を取得するアクションを指定します。たとえば、-q "fetch field1 / to=100 where field2=2"を指定します。
-S	厳密な SAS LASR Analytic Server アダプタスキーに従いません。デフォルトは、厳密なスキーマに従います。 **javaadapters.config** パラメータ名: nostrictschema
-s schema	明示的な SAS LASR Analytic Server スキーマを *rowname1*:*sastype1*:*sasformat1*:*label1*,...*rownameN*:*sastypeN*:*sasformatN*:*labelN* の形式で指定します。 **注:** *sasformat* を省略すると、SAS 形式は行に適用されません。*label* を省略すると、ラベルは行に適用されません。以前に -a を指定した場合、このオプションは無視されます。
-T stimeout	サーバーからの各応答を待機するタイムアウト間隔(秒単位)を指定します。デフォルトでは、タイムアウト間隔はありません
-t lasr_library_server_tag.table_name	LASR テーブルのフルネームを指定します。LASR テーブルのメタデータプロパティにサーバータグを配置する必要があります。
-U authurl	SAS LASR Analytic Server 認証サービスの URL を指定します。
-u	opcode=Upsert でイベントを作成します。デフォルトでは、イベントは opcode=Insert で構築されます。 **javaadapters.config** パラメータ名: publishwithupsert
-V	致命的なエラーが報告された場合は、アダプタを再始動します。 **javaadapters.config** パラメータ名: restartonerror

パラメータ	説明
-W maxevents	パブリッシュするイベントの最大数を指定します。
-X tklasrkey	Linux の場合は **tklasrkey.sh**、Microsoft Windows の場合は **tklasrkey.bat** へのパスを指定します。デフォルトでは、これらのファイルは**$DFESP_HOME/bin** にあります。
-Y authusername	SAS LASR Analytic Server 認証サービスのユーザー名を指定します。
-y transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** native トランスポートにはトランスポート構成ファイルは必要ありません。
-Z authpassword	SAS LASR Analytic Server 認証サービスのパスワードを指定します。
-z bufsize	ソケットバッファサイズを指定します。

パブリッシャアダプタのホットフェイルオーバーの実装については、"Kafka によるパブリッシャアダプタのフェイルオーバー" (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

Modbus コネクタとアダプタの使用

Modbus コネクタの使用

Modbus コネクタは、Modbus プロトコルによる SAS Event Stream Processing との間でのパブリッシュおよびサブスクライブ操作をサポートします。

Modbus コネクタでは、4 つのオブジェクトタイプが使用されます。各オブジェクトタイプには、0 から 65,534 までのアドレスの値があります。

表 19 Modbus コネクタのオブジェクトタイプ

オブジェクトタイプ	読み取り/書き込み	値
コイル	読み取りと書き込み	8 ビット(オン/オフ)
入力	読み取り専用	8 ビット(オン/オフ)
入力レジスタ	読み取り専用	16 ビット
保持レジスタ	読み取りと書き込み	16 ビット

Modbus パブリッシャコネクタは、各オブジェクトタイプの値を Modbus サーバーから読み取り、その値を SAS Event Stream Processing にパブリッシュします。Modbus サブスクライバコネクタは、SAS Event Stream Processing からデータを読み取り、その値を Modbus にパブリッシュします。サブスクライバは、読み取り専用オブジェクトタイプのコイルと保持レジスタのみをサポートします。

Modbus コネクタは、libmodbus ライブラリを使用して Modbus サーバーと通信します。libmodbus の詳細については、http://libmodbus.org/を参照してください。

Modbus PLC Simulator を使用して、Modbus インスタンスをシミュレートできます。Modbus PLC Simulator は、Modbus プロトコルをサポートする Windows 実行可能ファイルです。このシミュレータを使用すると、Modbus オブジェクトのすべてのアドレスの現在の値を表示および変更できます。Modbus コネクタはシミュレータと通信し、libmodbus ライブラリを介して Modbus アドレスを読み書きします。シミュレータは http://www.plcsimulator.org/でダウンロードできます。

Modbus パブリッシャコネクタは Modbus サーバーをポーリングしてデータを取得し、その値を実行中の ESP サーバのソースウィンドウにパブリッシュします。Modbus コネクタからデータを受信するには、ソースウィンドウに次のスキーマが必要です。

```
type*:string,address*:int32,value:int32
```

- type は Modbus オブジェクトタイプ(コイル、入力、入力レジスタ、保持レジスタ)
- address は Modbus オブジェクトアドレス(0 から 65,534 まで)
- value は Modbus オブジェクト値(コイルと入力の場合は 0 または 1、レジスタの場合は 16 ビット値)

パブリッシャコネクタが 0 以外の値を読み取ると、p,n,holding-register,1,118 などのアップサートイベントがモデルにパブリッシュされます。パブリッシャコネクタが 0 値を読み取ると、d,n,holding-register,1,0 などの削除イベントがモデルにパブリッシュされます。

Modbus サブスクライバコネクタは、実行中の ESP サーバ上のウィンドウにサブスクライブし、データを Modbus サーバーにパブリッシュします。サブスクライブされるウィンドウには、そのスキーマの type、address および value フィールドも含める必要があります。

サブスクライバコネクタが SAS Event Stream Processing からイベントを受信すると、値がイベントからプルされ、libmodbus ライブラリを使用して Modbus に送信されます。シミュレータを実行している場合、値はすぐに更新されます。

ポーリングするデータ型とアドレスを指定することができます。複数のアドレスを指定する場合は、各アドレスをカンマで区切ります。リスト内のアイテムは、個別のアドレスまたはアドレス範囲のいずれかです。たとえば、アドレス 10 から 20 まで、33、および 44 の保持レジスタをポーリングするには、次のようにコネクタプロパティを定義します。

```
<property name='holdingRegisters'>10-20,33,44</property>
```

注: データ型を指定するがアドレスは指定しない場合、コネクタは使用可能なすべてのアドレスをスキャンします。

表 20 Modbus パブリッシャコネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。pub である必要があります。
modbus	Modbus サーバーホストを指定します。ホスト名のかわりに、host:port を指定できます。ポートが指定されていない場合、デフォルトは 502 です。

表 21 Modbus パブリッシャコネクタのオプションパラメータ

パラメータ	説明
interval	オブジェクト値データが Modbus サーバーからプルされる間隔を秒単位で指定します。デフォルトは 10 です。
coils	ポーリングするコイルアドレスを指定します。
inputs	ポーリングする入力アドレスを指定します。
inputRegisters	ポーリングする入力レジスタアドレスを指定します。
holdingRegisters	ポーリングする保持レジスタアドレスを指定します。
slaveId	Modbus 環境でのデバイススレーブ ID を指定します。

表 22 Modbus サブスクライバコネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。sub である必要があります。
modbus	Modbus サーバーホストを指定します。ホスト名のかわりに、host:port の形式でポートを指定できます。ポートが指定されていない場合、デフォルトは 502 です。
snapshot	true の場合、起動時にウィンドウからスナップショットをプルします。

表 23 Modbus サブスクライバコネクタのオプションパラメータ

パラメータ	説明
slaveId	Modbus 環境でのデバイススレーブ ID を指定します。

Modbus アダプタの使用

Modbus アダプタは、SAS Event Stream Processing と Modbus サーバー間のパブリッシュおよびサブスクライブ操作をサポートします。

パブリッシャモードでは、アダプタで Modbus からデータが読み取られ、実行中の ESP サーバー上のソースウィンドウにパブリッシュされます。Modbus パブリッシャアダプタからのデータを受信するソースウィンドウは、Modbus パブリッシャコネクタを介してイベントを受信するソースウィンドウと同じスキーマルールに従う必要があります。

サブスクライバモードでは、アダプタは実行中の ESP サーバー上のウィンドウからイベントを受信し、そのデータを Modbus にパブリッシュします。Modbus サブスクライバアダプタにデータを送信するウィンドウは、Modbus サブスクライバコネクタを介してイベントを送信するウィンドウと同じスキーマルールに従う必要があります。

サブスクライバの使用:

dfesp_modbus_adapter -h *url* -k sub -S *slaveID* -s *server* <-C *configfilesection*> <-E *tokenlocation*> <-g *gdconfig*> <-l native | solace | tervela | rabbitmq | kafka> <-V> <-Z *transportconfigfile*>

パブリッシャ:

dfesp_modbus_adapter -h *url* -k pub -S *slaveID* -s *server* <-b *blocksize*> <-C *configfilesection*> <-c *coils*> <-E *tokenlocation*> <-g *gdconfig*> <-H *holdingRegisters*> <-I *inputs*> <-i *interval*> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxevents*> <-r *inputRegisters*> <-V> <-Z *transportconfigfile*>

表 24 Modbus アダプタのパラメータ

パラメータ	説明
-b blocksize	イベントブロックサイズを指定します。デフォルトは 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-c coils	読み取る Modbus コイルを指定します。
-E tokenlocation	OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。このトークンは、パブリッシュ/サブスクライブサーバーによる認証に必要です。
-g gdconfig	保証された配信構成ファイルを指定します。
-H holdingRegisters	読み取る Modbus 保持レジスタを指定します。
-h url	dfESP のパブリッシュとサブスクライブの標準 URL を dfESP://host:port/project/continuousquery/window の形式で指定します。 サブスクライバに?snapshot=false を追加します。 **注:** ?snapshot=true は無効です。 必要に応じて、サブスクライバに以下を追加します。 ?collapse=true | false ?rmretdel=true
-I inputs	読み取る Modbus 入力を指定します。
-i interval	Modbus のポーリング間隔を秒単位で指定します。デフォルトは 10 です。
-k sub | pub	パブリッシュまたはサブスクライブを指定します。
-l native | solace | tervela | rabbitmq | kafka	パブリッシュ/サブスクライブトランスポートを指定します。デフォルトは native です。
-m maxevents	パブリッシュするイベントの最大数を指定します。デフォルトでは、最大値はありません。
-r inputRegisters	読み取る Modbus 入力レジスタを指定します。

パラメータ	説明
-S slaveID	Modbus デバイスのスレーブ ID を指定します。
-s server	Modbus サーバーのホスト名を指定します。デフォルトポートは 502 です。別のポートを指定するには、-s server:port の形式を使用します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

MQTT コネクタとアダプタの使用

MQTT コネクタの使用

MQ Telemetry Transport (MQTT)は、制約の厳しいデバイスや低帯域幅、高待機時間、信頼性の低いネットワーク用に設計されたシンプルで軽量なパブリッシュ/サブスクライブメッセージングプロトコルです。設計の原則は、ネットワーク帯域幅とデバイスリソース要件を最小限に抑えながら、信頼性とある程度の確実な配信を保証することを試みることです。この原則は、プロトコルが接続されたデバイスの新しい "machine-to-machine" (M2M)または "Internet of Things"の世界に適しています。また、帯域幅やバッテリ電力が重視されるモバイルアプリケーションにも適しています。

MQTT コネクタは、パブリッシュ操作とサブスクライブ操作の両方のために MQTT ブローカと通信し、以下の機能を提供します。

- MQTT ブローカのトピックにサブスクライブし、受信したメッセージをソースウィンドウにパブリッシュします。
- ウィンドウをサブスクライブし、受信したイベントを MQTT ブローカトピックにパブリッシュします。
- 接続が失われた場合の MQTT ブローカーへの自動再接続。
- MQTT サーバー接続の SSL/TLS v1.2 暗号化と認証
- MQTT ブローカを介して送信されるイベントは、ESP バイナリイベントブロック、CSV、JSON、Google プロトコルバッファ、および不透明な文字列メッセージ形式としてエンコードできます。
- MQTT プロトコル v3.1.1 をサポートします。

Eclipse Mosquitto Client ライブラリ v1.4.5 ランタイムライブラリは、実行中のコネクタインスタンスをホストするプラットフォームにインストールする必要があります。MQTT ブローカとの最も高速なインターフェースを提供します。このライブラリはすべてのプラットフォームで 使用でき、http://mosquitto.org/download からダウンロードできます。ランタイム環境では、それらのライブラリへのパスを定義する必要があります(たとえば、**LD_LIBRARY_PATH** を指定するなど)。

MQTT コネクタは、Windows システムではサポートされていません。

必要に応じて、mqttpassword パラメータを暗号化されたパスワードで構成できます。暗号化されたパスワードのバージョンは、システムにインストールする必要がある OpenSSL を使用して生成することができます。SAS Event Stream Processing システムの暗号化と認証のオーバーレイをインストールする場合は、付属の OpenSSL 実行可能ファイルをインストールします。コンソールで次のコマンドを使用して、OpenSSL を起動し、暗号化されたパスワードを表示します。

echo "*mqttpassword*" | openssl enc -e -aes-256-cbc -a -salt -pass pass:"SASespMQTTconnectorUsedByUser=*mqttuserid*"

暗号化されたパスワードを mqttpassword パラメータにコピーし、mqttpasswordencrypted パラメータを有効にします。

表 25 MQTT サブスクライバコネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。
mqtthost	MQTT サーバーのホスト名を指定します。
mqttclientid	MQTT クライアント ID として使用する文字列を指定します。ヌルの場合、ランダムなクライアント ID が生成されます。ヌルの場合、mqttdonotcleansession は false でなければなりません。MQTT サーバーに接続されているすべてのクライアントで 一意でなければなりません。
mqtttopic	イベントをパブリッシュする MQTT トピックとして使用する文字列を指定します。
mqttqos	要求された QoS(Quality of Service)を指定します。値は 0,1 または 2 です。
mqttmsgtype	binary、CSV、JSON、またはサブスクライブされたウィンドウスキーマ内の文字列フィールドの名前を指定します。

表 26 MQTT パブリッシャコネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
mqtthost	MQTT サーバーのホスト名を指定します。
mqttclientid	MQTT クライアント ID として使用する文字列を指定します。ヌルの場合、ランダムなクライアント ID が生成されます。ヌルの場合、mqttdonotcleansession は false でなければなりません。MQTT サーバーに接続されているすべてのクライアントで 一意でなければなりません。

パラメータ	説明
mqtttopic	MQTT サブスクリプショントピックパターンとして 使用する文字列を指定します。
mqttqos	要求された QoS(Quality of Service)を指定します。値は 0,1 または 2 です。
mqttmsgtype	binary、CSV、JSON、または opaquestring を指定します。opaquestring の場合、ソースウィンドウスキーマに 3 つのフィールドがある場合、index:int64,topic:string,message:string とみなされます。ソースウィンドウスキーマに 2 つのフィールドがある場合は、index:int64,message:string とみなされます。

表 27 MQTT パブリッシャコネクタのオプションパラメータ

パラメータ	説明
mqttuserid	MQTT サーバーとのコネクタのセッションを認証するために必要なユーザー名を指定します。
mqttpassword	mqttuserid に関連付けられたパスワードを指定します。
mqttport	MQTT サーバーポートを指定します。デフォルトは 1883 です。
mqttacceptretainedmsg	保持されたメッセージを受け取るために受け入れる場合は true に設定されます。デフォルトは“false”です。
mqttcleansession	MQTT サーバーに切断時にすべてのメッセージとサブスクリプションを消去するように指示する場合は true に設定し、それを保持するように指示するには false を設定します。デフォルトは“true”です。
mqttkeepaliveinterval	その時刻に他のメッセージが交換されていない場合、ブローカが PING メッセージをクライアントに送信するまでの秒数を指定します。デフォルトは 10 です。
publishwithupsert	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
transactional	mqttmsgtype=CSV の場合、イベントブロックの種類をトランザクションに設定します。デフォルトのイベントブロックの種類は標準です。
blocksize	mqttmsgtype=CSV の場合、公開されたイベントブロックに含めるイベント数を指定します。デフォルト値は 1 です。
ignorecsvparseerrors	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。
csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は、以下の文字です。
noautogenfield	ソースウィンドウで自動的に生成されるキーフィールドが入力イベントにないことを指定します。

パラメータ	説明
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
protofile	イベントブロックを protobuf メッセージに変換するために使用される Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
mqttssl	SSL/TLS を使用して MQTT ブローカーに接続することを指定します。デフォルトは“false”です。SSL/TLS を使用するには、SAS ESP 暗号化オーバーレイをインストールする必要があります。
mqttsslcafile	mqttssl=true の場合、PEM エンコードされたトラステッド CA 証明書ファイルを含むファイルへのパスを指定します。mqttsslcafile または mqttsslcapath のいずれかを指定する必要があります。
mqttsslcapath	mqttssl=true の場合、PEM エンコードされたトラステッド CA 証明書ファイルを含むディレクトリへのパスを指定します。このディレクトリの構成の詳細については、**mosquitto.conf** を参照してください。mqttsslcafile または mqttsslcapath のいずれかを指定する必要があります。
mqttsslcertfile	mqttssl=true の場合、このクライアントの PEM でエンコードされた証明書ファイルを含むファイルへのパスを指定します。いずれかが存在する場合は、mqttsslcertfile と mqttsslkeyfile の両方を指定する必要があります。
mqttsslkeyfile	mqttssl=true の場合、このクライアントの PEM エンコードされた秘密鍵を含むファイルへのパスを指定します。いずれかが存在する場合は、mqttsslcertfile と mqttsslkeyfile の両方を指定する必要があります。
mqttsslpassword	mqttssl=true で、鍵ファイルが暗号化されている場合は、復号化のためのパスワードを指定します。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
mqttpasswordencrypted	mqttpassword が暗号化されることを指定します。
maxevents	パブリッシュするイベントの最大数を指定します。

表 28 MQTT サブスクライバコネクタのオプションパラメータ

パラメータ	説明
mqttuserid	MQTT サーバーとのコネクタのセッションを認証するために必要なユーザー名を指定します。
mqttpassword	mqttuserid に関連付けられたパスワードを指定します。
mqttport	MQTT サーバーポートを指定します。デフォルトは 1883 です。
mqttretainmsg	パブリッシュされたメッセージを MQTT サーバーに保持するには、true に設定します。デフォルトは“false”です。
mqttdonotcleansession	すべてのメッセージとサブスクリプションを 保持するのではなく、切断した状態に保つように MQTT サーバーに指示します。デフォルトは“false”です。
mqttkeepaliveinterval	その時刻に他のメッセージが交換されていない場合、ブローカが PING メッセージをクライアントに送信するまでの秒数を指定します。デフォルトは 10 です。
mqttmsgmaxdeliveryattempts	失敗した場合にコネクタがメッセージを再送しようとする回数を指定します。デフォルトは 20 です。
mqttmsgdelaydeliveryattempts	mqttmsgmaxdeliveryattempts で指定された配信試行間の遅延をミリ秒単位で指定します。デフォルトは 500 です。
mqttmsgwaitbeforeretry	MQTT ブローカへのメッセージの再送を待つ秒数を指定します。これは、QoS> 0 のメッセージをパブリッシュする場合に適用されます。デフォルトは 20 です。
mqttmaxinflightmsg	同時に送信できる QoS 1 および 2 メッセージの数を指定します。デフォルトは 20 です。
collapse	サブスクライバ出力をパブリッシュ可能にするために UPDATE_BLOCK イベントの変換を有効にします。
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバによって受信されたイベントブロックからすべての 削除イベントを削除することを指定します。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
protofile	イベントブロックを protobuf メッセージに変換するために使用される Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。

パラメータ	説明
mqttssl	SSL/TLS を使用して MQTT ブローカーに接続することを指定します。デフォルトは“false”です。SSL/TLS を使用するには、SAS ESP 暗号化オーバーレイをインストールする必要があります。
mqttsslcafile	mqttssl=true の場合、PEM エンコードされたトラステッド CA 証明書ファイルを含むファイルへのパスを指定します。mqttsslcafile または mqttsslcapath のいずれかを指定する必要があります。
mqttsslcapath	mqttssl=true の場合、PEM エンコードされたトラステッド CA 証明書ファイルを含むディレクトリへのパスを指定します。このディレクトリの構成の詳細については、**mosquitto.conf** を参照してください。mqttsslcafile または mqttsslcapath のいずれかを指定する必要があります。
mqttsslcertfile	mqttssl=true の場合、このクライアントの PEM でエンコードされた証明書ファイルを含むファイルへのパスを指定します。いずれかが存在する場合は、mqttsslcertfile と mqttsslkeyfile の両方を指定する必要があります。
mqttsslkeyfile	mqttssl=true の場合、このクライアントの PEM エンコードされた秘密鍵を含むファイルへのパスを指定します。いずれかが存在する場合は、mqttsslcertfile と mqttsslkeyfile の両方を指定する必要があります。
mqttsslpassword	mqttssl=true で、鍵ファイルが暗号化されている場合は、復号化のためのパスワードを指定します。
csvincludeschema	never、once、または pereventblock を指定します。デフォルト値は“never”です。mqttmsgtype = CSV の場合、出力 CSV にウィンドウのシリアライズされたスキーマを付加します。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
mqttpasswordencrypted	mqttpassword が暗号化されることを指定します。
addcsvopcode	CSV イベントを入力するための演算コードとカンマを先頭に付けます。演算コードは、publishwithupsert が有効になっていない場合は挿入です。
addcsvflags	入力 CSV イベントに挿入するイベントの種類を指定します(カンマ)。有効な値は normal と partialupdate です。
csvmsgperevent	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
csvmsgpereventblock	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

MQTT アダプタの使用

MQTT アダプタは、MQTT サーバーに対するパブリッシュおよびサブスクライブ操作をサポートします。このアダプタを使用するには、Eclipse Mosquitto Client ライブラリをインストールする必要があります。

MQTT アダプタは、Windows システムではサポートされていません。

注: このコネクタへの参照をファイル**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.excluded** (Linux)から削除します。

サブスクライバの使用:

dfesp_mqtt_adapter -c *mqttclientid* -h *url* -k sub -q 0 | 1 | 2 -z binary | csv | json -s *mqtthost* -t *mqtttopic* <-a *mqttkeepaliveinterval*> <-B *mqttmsgwaitbeforeretry*> <-C *configfilesection*> <-d *dateformat*> <-E *tokenlocation*> <-f *protofile*> <-G *mqttmaxinflightmsg*> <-g *gdconfig*> <-i *mqttsslcafile*> <-j trace | debug | info | warn | error | fatal | off><-K *mqttsslkeyfile*> <-L> <-l native | solace | tervela | rabbitmq | kafka><-M> <-m *protomsg*> <-n> <-o *mqttport*> <-P *mqttsslcapath*> <-p *mqttpassword*> <-r> <-S> <-T *mqttsslcertfile*> <-u *mqttuserid*> <-V> <-v *mqttmsgmaxdeliveryattempts*> <-W *mqttsslpassword*> <-w never | once | pereventblock><-X> <-x *mqttmsgdelaydeliveryattempts*> <-y *logconfigfile*> <-Z *transportconfigfile*> <-3 *doubleprecision*>

パブリッシャの使用:

dfesp_mqtt_adapter -c *mqttclientid* -h *url* -k pub -q 0 | 1 | 2 -z binary | csv | json -s *mqtthost* -t *mqtttopic* <-A> <-a *mqttkeepaliveinterval*> <-b *blocksize*> <-C *configfilesection*> <-D csvfielddelimiter> <-d *dateformat*> <-E *tokenlocation*> <-e> <-F *eventtype*> <-f *protofile*> <-g *gdconfig*> <-I> <-i *mqttsslcafile*> <-J> <-j trace | debug | info | warn | error | fatal | off><-K *mqttsslkeyfile*> <-l native | solace | tervela | rabbitmq | kafka><-m *protomsg*> <-n> <-O> <-o *mqttport*> <-P *mqttsslcapath*> <-p *mqttpassword*> <-Q> <-R> <-S> <-T *mqttsslcertfile*><-u *mqttuserid*> <-V> <-W *mqttsslpassword*> <-X> <-Y *maxevents*> <-y *logconfigfile*> <-Z *transportconfigfile*>

パラメータ	説明
-A	ソースウィンドウで自動的に生成されるキーフィールドが入力イベントにないことを指定します。
-a mqttkeepaliveinterval	その時刻に他のメッセージが交換されていない場合、ブローカが PING メッセージをクライアントに送信するまでの秒数を指定します。デフォルトは 10 秒です。
-B mqttmsgwaitbeforeretry	MQTT ブローカへのメッセージの再送を待つ秒数を指定します(QoS> 0 のメッセージにのみ適用されます)。デフォルトは 20 です。
-b blocksize	イベントブロックサイズを指定します。デフォルトは 1 です。
-C configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-c mqttclientid	MQTT クライアント ID として使用する文字列を指定します。ヌルの場合、ランダムなクライアント ID が生成されます。ヌルの場合、mqttcleansession は true でなければなりません。MQTT サーバーに接続されているすべてのクライアントで一意でなければなりません。

パラメータ	説明
-D csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は、以下の文字です。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e	イベントがトランザクションであることを指定します。
-F eventtype	入力 CSV イベントに挿入するイベントの種類をカンマで指定します。有効な値は "normal"と "partialupdate"です。
-f protofile	Google プロトコルバッファーのサポートに使用される**.proto** ファイルを指定します。
-G mqttmaxinflightmsg	同時に送信できる QoS 1 および 2 メッセージの数を指定します。デフォルトは 20 です。
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP のパブリッシュとサブスクライブの標準 URL を dfESP://host:port/project/continuousquery/window.の形式で指定します。 サブスクライバに次のものを追加します。 ?snapshot=false 。 必要に応じて、サブスクライバに以下を追加します。 ?collapse=true | false ?rmretdel=true
-I	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。
-i mqttsslcafile	mqttssl=true の場合、PEM エンコードされたトラステッド CA 証明書ファイルを含むファイルへのパスを指定します。**mqttsslcafile** または **mqttsslcapath** のいずれかを指定する必要があります。
-J	保持されているメッセージの受信を有効にします。デフォルトは“false”です。
-j trace | debug | info | warn | error | fatal | off	ロギングレベルを指定します。デフォルトは“warn”です。
-K mqttsslkeyfile	mqttssl=true の場合、このクライアントの PEM エンコードされた秘密鍵を含むファイルへのパスを指定します。いずれかが存在する場合は、**mqttsslcertfile** と **mqttsslkeyfile** の両方を指定する必要があります。
-k sub | pub	サブスクライバの場合は sub、パブリッシャの場合は pub を指定します。
-L	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。

パラメータ	説明
-l native | solace | tervela | rabbitmq | kafka	パブリッシュ/サブスクライブトランスポートを指定します。デフォルトは native です。
-M	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
-m protomsg	protofile パラメータで指定された**.proto** ファイル内のメッセージ自体を指定します。
-n	すべてのメッセージとサブスクリプションを 切断する代わりに、MQTT サーバーにすべてのメッセージとサブスクリプションを 保持するように指示します。デフォルトは“false”です。
-O	CSV イベントを入力するための演算コードとカンマを先頭に付けます。演算コードは、-R が有効になっていない場合は挿入されます。
-o mqttport	MQTT サーバーポートを指定します。デフォルトは 1883 です。
-P mqttsslcapath	mqttssl=true の場合、PEM エンコードされたトラステッド CA 証明書ファイルを含むディレクトリへのパスを指定します。このディレクトリの構成の詳細については、**mosquitto.conf** を参照してください。**mqttsslcafile** または **mqttsslcapath** のいずれかを指定する必要があります。
-p mqttpassword	mqttuserid に関連付けられたパスワードを指定します。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-q 0 | 1 | 2	要求された QoS(Quality of Service)を指定します。
-R	opcode=Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-r	パブリッシュされたメッセージを MQTT サーバーに保持します。デフォルトは“false”です。
-S	SSL/TLS を使用して MQTT ブローカに接続します。デフォルトは“false”です。SSL/TLS を使用するには、SAS ESP 暗号化オーバーレイをインストールする必要があります。
-s mqtthost	MQTT サーバーのホスト名を指定します。
-t mqtttopic	MQTT トピックを指定します。
-T mqttsslcertfile	mqttssl=true の場合、このクライアントの PEM でエンコードされた証明書ファイルを含むファイルへのパスを指定します。いずれかが存在する場合は、**mqttsslcertfile** と **mqttsslkeyfile** の両方を指定する必要があります。
-u mqttuserid	MQTT サーバーとのセッションを認証するために必要なユーザー名を指定します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。

パラメータ	説明
-v mqttmsgmaxdeliveryattempts	失敗した場合にメッセージの再送信を試行する回数を指定します。デフォルトは 20 です。
-W mqttsslpassword	mqttssl=true で、**keyfile** が暗号化されている場合は、復号化のためのパスワードを指定します。
-w never | once | pereventblock	rmqtype=CSV の場合、ウィンドウのシリアライズされたスキーマに出力 CSV データをいつ付加するかを指定します。デフォルト値は“never”です。
-X	mqttpassword が暗号化されることを指定します。
-x mqttmsgdelaydeliveryattempts	*mqttmsgmaxdeliveryattempts* で指定された配信試行間の遅延をミリ秒単位で指定します。デフォルトは 500 です。
-Y maxevents	パブリッシュするイベントの最大数を指定します。
-y logconfigfile	ロギング構成ファイルを指定します。デフォルトでは、何もありません。
-z binary | csv | json | opaquestring	メッセージの形式を指定します。opaquestring はパブリッシャにのみ有効です。opaquestring の場合、ソースウィンドウスキーマに 3 つのフィールドがある場合、 "index:int64,topic:string,message:string"とみなされます。ソースウィンドウスキーマに 2 つのフィールドがある場合は、"index:int64,message:string"とみなされます。サブスクライバの場合、サブスクライブされたウィンドウスキーマ内の文字列フィールドの名前もサポートされます。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。
-3 doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

Nurego コネクタの使用

Nurego コネクタはサブスクライバ専用で、測定ウィンドウ.での使用を目的としています。測定ウィンドウにサブスクライブすると、コネクタは各サブスクリプションイベントごとに REST 要求を作成します。HTTPS プロトコルを使用して要求を Nurego サーバーに転送します。

Nurego コネクタの主な使用例は、Nurego サーバーに使用情報(処理されたイベント数)を定期的に提供することです。REST 要求形式は Nurego 固有のものなので、このコネクタは一般用ではありません。

必須構成パラメータ serviceurl を指定して REST 要求を入力する Nurego サービスの URL を指定します。serviceurl は、単一使用報告を受け入れる Nurego REST API エンドポイントに設定する必要があります。https://api.nurego.com/v1/subscriptions/*subscription_id*/usage

REST 要求の JSON ペイロードの形式と内容は次のとおりです。

```
{
  “provider”: “cloud-foundry”,
  “feature_id”: “num_events”,
  “amount”: “<number of events>”,
  “usage_date”: “<usage_date>”
}
```

コネクタは、測定ウィンドウによって生成されたすべてのイベントに対してこの JSON 要求を作成して送信します。(デフォルトでは、測定ウィンドウは 5 秒ごとにイベントを生成します)。計量間隔が 30 秒を超える値に再構成された場合、Nurego コネクタはエラーをログに記録し、起動時に停止します。

表 29 Nurego コネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。
serviceurl	ターゲット Nurego REST サービスの URL を指定します。
certificate	Nurego REST サービスへの HTTPS 接続を確立するために使用されるクライアント証明書の絶対パスとファイル名を指定します。
username	Nurego への新しいトークンの要求で使用するユーザー名を指定します。
password	Nurego への新しいトークンの要求で使用するパスワードを指定します。
instanceid	Nurego への新しいトークンへの要求で使用するインスタンス ID を指定します。

表 30 Nurego コネクタのオプションパラメータ

パラメータ	説明
certpassword	証明書に構成されているクライアント証明書に関連付けられているパスワードを指定します。
collapse	サブスクライバ出力をパブリッシュ可能にするために UPDATE_BLOCK イベントの変換を有効にします。
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバによって受信されたイベントブロックからすべての削除イベントを削除することを指定します。

configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。

OPC-UA コネクタとアダプタの使用

OPC-UA コネクタの使用

OPC-UA コネクタは、OPC-UA ノードの固定セットの値属性に対するパブリッシュおよびサブスクライブ操作のために OPC-UA サーバーと通信します。OPC-UA サーバーのエンドポイントは、必要なコネクタ構成パラメータです。

OPC-UA ノードのセットは、OPC-UA サーバー内の共通の名前空間に属していなければなりません。デフォルトでは、コネクタは名前空間 index = 0 の名前空間を使用します。オプションのコネクタパラメータを構成して、異なる名前空間 URI を指定することができます。異なるサーバー名空間内の追加のノードにアクセスするには、別のソースウィンドウに接続されたコネクタの別個のインスタンスを実行する必要があります。

コネクタによってアクセスされる OPC-UA ノードのセットは、コネクタに接続されたウィンドウのスキーマによって定義されます。デフォルトでは、ウィンドウスキーマのフィールド名は、構成済みの OPC-UA サーバーネームスペース内の OPC-UA ノードのノード ID を指定する必要があります。コネクタで構成する適切なノード ID は、ターゲット OPC-UA サーバーに対して標準の OPC-UA ブラウズツールを使用して見つけることができます。

または、各ウィンドウフィールド名をノード ID にマップするオプションのコネクタパラメータを指定することもできます。その場合、ウインドウフィールドは必要に応じて名前を付けることができます。

ノード ID 形式は、ノード識別子の種類とノード識別子を s_MyTemperature の形式で含む固定表記法です。最初の文字は識別子の種類で、's' (String)、'i' (Integer)、'g' (GUID)、または'b' (Opaque)でなければなりません。アンダースコアの後には識別子文字列が続きます。種類が Opaque の場合、文字列は base64 でエンコードされている必要があります。

ウィンドウスキーマのフィールドの種類は、対応する OPC-UA ノードのタイプと一致する必要があります。サポートされているマッピングは次のとおりです。

表 31 パブリッシャ向けにサポートされているマッピング:

ESP の種類	OPC -UA の種類
ESP_INT32	INT32 または UINT32 または INT16 または UINT16 または BOOLEAN
ESP_INT64	INT64 または UINT64 または INT32 または UINT32 または INT16 または UINT16 または BOOLEAN
ESP_DOUBLE	DOUBLE または FLOAT または BOOLEAN
ESP_UTF8STR	STRING
ESP_DATETIME	DATETIME

ESP の種類	OPC -UA の種類
ESP_TIMESTAMP	DATETIME
ESP_MONEY	サポートされていません

表 32 サブスクライバでサポートされているマッピング:

ESP の種類	OPC -UA の種類
ESP_INT32	INT32 または UINT32
ESP_INT64	INT64 または UINT64
ESP_DOUBLE	DOUBLE
ESP_UTF8STR	STRING
ESP_DATETIME	DATETIME
ESP_TIMESTAMP	DATETIME
ESP_MONEY	サポートされていません

OPC-UA パブリッシャコネクタには、ソースウィンドウスキーマの前面に 1 つの追加フィールドが必要です。この追加フィールドは、パブリッシュされたすべてのイベントでコネクタによってインクリメントされる 64 ビット整数です。他のフィールドが適切でない場合は、このフィールドをウィンドウキーフィールドとして 使用できます。

パブリッシャは、ソースウィンドウスキーマ内の OPC-UA ノードの値属性の値が変更されたときにイベントをパブリッシュします。ソースウィンドウに他の OPC-UA ノードを表す追加フィールドが含まれている場合、これらのフィールドの値は変更されません。デフォルトでは、アップサートを代わりに使用するようにコネクタを構成しないかぎり、イベントに挿入演算コードが含まれます。

また、この間隔の各有効期限がソースウィンドウ内のすべての OPC-UA ノードの現在値の取得を生成し、これらの値を含むイベントが公開される間隔を秒単位で指定するオプションのパブリッシャパラメータを構成することもできます。

OPC-UA サブスクライバコネクタは、サブスクライブされたウィンドウのイベントが生成されると、サブスクライブされたイベントのすべてのフィールドの 値を対応する OPC-UA ノードの値属性に書き込みます。挿入と更新を除くすべての 演算コードを無視します。

表 33 OPC-UA コネクタの必須パラメータ

パラメータ	説明
type	パブリッシュまたはサブスクライブを指定します。
opcuaendpoint	OPC-UA サーバーのエンドポイント（opc.tcp://の後の部分のみ）を指定します。

表 34 パブリッシャ OPC-UA コネクタのオプションパラメータ

パラメータ	説明
opcuanamespaceuri	OPC-UA サーバーの名前空間 URI を指定します。デフォルトは、index=0 の名前空間です。
opcuausername	OPC-UA ユーザー名を指定します。デフォルトでは、何もありません。
opcuapassword	OPC-UA パスワードを指定します。デフォルトでは、何もありません。
opcuanodeids	ESP ウィンドウのスキーマフィールドにマップするノード ID のコンマ区切りリストを <identifier type>_<identifier>と入力します。リストのサイズは、サブスクライブしたウィンドウのスキーマ-1 のフィールドの数と等しくなければなりません。ウィンドウのフィールド名はデフォルトでノード ID 形式になっています。
publishinterval	ソースウィンドウスキーマ内のすべてのノードの現在の値がパブリッシュされる間隔を秒単位で指定します。デフォルトでは、1 つ以上の値が変更されたときにパブリッシュします。
transactional	イベントブロックの種類をトランザクションに設定します。デフォルト値は“normal”です。
blocksize	パブリッシュされたイベントブロックに含めるイベントの数を指定します。デフォルト値は 1 です。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
publishwithupsert	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
maxevents	パブリッシュするイベントの最大数を指定します。

表 35 サブスクライバ OPC-UA コネクタのオプションパラメータ

パラメータ	説明
opcuanamespaceuri	OPC-UA サーバーの名前空間 URI を指定します。デフォルトは、index=0 の名前空間です。
opcuausername	OPC-UA ユーザー名を指定します。デフォルトでは、何もありません。
opcuapassword	OPC-UA パスワードを指定します。デフォルトでは、何もありません。

パラメータ	説明
opcuanodeids	ESP ウィンドウのスキーマフィールドにマップするノード ID のコンマ区切りリストを <identifier type>_<identifier>と入力します。リストサイズは、サブスクライブしたウィンドウスキーマのフィールド数と等しくなければなりません。ウィンドウのフィールド名はデフォルトでノード ID 形式になっています。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。

OPC-UA アダプタの使用

OPC-UA パブリッシャおよびサブスクライバアダプタと OPC-UA パブリッシャおよびサブスクライバコネクタは、一部のアダプタのみのオプションパラメータを除き、構成パラメータを共有します。

サブスクライバの使用:

dfesp_opcua_adapter -h *url* -k sub -s *opcuaendpoint* <-C *configfilesection*> <-E *tokenlocation*> <-g *gdconfig*> <-i *opcuanodeids*> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-n *opcuanamespaceuri*> <-p *opcuapassword*> <-u *opcuausername*> <-V> <-y *logconfigfile*> <-Z *transportconfigfile*>

パブリッシャ:

dfesp_opcua_adapter -h *url* -k pub -s *opcuaendpoint* <-b *blocksize*> <-C *configfilesection*> <-E *tokenlocation*> <-e > <-g *gdconfig*> <-i *opcuanodeids*> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxevents*> <-n *opcuanamespaceuri*> <-p *opcuapassword*> <-Q> <-R> <-u *opcuausername*> <-V> <-v *publishinterval*> <-y *logconfigfile*> <-Z *transportconfigfile*>

パラメータ	説明
-b blocksize	イベントブロックサイズを指定します。デフォルトは 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-E tokenlocation	OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。このトークンは、パブリッシュ/サブスクライブサーバーによる認証に必要です。デフォルトでは、何もありません。
-e	イベントがトランザクションであることを指定します。
-g gdconfig	保証された配信構成ファイルを指定します。

パラメータ	説明
-h url	dfESP のパブリッシュとサブスクライブの標準 URL を dfESP://host:port/project/continuousquery/window の形式で指定します。 サブスクライバに次のものを追加します。 ?snapshot=false 。 **注:** ?snapshot=true は無効です。
-i opcuanodeids	<identifier type>_<identifier>の形式で、ESP ウィンドウのスキーマフィールドにマップするノード ID のコンマ区切りリストを指定します。の形式で、ESP ウィンドウのスキーマフィールドにマップするノード ID のコンマ区切りリストを指定します。デフォルトでは、ウィンドウフィールド名はノード ID 形式であると想定されています。
-j trace | debug | info | warn | error | fatal | off	ロギングレベルを指定します。デフォルトは“warn”です。
-k sub | pub	サブスクライバの場合は sub、パブリッシャの場合は pub を指定します。
-l native| solace | tervela | rabbitmq | kafka	パブリッシュ/サブスクライブトランスポートを指定します。デフォルトは native です。
-m maxevents	パブリッシュするイベントの最大数を指定します。
-n opcuanamespaceuri	OPC-UA サーバーの名前空間 URI を指定します。デフォルトは、index=0 の名前空間です。
-p opcuapassword	OPC-UA パスワードを指定します。デフォルトでは、何もありません。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-R	opcode=Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-s opcuaendpoint	opc.tcp://接頭辞を付けずに、OPC-UA サーバーのエンドポイントを指定します。
-u opcuausername	OPC-UA ユーザー名を指定します。デフォルトでは、何もありません。
-V	致命的なエラーの後にアダプタを再始動します。
–v publishinterval	すべての現在のノード ID 値がソースウィンドウにパブリッシュされる間隔を秒単位で指定します。デフォルトでは、値は変更されたときにパブリッシュされます。
-y logconfigfile	ロギング構成ファイルを指定します。デフォルトでは、何もありません。

パラメータ	説明
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 。-l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、*"Kafka によるパブリッシャアダプタのフェイルオーバー" (SAS Event Stream Processing: 高度なトピック)*を参照してください。

PI コネクタとアダプタの使用

PI コネクタの使用

PI コネクタは、PI アセットフレームワーク(AF)サーバーのパブリッシュおよびサブスクライブ操作をサポートします。モデルは、AF 階層で AF エレメントが所有する AF 属性に関連付けられた値を保持するための固定スキーマのウィンドウを実装する必要があります。AF データ参照は、これらのエレメントの PI ポイントとして定義することができます。

注: PI コネクタのサポートは、64 ビット Microsoft Windows プラットフォームでのみ使用できます。

OSIsoft の PI アセットフレームワーク(PI AF)クライアントは、コネクタの実行中のインスタンスをホストする Microsoft Windows プラットフォームにインストールする必要があります。コネクタは、ターゲットプラットフォームに.NET 4.0 がインストールされている OSIsoft.AFSDK.DLL パブリックアセンブリをロードします。ランタイム環境では、OSIsoft.AFSDK.dll へのパスを定義する必要があります。

注: 使用する前に、このコネクタへの参照をファイル**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.excluded** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.excluded** (Windows)から削除します。

PI コネクタは、**$DFESP_HOME/bin/plugins** ディレクトリにある esp_pi_afsdk52 DLL もロードします。ランタイム環境でもこの DLL へのパスが定義されていることを確認してください。

コネクタに関連付けられているウィンドウでは、次のスキーマを使用する必要があります。

ID*:int32,*elementindex*:string,*element*:string,*attribute*:string,*value*:*variable*:,
timestamp:*stamp*,*status*:string

次のスキーマ値を使用します。

スキーマ値	説明
elementindex	コネクタ *afelement* 構成パラメータの値。AF エレメント名または AF エレメントテンプレート名を指定します。
エレメント	*値*に関連付けられた AF エレメントの名前。
属性	AF *エレメント*が所有する AF 属性の名前。

スキーマ値	説明
値	AF *属性*の値。
タイムスタンプ	*値*に関連付けられたタイムスタンプ。
ステータス	*値*に関連付けられたステータス。

注: *値*フィールドのタイプは、AF 階層で定義された AF 属性の種類によって決まります。

イベントストリームプロセッサのデータ型の AF 属性への次のマッピングがサポートされています。

イベントストリームプロセッサのデータ型	AF 属性
ESP_DOUBLE	TypeCode :: Single TypeCode :: Double
ESP_INT32	TypeCode :: Byte TypeCode :: SByte TypeCode :: Char TypeCode :: Int16 TypeCode :: UInt16 TypeCode :: Int32 TypeCode :: UInt32
ESP_INT64	TypeCode :: Int64 TypeCode :: UInt64
ESP_UTF8STR	TypeCode :: String
ESP_TIMESTAMP	TypeCode :: DateTime

属性に TypeCode::Object がある場合、接続されているのは、基になる PI ポイントの種類を使用します。有効なイベントストリーム処理データ型から PI ポイントへのマッピングは、次のとおりです。

イベントストリームプロセッサのデータ型	PI ポイント
ESP_INT32	PIPointType :: Int16 PIPointType :: Int32
ESP_DOUBLE	PIPointType :: Float16 PIPointType :: Float32
ESP_TIMESTAMP	PIPointType ::タイムスタンプ
ESP_UTF8STR	PIPointType :: String

PI コネクタでは、次のパラメータを使用します。

表 36 PI サブスクライバコネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
afelement	AF エレメントまたはエレメントプレート名を指定します。ワイルドカードがサポートされています。
iselementtemplate	afelement パラメータがエレメントテンプレート名であるかどうかを指定します。デフォルトでは、afelement パラメータはエレメント名を指定します。有効な値は TRUE または FALSE です。
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。

表 37 PI パブリッシャコネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
afelement	AF エレメントまたはエレメントプレート名を指定します。ワイルドカードがサポートされています。
iselementtemplate	afelement パラメータがエレメントテンプレート名であることを指定します。デフォルトでは、afelement パラメータはエレメント名を指定します。

表 38 PI サブスクライバコネクタのオプションパラメータ

パラメータ	説明
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバが受信したイベントブロックからすべての削除イベントを削除します。
pisystem	PI システムを指定します。デフォルトは、PI AF クライアントで構成されている PI システムです。
afdatabase	AF データベースを指定します。デフォルトは、PI AF クライアントで構成されている AF データベースです。
afrootelement	パラメータ afelement を検索する AF 階層内のルートエレメントを指定します。デフォルトは、AF データベースの最上位エレメントです。
afattribute	エレメント内の特定の属性を指定します。デフォルトはエレメント内のすべての属性です。

パラメータ	説明
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。

表 39 PI パブリッシャコネクタのオプションパラメータ

パラメータ	説明
blocksize	パブリッシュされたイベントブロックに含めるイベントの数を指定します。デフォルト値は 1 です。
transactional	イベントブロックの種類をトランザクションに設定します。デフォルトのイベントブロックの種類は標準です。
pisystem	PI システムを指定します。デフォルトは、PI AF クライアントで構成されている PI システムです。
afdatabase	AF データベースを指定します。デフォルトは、PI AF クライアントで構成されている AF データベースです。
afrootelement	パラメータ afelement を検索する AF 階層内のルートエレメントを指定します。デフォルトは、AF データベースの最上位エレメントです。
afattribute	エレメント内の特定の属性を指定します。デフォルトはエレメント内のすべての属性です。
archivetimestamp	PI システムに接続するときに、指定したタイムスタンプ以降のすべてのアーカイブされた値をパブリッシュすることを指定します。デフォルトでは、新しい値のみをパブリッシュします。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
publishwithupsert	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
maxevents	パブリッシュするイベントの最大数を指定します。

PI アダプタの使用

PI アダプタは、PI 資産フレームワーク(PI)サーバーに対してパブリッシュおよびサブスクライブ操作をサポートします。アダプタを使用するには、OS AFoft から PI AF クライアントをインストールする必要があります。

注: PI アダプタのサポートは、64 ビット Microsoft Windows プラットフォームでのみ使用できます。

サブスクライバの使用:

dfesp_pi_adapter -h *url* -k sub -s *afelement* <-a *afattribute* > <-C [*configfilesection*]> <-d *afdatabase* > <-E *tokenlocation* > <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal> <-l native | solace | tervela | rabbitmq | kafka> <-p *pisystem* > <-r *afrootelement* > <-t> <-V> <-y *logconfigfile* > <-Z *transportconfigfile*>

パブリッシャ:

dfesp_pi_adapter -h *url* -k pub -s *afelement* <-a *afattribute* > <-b *blocksize* > <-C [*configfilesection*]> <-c> <-d *afdatabase* > <-E *tokenlocation* > <-e> <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxevents*> <-p *pisystem* > <-Q> <-R> <-r *afrootelement* > <-t> <-V> <-y *logconfigfile* > <-Z *transportconfigfile*>

パラメータ	説明
-a afattribute	*afelement* 内の属性を指定し、他の属性は無視します。
-b blocksize	ブロックサイズを指定します。デフォルト値は 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-c archivetimestamp	AF サーバーに接続するときに、指定されたタイムスタンプ以降のすべてのアーカイブされた値を取得するように指定します。
-d afdatabase	AF データベースを指定します。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e	イベントがトランザクションであることを指定します。
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP のパブリッシュとサブスクライブの標準 URL を dfESP://host:port/project/continuousquery/window の形式で指定します。 サブスクライバに次のものを追加します。?snapshot=true | false。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに以下を追加します。 ?collapse=true | false?rmretdel=true | false。
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-k	サブスクライバ使用の場合は sub を、パブリッシャ使用の場合は pub を指定します。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。

パラメータ	説明
-m maxevents	パブリッシュするイベントの最大数を指定します。
-p pisystem	PI システムを指定します。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-R	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-r afrootelement	*afelement* を検索する AF 階層内のルートエレメントを指定します。
-s afelement	AF エレメントまたはエレメントテンプレート名を指定します。ワイルドカードがサポートされています。
-t	*afelement* がエレメントテンプレートの名前であることを指定します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-y logconfigfile	ログ構成ファイルを指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

プロジェクトパブリッシュコネクタの使用

プロジェクトパブリッシュコネクタを使用して、イベントストリーム処理モデル内の別のプロジェクトのウィンドウによって生成されたイベントブロックをサブスクライブします。コネクタは、そのウィンドウのイベントブロックを受け取り、それを関連付けられたウィンドウにインジェクトします。ウィンドウスキーマは同一でなければなりません。ソースプロジェクトが停止すると、パブリッシャへのイベントブロックのフローは停止します。

プロジェクトパブリッシュコネクタには、メモリにイベントの参照コピーが 1 つだけ存在するように、イベントの参照カウント付きコピーがあります。

表 40 プロジェクトパブリッシュコネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
srcproject	ソースプロジェクトの名前を指定します。
srccontinuousquery	ソースの連続クエリの名前を指定します。
srcwindow	ソースウィンドウの名前を指定します。

表 41 プロジェクトパブリッシュコネクタのオプションパラメータ

パラメータ	説明
maxevents	パブリッシュするイベントの最大数を指定します。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。

Pylon パブリッシャコネクタとアダプタの使用

Pylon パブリッシャコネクタの使用

Pylon コネクタは Basler GigE カメラと通信し、キャプチャされたフレームをソースウィンドウに連続的にパブリッシュします。カメラには既知の固定 IP アドレスが必要です。また、接続されているイーサネットネットワークケーブルで、パワーオーバーイーサネットを使用して電力を供給する必要があります。

カメラに IP 構成がないか、IP アドレスが不明な場合は、Pylon IP Configurator を実行する必要があります。Pylon IP Configurator は、https://www.baslerweb.com/en/support/downloads/software-downloads/からダウンロードできる Basler Pylon Camera Software Suite 内にあります。カメラを検出するには、ソフトウェアを実行しているマシンと同じサブネットに接続する必要があります。その後、その IP アドレスをコネクタ構成にコピーして、コネクタを実行します。カメラの IP アドレスが構成されていない場合、コネクタはローカルサブネットで最初に検出されたカメラに接続します。カメラの IP アドレスなしのコネクタ実行は、テストでは役立つ場合がありますが、プロダクション配置ではお勧めできません。

パフォーマンスを最適化するには、ジャンボフレームをサポートするために、カメラとコネクタ実行マシン間のすべてのネットワークインターフェイスを 構成します。必ず同じ MTU 値で camerapacketsize コネクタパラメータを構成してください。

コネクタが受信したフレームデータは圧縮解除され、ESP_BINARY 型のソースウィンドウフィールドにコピーされます。ソースウィンドウのスキーマは、64 ビット整数の後にバイナリ BLOB フィールドを続けて構成する必要があります(例: id*:int64,frame:blob)。整数フィールド id は、パブリッシュされたイベントごとにコネクタによって増分され、ウィンドウキーフィールドとして機能します。

デフォルトでは、フレームはカメラによる提供が可能な範囲でできるだけ速くパブリッシュされますが、ユーザーはより低い固定速度でフレームをパブリッシュするようにコネクタを構成できます。ピクセル形式や関心領域の幅と高さなどの基本フレームパラメータは、コネクタ構成パラメータで構成可能です。

カメラ構成パラメータの完全なセットへのフルアクセスは、前述の Basler Pylon Camera Software Suite でのみ使用できます。このユーティリティでは、カメラの構成をファイルに保存することができ、そのファイルへのパスをコネクタで構成して、有効なカメラ構成でカメラを実行できるようにします。

複数のコネクタインスタンスを開始して、異なるカメラから同時にフレームをキャプチャし、それらをソースウィンドウの任意の組み合わせにパブリッシュすることができます。

注: 実行中のコネクタインスタンスをホストするプラットフォームに Pylon ランタイムライブラリをインストールする必要があります。これらのライブラリは、Pylon Camera Software Suite インストールの一部としてインストールされ、同じ**/pylon***ディレクトリにあります。ランタイム環境では、それらのライブラリへのパスを定義する必要があります(たとえば、Linux プラットフォームでは LD_LIBRARY_PATH を指定します)。

表 42 Pylon コネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。pub である必要があります。

表 43 Pylon コネクタのオプションパラメータ

パラメータ	説明
cameraipaddress	カメラの IP アドレスを指定します。デフォルト値は、ローカルサブネットで検出された最初のカメラのアドレスです。
maxnumframes	パブリッシュするフレームの最大数を指定します。デフォルトでは、最大値はありません。
maxframerate	パブリッシュする 1 秒あたりの最大フレーム数を指定します。デフォルト値は、フレームをカメラから受信する速度です。
camerafeaturesfile	ロードする Pylon 機能ストリーム(*.pfs)構成ファイルを指定します。デフォルトでは、現在のカメラ構成を変更せずに使用します。
camerawidth	関心領域の幅を指定します。デフォルト値は、現在のカメラ構成の値です。
cameraheight	関心領域の高さを指定します。デフォルト値は、現在のカメラ構成の値です。
camerapixelformat	画像のピクセル形式を指定します。デフォルト値は、現在のカメラ構成の形式です。
camerapacketsize	イーサネットパケットサイズを指定します。デフォルト値は、現在のカメラ構成の値です。
transactional	イベントブロックの種類をトランザクションに設定します。デフォルトのイベントブロックの種類は標準です。

パラメータ	説明
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
publishwithupsert	opcode=Insert の代わりに opcode=Upsert を使用してイベントを作成します。
cameraxoffset	関心領域横方向オフセットを指定します。デフォルト値は、現在のカメラ構成の値です。
camerayoffset	関心領域縦方向オフセットを指定します。デフォルト値は、現在のカメラ構成の値です。
maxevents	パブリッシュするイベントの最大数を指定します。

Pylon パブリッシャアダプタの使用

Pylon パブリッシャアダプタでは、Pylon パブリッシャコネクタと同じ機能が提供され、アダプタのみのオプションパラメータがいくつか追加されています。

使用法:

dfesp_pylon_adapter -h *url* <-a *camerafeaturesfile*> <-C *configfilesection*> <-E *tokenlocation*> <-e> <-f *maxframerate*> <-g *gdconfig*> <-i *cameraipaddress*> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxevents*> <-n *maxnumframes*> <-p *camerapixelformat*> <-Q> <-R> <-s *camerapacketsize*> <-t *cameraheight*> <-V> <-w *camerawidth*> <-y *logconfigfile*> <-Z *transportconfigfile*> <-X *cameraxoffset*> <-Y *camerayoffset*>

パラメータ	説明
-a camerafeaturesfile	ロードする Pylon 機能ストリーム(*.pfs)構成ファイルを指定します。デフォルトでは、現在のカメラ構成を変更せずに使用します。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e	イベントが transactional であることを指定します。

パラメータ	説明
-f maxframerate	パブリッシュする 1 秒あたりの最大フレーム数を指定します。デフォルト値は、フレームをカメラから受信する速度です。
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP のパブリッシュとサブスクライブの標準 URL を dfESP://host:port/project/continuousquery/window の形式で指定します。
-i cameraipaddress	カメラの IP アドレスを指定します。デフォルト値は、ローカルサブネットで検出された最初のカメラのアドレスです。
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-m maxevents	パブリッシュするイベントの最大数を指定します。
-n maxnumframes	パブリッシュするフレームの最大数を指定します。デフォルトでは、最大値はありません。
-p camerapixelformat	画像のピクセル形式を指定します。デフォルト値は、現在のカメラ構成の形式です。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-R	opcode=Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-s camerapacketsize	イーサネットパケットサイズを指定します。デフォルト値は、現在のカメラ構成の値です。
-t cameraheight	関心領域の高さを指定します。デフォルト値は、現在のカメラ構成の値です。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-w camerawidth	関心領域の幅を指定します。デフォルト値は、現在のカメラ構成の値です。
-y logconfigfile	ログ構成ファイルを指定します。

パラメータ	説明
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。
-X cameraxoffset	関心領域横方向オフセットを指定します。デフォルト値は、現在のカメラ構成の値です。
-Y camerayoffset	関心領域縦方向オフセットを指定します。デフォルト値は、現在のカメラ構成の値です。

Rabbit MQ コネクタとアダプタの使用

Rabbit MQ コネクタの使用

Rabbit MQ コネクタは、パブリッシュおよびサブスクライブ操作のために Rabbit MQ サーバーと通信します。コネクタによって提供されるバス接続により、エンジンが個々のパブリッシュ/サブスクライブ接続を管理する必要がなくなります。コネクタは、単一のエンジンに対する同時パブリッシュ/サブスクライブ接続の大容量を実現します。

Rabbit MQ サブスクライバコネクタは、イベントブロックを受信し、Rabbit MQ ルーティングキーにパブリッシュします。Rabbit MQ パブリッシャコネクタは、ダイナミックに作成された Rabbit MQ キューからイベントブロックを読み込み、イベントストリームを処理するソースウィンドウにインジェクトします。

Rabbit MQ サーバーを介して送信されるイベントブロックは、バイナリ、CSV、Google の protobufs、または JSON メッセージとしてエンコードできます。コネクタはバイナリ形式との変換を行います。メッセージ形式はコネクタ構成パラメータです。

Rabbit MQ コネクタは、ホットフェイルオーバー操作をサポートしています。このモードでは、Rabbit MQ サーバーに presence-exchange プラグインをインストールする必要があります。プラグインは https://github.com/tonyg/presence-exchange からダウンロードできます。

存在するプレゼンス交換バージョンが、インストールされている Rabbit MQ サーバーのプレゼンス交換バージョンと一致していることを確認します。たとえば、サーバーバージョン 3.5.*x* を使用する場合は、プレゼンス交換バージョン 3.5.*y* を使用できます。*x* と *y* は異なりますが、両方ともバージョン 3.5 です。ミスマッチしたバージョン(3.4.*x* や 3.3.*y* など)が動作する場合もありますが、この種類のミスマッチはお勧めしません。

パブリッシュ/サブスクライブのクライアントプラグインライブラリを処理する、対応するイベントストリームが使用可能です。このライブラリにより、パブリッシュ/サブスクライブクライアントアプリケーションを処理する標準イベントストリームが、Rabbit MQ サーバーを介して Event Stream Processing Server とイベントブロックを交換することができます。交換は直接 TCP 接続ではなく、Rabbit MQ サーバーを介して行われます。この交換を有効にするには、C_dfESPpubsubSetPubsubLib()の呼び出しを追加します。

ホットフェイルオーバー操作用に構成されている場合、コネクタのアクティブ/スタンバイ状態は Rabbit MQ サーバーと調整されます。したがって、アクティブコネクタに 障害が発生すると、スタンバイコネクタがアクティ

ブになります。成功したスイッチオーバーを保証するには、関係するすべてのコネクタが次の条件を満たす必要があります。

- 同一の ESP モデルに属していなければなりません。
- 同時にメッセージフローを開始する必要があります。これは、すべてのコネクタでメッセージ ID を同期させる必要があるためです。

新しいサブスクライバコネクタがアクティブになると、アウトバウンドメッセージフローは同期したままになります。これは、スタンバイコネクタによるメッセージのバッファリングと、Rabbit MQ サーバーとの再開されたフローの調整によるものです。メッセージバッファのサイズは、サブスクライバコネクタに必要なパラメータです。

サブスクライバ Rabbit MQ コネクタを構成して、ウィンドウコンテンツのカスタムスナップショットを任意のサブスクライバクライアントに送信できます。クライアントは、Rabbit MQ サーバーへの新しい接続を確立している必要があります。これにより、遅れたサブスクライバは接続時に追いつくことができます。また、この機能を使用するには、Rabbit MQ サーバーにプレゼンス交換プラグインをインストールする必要があります。

コネクタが開始すると、トピック urlhostport/M (ここで、*urlhostport* はコネクタ構成パラメータ)がサブスクライブされます。これにより、コネクタは、その *host:port* の組み合わせに関連付けられたエンジン内のウィンドウにパブリッシュまたはサブスクライブするクライアントからのメタデータ要求を受信できます。メタデータ応答は、以下のいくつかの組み合わせで構成されます。

- コネクタに関連付けられたウィンドウのプロジェクト名
- コネクタに関連付けられたウィンドウのクエリ名
- コネクタに関連付けられたウィンドウのウィンドウ名
- ウィンドウのシリアライズされたスキーマ

実行中のコネクタインスタンスをホストするプラットフォームに Rabbit MQ クライアントランタイムライブラリをインストールする必要があります。コネクタは rabbitmq-c v0.5.2 C ライブラリを使用します。これは https://github.com/alanxz/rabbitmq-c からダウンロードできます。ランタイム環境では、それらのライブラリへのパスを定義する必要があります(たとえば、Linux プラットフォームの場合は LD_LIBRARY_PATH に指定)。SSL サポートを有効にした rabbitmq-c ライブラリを構築する場合は、その SSL ライブラリへのパスをランタイム環境に含めるようにしてください。

パブリッシャによって作成されたキューの場合、オプションの buspersistence パラメータは、auto-delete と durable の両方を制御します。

buspersistence **パラメータの設定**	**効果**
true	auto-delete = false durable = true
false	auto-delete = true durable = false

それらのキューから消費するときは、次のことが成り立ちます。

buspersistence **パラメータの設定**	**効果**
true	exclusive = true noack = false
false	exclusive = false noack = true

パブリッシャコネクタが自動削除を無効にして持続受信キューを作成すると、noack = false でそのキューから消費されますが、メッセージを明示的に確認しません。これにより、再起動された Event Stream Processing Server は持続メッセージを受信できます。特定の年代よりも古いメッセージのグループを buspersistent パブリッシャに確認させる場合は、ackwindow と acktimer パラメータの組み合わせを構成します。これにより、管理者の介入を防ぎながら、メッセージキューの無制限化を防ぎます。

キュー名は、buspersistencequeue パラメータに構成されたトピックパラメータを追加します。buspersistencequeue パラメータは、同じ Rabbit MQ 交換を使用するすべてのパブリッシャコネクタで一意である必要があります。パブリッシャコネクタは、buspersistence が使用可能になっているときに排他モードでキューを消費することによって、これを強制します。

サブスクライバコネクタでは、buspersistence を有効にすると、配信モードが **persistent** に設定された状態でメッセージが送信されます。

パブリッシュまたはサブスクライブによって作成されたエクスチェンジの場合、buspersistence は durable のみを制御します。つまり、buspersistence = true、durable = true、buspersistence = false の時、durable = false です。

必要に応じて、rmqpassword パラメータを暗号化されたパスワードで構成できます。暗号化されたパスワードのバージョンは、システムにインストールする必要がある OpenSSL を使用して生成することができます。SAS Event Stream Processing システム暗号化および認証オーバーレイをインストールしている場合は、付属の OpenSSL 実行可能ファイルを使用できます。コンソールで次のコマンドを使用して、OpenSSL を起動し、暗号化されたパスワードを表示します。

echo "*rmqpassword*" | openssl enc -e -aes-256-cbc -a -salt -pass pass:"SASespRMQconnectorUsedByUser=*rmquserid*"

暗号化されたパスワードを rmqpassword パラメータにコピーし、rmqpasswordencrypted パラメータを有効にします。

表 44 サブスクライバ Rabbit MQ コネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
rmquserid	Rabbit MQ サーバーとのコネクタのセッションを認証するために必要なユーザー名を指定します。
rmqpassword	rmquserid に関連付けられたパスワードを指定します。
rmqhost	Rabbit MQ サーバーのホスト名を指定します。
rmqport	Rabbit MQ サーバーポートを指定します。
rmqexchange	存在しない場合は、コネクタによって作成された Rabbit MQ 交換を指定します。
rmqtopic	メッセージがパブリッシュされる Rabbit MQ ルーティングキーを指定します。
rmqtype	binary、CSV、JSON、またはサブスクライブされたウィンドウスキーマ内の文字列フィールドの名前を指定します。
urlhostport	開始メタデータ要求の開始時にサブスクライブされているメタデータトピックの *host:port* フィールドを指定します。
numbufferedmsgs	スタンバイサブスクライバコネクタによってバッファされるメッセージの最大数を指定します。超過すると、最も古いメッセージは破棄されます。コネクタがアクティブになると、バッファがフラッシュされ、メッセージ ID シーケンスを維持するためにバッファされたメッセージが必要に応じてファブリックに送信されます。

パラメータ	説明
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 このパラメータは、buspersistence または hotfailover が使用可能な場合は無効です。

表 45 パブリッシャ Rabbit MQ コネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
rmquserid	Rabbit MQ サーバーとのコネクタのセッションを認証するために必要なユーザー名を指定します。
rmqpassword	rmquserid に関連付けられたパスワードを指定します。
rmqhost	Rabbit MQ サーバーのホスト名を指定します。
rmqport	Rabbit MQ サーバーポートを指定します。
rmqexchange	存在しない場合は、コネクタによって 作成された Rabbit MQ 交換を指定します。
rmqtopic	メッセージがパブリッシュされる Rabbit MQ ルーティングキーを指定します。
rmqtype	binary、CSV、JSON、または opaquestring を指定します。opaquestring の場合、ソースウィンドウスキーマは index:int64,message:string とみなされます。
urlhostport	開始メタデータ要求の開始時にサブスクライブされているメタデータトピックの *host:port* フィールドを指定します。

表 46 サブスクライバ Rabbit MQ コネクタのオプションパラメータ

パラメータ	説明
collapse	サブスクライバ出力をパブリッシュ可能にするために UPDATE_BLOCK イベントの変換を有効にします。
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバによって受信されたイベントブロックからすべての削除イベントを削除することを指定します。
hotfailover	ホットフェイルオーバーモードを有効にします。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。

パラメータ	説明
buspersistence	永続配信モードを使用してメッセージを送信するように指定します。
protofile	イベントブロックを protobuf メッセージに変換するために使用される Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
csvincludeschema	rmqtype=CSV の場合、ウィンドウのシリアライズされたスキーマに出力 CSV データをいつ付加するかを指定します。有効な値は、never、once、および pereventblock です。デフォルト値は“never”です。
useclientmsgid	フェイルオーバー操作を実行してイベントブロックからメッセージ ID を抽出するときは、エンジンが生成したメッセージ ID の代わりにクライアントが生成したメッセージ ID を使用します。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
rmqpasswordencrypted	rmqpassword が暗号化されることを指定します。
rmqvhost	Rabbit MQ 仮想ホストを指定します。デフォルトは/です。
csvmsgperevent	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
csvmsgpereventblock	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
rmqcontenttype	RabbitMQ に送信されるメッセージの content_type パラメータの値を指定します。デフォルト値は、次のようにメッセージ形式によって異なります。 ■ バイナリ application/octet-stream ■ csv text/csv;charset=utf-8 ■ json application/json ■ protobufs application/x-protobuf ■ opaque text/csv;charset=utf-8

パラメータ	説明
rmqheaders	RabbitMQ に送信されるメッセージのキー値のオプションのヘッダーのコンマ区切りのリスト。デフォルト値はヘッダーなしです。
rmqssl	Rabbit MQ サーバーへの接続で SSL 暗号化を有効にします。
rmqsslcacert	rmqssl が有効な場合は、SSL CA 証明書.pem ファイルのフルパスを指定します。
rmqsslkey	rmqssl が有効な場合は、SSL キー.pem ファイルのフルパスを指定します。
rmqsslcert	rmqssl が有効な場合は、SSL 証明書.pem ファイルのフルパスを指定します。
doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

表 47 パブリッシャ Rabbit MQ コネクタのオプションパラメータ

パラメータ	説明
transactional	rmqtype=CSV の場合、イベントブロックの種類を transactional に設定します。デフォルトのイベントブロックの種類は標準です。
blocksize	rmqtype=CSV の場合、公開されたイベントブロックに含めるイベント数を指定します。デフォルト値は 1 です。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
buspersistence	auto-delete と durable の両方を制御します。
buspersistencequeue	持続パブリッシャが使用するキュー名を指定します。
ignorecsvparseerrors	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。
protofile	イベントブロックを protobuf メッセージに変換するために使用される Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。

パラメータ	説明
csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。
noautogenfield	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。
ackwindow	Rabbit MQ から受信したメッセージを未確認のままにする時間(秒単位)を指定します。デフォルトでメッセージが確認応答されないときの buspersistence = true の場合にのみ適用されます。構成されると、メッセージは受信された後、この秒数で確認応答されます。acktimer が構成されている場合は構成する必要があります。
acktimer	ackwindow パラメータによってトリガーされる確認応答を送信するかどうかをチェックする頻度(秒単位)を指定します。ackwindow が構成されている場合は構成する必要があります。
publishwithupsert	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
rmqpasswordencrypted	rmqpassword が暗号化されることを指定します。
addcsvopcode	CSV イベントを入力するための演算コードとカンマを先頭に付けます。演算コードは、publishwithupsert が有効になっていない場合は挿入です。
addcsvflags	入力 CSV イベントに挿入するイベントの種類を指定します(カンマ)。有効な値は normal と partialupdate です。
rmqvhost	Rabbit MQ 仮想ホストを指定します。デフォルトは/です。
useclientmsgid	ソースウィンドウがディスクからディスクに 復元された場合、復元されたウィンドウに最大のメッセージ ID よりも小さいメッセージ ID を含む受信したバイナリイベントブロックは無視されます。
rmqssl	Rabbit MQ サーバーへの接続で SSL 暗号化を有効にします。
rmqsslcacert	rmqssl が有効な場合は、SSL CA 証明書.pem ファイルのフルパスを指定します。
rmqsslkey	rmqssl が有効な場合は、SSL キー.pem ファイルのフルパスを指定します。
rmqsslcert	rmqssl が有効な場合は、SSL 証明書.pem ファイルのフルパスを指定します。
maxevents	パブリッシュするイベントの最大数を指定します。

Rabbit MQ アダプタの使用

Rabbit MQ アダプタは、Rabbit MQ サーバでのパブリッシュおよびサブスクライブ操作をサポートします。アダプタを使用するには、**rabbitmq-c** V0.5.2 クライアントランタイムライブラリをインストールする 必要があります。

注: 使用する前に、このコネクタへの参照をファイル**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.excluded** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.excluded** (Windows)から削除します。

サブスクライバの使用:

dfesp_rmq_adapter -h *url* -k sub -n *numbufferedmsgs* -o *urlhostport* -p *rmqpassword* -r *rmqport* -s *rmqhost* -t *rmqtopic* -u *rmquserid* -v *rmqexchange* -z binary | csv | json <-C *configfilesection* > <-c *rmqcontenttype*> <-d *dateformat* > <-E *tokenlocation* > <-f *protofile* > <-g *gdconfig* > <-H *rmqheaders*> <-j trace | debug | info | warn | error | fatal | off> <-K *rmqsslkey*> <-L> <-l native | solace | tervela | rabbitmq | kafka> <-M> <-m *protomsg* > <-S> <-T *rmqsslcacert*> <-U *rmqsslcert*> <-V> <-w never | once | pereventblock> <-X> <-x > <-Y *rmqvhost*> <-y *logconfigfile* > <-Z *transportconfigfile*> <-3 *doubleprecision*>

パブリッシャ:

dfesp_rmq_adapter -h *url* -k pub -o *urlhostport* -p *rmqpassword* -r *rmqport* -s *rmqhost* -t *rmqtopic* -u *rmquserid* -v *rmqexchange* -z binary | csv | json | opaquestring <-A> < -a *ackwindow* > <-b *blocksize* > <-C *configfilesection* > <-D *csvfielddelimiter* > <-d *dateformat* > <-E *tokenlocation* > <-e > <-F *eventtype*> <-f *protofile* > <-g *gdconfig* > <-I > <-i *acktimer* > <-j trace | debug | info | warn | error | fatal | off> <-K *rmqsslkey*> <-l native | solace | tervela | rabbitmq | kafka> <-m *protomsg* > <-N *maxevents*> <-O> <-Q> <-q *buspersistencequeue* > <-R> <-S> <-T *rmqsslcacert*> <-U *rmqsslcert*> <-V> <-X> < -x > <-Y *rmqvhost*> <-y *logconfigfile* > <-Z *transportconfigfile*>

パラメータ	説明
-A	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。
-a ackwindow	buspersistence が有効になっているときに Rabbitmq から受信した未確認のメッセージを残す時間を指定します。
-b blocksize	イベントブロックサイズを指定します。デフォルトは 1 です。
-C configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-c rmqcontenttype	RabbitMQ に送信されるメッセージの content_type パラメータの値を指定します。デフォルト値は、次のようにメッセージ形式によって異なります。 ■ バイナリ "application/octet-stream" ■ csv "text/csv;charset=utf-8" ■ json "application/json" ■ protobufs "application/x-protobuf" ■ 不透明 "text/csv;charset=utf-8"
-D csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。

パラメータ	説明
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e	イベントがトランザクションであることを指定します。
-F eventtype	入力 CSV イベントに挿入するイベントの種類をカンマで指定します。有効な値は "normal"と "partialupdate"です。
-f protofile	Google プロトコルバッファーのサポートに使用される**.proto** ファイルを指定します。
-g gdconfig	保証された配信構成ファイルを指定します。
-H rmqheaders	RabbitMQ に送信されるメッセージのキー値のオプションヘッダーのコンマ区切りリストを指定します。デフォルト値はヘッダーなしです。
-h url	dfESP のパブリッシュとサブスクライブの標準 URL を dfESP://host:port/project/continuousquery/window の形式で指定します。 サブスクライバに次のものを追加します。 ?snapshot=true | false 。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに以下を追加します。 ■ ?collapse=true | false ■ ?rmretdel=true | false
-I	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。
-i acktimer	ackwindow パラメータによってトリガーされる 確認応答を送信するかどうかをチェックする頻度の時間間隔を指定します。
-j trace | debug | info | warn | error | fatal | off	ロギングレベルを指定します。デフォルトは“warn”です。
-K rmqsslkey	rmqssl が有効な場合は、SSL キー.pem ファイルのフルパスを指定します。

パラメータ	説明
-k	サブスクライバの場合は sub、パブリッシャの場合は pub を指定します。
-L	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-M	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
-m protomsg	protofile パラメータで指定された**.proto** ファイル内のメッセージ自体を指定します。
-N maxevents	パブリッシュするイベントの最大数を指定します。
-n numbufferedmsgs	スタンバイサブスクライバコネクタによってバッファされるメッセージの最大数を指定します。
-O	CSV イベントを入力するための演算コードとカンマを先頭に付けます。演算コードは、 -R が有効になっていない場合は挿入されます。
-o urlhostport	コネクタがサブスクライブするメタデータトピック内の*ホスト*:*ポート*フィールドを指定します。
-p rmqpassword	Rabbit MQ のパスワードを指定します。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-q buspersistencequeue	持続パブリッシャが使用するキュー名を指定します。
-R	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-r rmqport	Rabbit MQ ポートを指定します。
-S	Rabbit MQ サーバーへの接続で SSL 暗号化を有効にします。
-s rmqhost	Rabbit MQ ホストを指定します。
-T rmqsslcacert	rmqssl が有効な場合は、SSL CA 証明書.pem ファイルのフルパスを指定します。
-t rmqtopic	Rabbit MQ ルーティングキーを指定します。

パラメータ	説明
-U rmqsslcert	rmqssl が有効な場合は、SSL 証明書.pem ファイルのフルパスを指定します。
-u rmquserid	Rabbit MQ ユーザー名を指定します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-v rmqexchange	Rabbit MQ 交換を指定します。
-w never | once | pereventblock	rmqtype=CSV の場合、ウィンドウのシリアライズされたスキーマに出力 CSV データをいつ付加するかを指定します。デフォルト値は“never”です。
-X	rmqpassword が暗号化されることを指定します。
-x	耐久性のあるキューと持続メッセージを使用します。
-Y rmqvhost	Rabbit MQ 仮想ホストを指定します。デフォルトは/です。
-y logconfigfile	ロギング構成ファイルを指定します。デフォルトでは、何もありません。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。
-z binary | csv | json | opaquestring	メッセージの形式を指定します。opaquestring はパブリッシャにのみ有効です。opaquestring の場合、ソースウィンドウスキーマは "index:int64,message:string"とみなされます。サブスクライバの場合、サブスクライブされたウィンドウスキーマ内の文字列フィールドの名前もサポートされます。
-3 doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

REST サブスクライバアダプタの使用

REST アダプタは、構成済みの REST サービスへの HTTP POST 要求を生成するためのサブスクライブ操作をサポートします。サブスクリプションされたイベントごとに、アダプタは opaquejson パラメータを構成しないかぎ

り、イベントのすべてのフィールドを使用する JSON 文字列をフォーマットします。その場合、フィールドは 1 つだけ使用されます。文字列をフォーマットした後、アダプタは HTTP POST を介して JSON をアダプタパラメータ resturl で構成されている REST サービスに転送します。HTTP POST が失敗した場合の HTTP Content-Type および再試行回数を構成することもできます。追加の HTTP POST ヘッダーは、httprequestproperties パラメータで指定できます。

HTTP レスポンスは、無関係のソースウィンドウに転送できます。これには、JSON 形式の応答からイベントを構築し、それを espresprurl アダプタパラメータで構成された ESP URL に転送する必要があります。

アンダースコアを含むサブスクライブされたウィンドウスキーマ内のフィールド名は、ネストされた JSON フィールドに変換されます。たとえば、フィールド名 foo_bar では、"foo":{"bar":"value"という JSON が生成されます。フィールド名 foobar では、"foobar":"value"という JSON が生成されます。

JSON レスポンスに 1 つ以上の配列が含まれている場合、イベントを正常に構築するには、次のいずれかの条件が満たされている必要があります。

- 配列は、応答全体を含む外部配列です。この場合、配列内のすべてのエントリに対してイベントが作成されます。
- 配列には、いくつかのキーに複数の値が含まれています。この場合、同じネストレベルの他のすべてのキーには、同じサイズの配列内の値が含まれていなければなりません。そのイベントは、その配列の各配列内のそのインデックスからの値を使用して、各配列インデックスに対して構築されます。

応答イベントフィールドは、元のサブスクライブイベントのフィールドに追加されます。したがって、espresprurl のソースウィンドウのスキーマにある開始フィールドが、サブスクライブウィンドウの完全なスキーマと正確に一致するように注意する必要があります。また、応答フィールドは開始フィールドの後になければなりません。

各 HTTP 要求 - 応答アクションは、別々のアダプタスレッドで実行されます。この方法では、アダプタのメモリ使用量は、以前の要求応答が完了するのに同期して待っている、キューに入れられたサブスクライブされたイベントでは増加しません。

使用法:

dfesp_rest_subscriber -r *resturl* -t *httpcontenttype* -u *url* <-c [*configfilesection*]> <-d *dateformat* > <-e *espresprurl* > <- g *gdconfigfile* > <-i><-j *opaquejson* > <-l native | solace | tervela | rabbitmq | kafka> <-m *maxnumthreads* > <-O *tokenlocation* > <-o severe | warning | info> <-p *httpretries* > <-q *httprequestproperties*> <-s *httpstatuscodes* > <-V> <-v *httpretryinterval*> <-Z *transportconfigfile*>

パラメータ	説明
-c [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-e espresprurl	応答を公開するパブリッシュ/サブスクライブ標準 URL を指定します。
-g gdconfigfile	クライアントの保証された配信構成ファイルを指定します。

パラメータ	説明
-i	サブスクライブされたイベントをドロップし、HTTP 接続が失敗したときに続行することを指定します。デフォルトは終了します。 **javaadapters.config** パラメータ名: ignorefailedhttpconnects
-j opaquejson	完全な JSON リクエストを抽出するためのサブスクライブウィンドウフィールドを指定します。これは、すべてのウィンドウフィールドからのリクエストを作成するのとは対照的です。フィールドの種類は ESP_UTF8STR でなければなりません。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-m maxnumthreads	アダプタが使用するスレッドの最大数を指定します。これにより、REST サービスへの同時接続数が制限されます。デフォルト値は 32 です。
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-o severe | warning | info	アプリケーションのログレベルを指定します。
-p httpretries	失敗した HTTP 投稿を再試行する回数を指定します。デフォルト値は 0 です。値-1 は、無限に投稿を再試行することを指定します。
-r resturl	ターゲット REST サービスの URL を指定します。
-s httpstatuscodes	HTTP ポストに応答して許容可能なステータスコードをコンマで区切ったリストを指定します。デフォルトの応答は 201 です。
-t httpcontenttype	HTTP ポストで使用される Content-Type 文字列の値を指定します。
-u url	dfESP サブスクライブの標準 URL を "dfESP://*host*:*port*/*project*/*contquery*/*window*?snapshot=true|false"の形式で指定します。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて次の行を追加します。 ?collapse=true | false ?rmretdel=true | false
-V	致命的なエラーが報告された場合は、アダプタを再起動します。 **javaadapters.config** パラメータ名: restartonerror
-v httpretryinterval	HTTP POST の再試行間隔(秒)を指定します。デフォルトは 0 です。

パラメータ	説明
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。
-q httprequestproperties	REST サービスへの HTTP POST 要求に含める任意の数の追加ヘッダーを指定します。セミコロンで区切られた<*key*>:<*value*>ペアとして指定します。複数の値をサポートするキーの場合は、それらの値をカンマで区切ります。

SAS データセットアダプタの使用

SAS データセットアダプタは、Java パブリッシャとサブスクライバの SAS Event Stream Processing クライアントをバンドルする dfx-esp-dataset-adapter.jar にあります。

アダプタは SAS Java Database Connectivity (JDBC)を使用して SAS Workspace Server に接続します。この SAS Workspace Server は、データセットの読み書きを管理します。SAS データセット名と SAS Workspace Server のユーザー資格情報は、必要なパラメータとしてアダプタに渡されます。

SAS JDBC では、ログ用に log4j- *version* .jar が必要です。DFESP_LOG4J_JAR 環境変数の値を log4j- *version* .jar へのパスに構成します。

注: JAR ファイルは**$DFESP_HOME/lib** にあります。

環境変数 DFESP_LOG4J_XML の値を log4j.xml ファイルの場所に構成します。

-w true が構成されていない場合、ユーザーパスワードは暗号化された形式で渡されなければなりません。暗号化されたパスワードのバージョンは、システムにインストールする必要がある OpenSSL を使用して生成することができます。SAS Event Stream Processing システムの暗号化と認証のオーバーレイをインストールするときは、付属の OpenSSL 実行可能ファイルをインストールします。

コンソールで次のコマンドを使用して OpenSSL を起動し、暗号化された *password* を表示します。

echo "*password*" | **openssl** enc -e -aes-256-cbc -a -salt -pass pass:"SASespDSadapterUsedByUser=*userid*""

サブスクライバクライアントはイベントブロックを受信し、作成する SAS データセットに対応する行を追加します。また、データセットの更新および削除操作もサポートされています。

注: 既存のデータセットで SAS データセットアダプタを呼び出すと、データセットが上書きされます。

Workspace Server では、ローカルファイルシステムにデータセットが作成されるため、データセット名は次の SAS 命名規則に準拠している必要があります。

- 名前の長さは最大 32 文字です。
- 名前は、ラテンアルファベット(A-Z、a-z)の文字またはアンダースコアで始まる必要があります。後続の文字には、ラテンアルファベット、数字、またはアンダースコアの文字を使用できます。
- 名前には、アンダースコア以外の空白や特殊文字は使用できません。
- 名前には大文字と小文字が混在しています。SAS は内部的にメンバー名を大文字に変換します。

–s オプションを使用して、データセットスキーマを明示的に指定することができます。–a スイッチを使用すると、次のことができます。

- 分析する受信イベントの数を指定します
- データセットの行を作成する前に、データセットの適切な行サイズを抽出します

–s または–a スイッチが存在する必要があります。

オプションの periodicity または maxnumrows パラメータを使用して定期的に SAS データセットを書き込むようにサブスクライバクライアントを構成することもできます。このように構成すると、書き込まれた各ファイルのファイル名にタイムスタンプが追加されます。periodicity または maxnumrows が構成されている場合は、参照されている行が現在開いているデータセットに存在しない可能性があるため、更新および削除操作はサポートされないことに注意してください。

複数の ESP URL を持つサブスクライバクライアントを構成した場合、複数のモデルの複数のウィンドウからのイベントが単一の出力データセットに集約されます。この場合、サブスクライブされた各ウィンドウは同じスキーマを持つ必要があります。また、データセットに行が追加される順序は保証されません。

パブリッシャクライアントは次の処理を行います。

- SAS データセットから行を読み込みます
- opcode = insert でイベントに変換します（-r を指定しない限り）
- エンジンのソースウィンドウにイベントブロックをインジェクトします

パブリッシャクライアントを複数の URL で構成すると、生成された各イベントブロックが複数のソースウィンドウにインジェクトされます。各ソースウィンドウには同じスキーマが必要です。

サブスクライバの使用:

dfesp_dataset_adapter -d *wssurl* -f *dsname* -h *url1* ... *urlN* -k sub -u *username* -x *password* <-a *obstoanalyze* > <-C *configfilesection* > <-g *gdconfigfile* > <-L native | solace | tervela | rabbitmq | kafka> <-l *loglevel* > <-m *maxnumrows* > <-O *tokenlocation* > <-p *periodicity* > <-s *dsschema*> <-V> <-w> <-Z *transportconfigfile*> <-z *bufsize* >

パブリッシャ:

dfesp_dataset_adapter -d *wssurl* -f *dsname* -h *url1* ... *urlN* -k pub -u *username* -x *password* <-b *blocksize* > <-C *configfilesection* > <-e> <-g *gdconfigfile* > <-L native | solace | tervela | rabbitmq | kafka> <-l *loglevel* > <-O *tokenlocation* > <-Q> <-q *sqlquery* > <-r> <-V> <-W *maxevents*> <-w> <-Z *transportconfigfile*> <-z *bufsize* >

パラメータ	説明
-a obstoanalyze	出力 SAS データセット構造を定義するために分析するオブザベーション数を指定します。デフォルト値は 4096 です。以前に-s オプションを指定していた場合、このオプションは無視されます。
-b blocksize	サブスクライバの場合、データセットに追加するイベントブロックの数を一度に指定します。パブリッシャの場合は、ESP ソースウィンドウにパブリッシュするイベントブロックサイズを指定します。デフォルト値は 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-d wssurl	SAS Workspace Server の接続 URL を “host:port”の形式で指定します。

パラメータ	説明
-e	イベントブロックがトランザクションであることを指定します。デフォルトでは、イベントブロックは正常です。 **javaadapters.config** パラメータ名:transactional
-f dsname	サブスクライバ出力ファイルまたはパブリッシャ入力ファイルを指定します。dsname には、拡張子を含む完全パスを指定する必要があります。次に例を示します。 -f "D:/sas/SASBI/Lev1/AppData/filename.sas7bdat"
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	1 つ以上の標準 dfESP URL を次の形式で指定します。 dfESP://host:port/project/continuousquery/window サブスクライバの URL に次の行を追加します。 ?snapshot=true | false ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに以下を追加します。 ?collapse=true | false ?rmretdel=true | false
-k pub | sub	パブリッシャまたはサブスクライバを指定します。
-L native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、*"Java クライアント用の代替トランスポートライブラリの使用" (SAS Event Stream Processing: パブリッシュ/サブスクライブ API)*で指定された必要なクライアント構成ファイルを使用します。
-l loglevel	Java 標準のログレベルを指定します。SEVERE | WARNING | INFO のいずれかの値を使用します。デフォルト値は INFO です。
-m maxnumrows	出力ファイルの周期性を行数で指定します。データが挿入専用でない場合は無効です。
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します
-p periodicity	出力ファイルの周期性を秒単位で指定します。データが挿入専用でない場合は無効です。
-Q	すべてのイベントがソースウィンドウにインジェクトされた後、プロジェクトを休止します。 **javaadapters.config** パラメータ名:quiesceproject
-q sqlquery	SAS データセットのコンテンツを取得するための SQL クエリを指定します。たとえば、 "select * from dataset" 。 **注:** 拡張子なしで-f オプションに指定した *dataset* を使用します。
-r	opcode=Upsert でイベントを作成します。デフォルトでは、イベントは opcode=Insert で構築されます。 **javaadapters.config** パラメータ名: publishwithupsert

パラメータ	説明
-s dsschema	出力データセットスキーマを次の形式で指定します。 "rowname1:sastype1:sasformat1:label1,...,rownameN:sastypeN:sasformatN:labelN" *sasformatX* を省略すると、行に SAS 形式は適用されません。*labelX* を省略すると、ラベルは行に適用されません。 'char'型のサイズを指定する必要があります。'num'型のサイズはデフォルトサイズ 8 であり、指定する必要はありません。 ラベルは引用符で囲む必要はありません。 次の例を参照してください。 -s "ID:num,OUTPUT_DTTM:num:DATETIME20.:Output Datetime,VIN:char(17)" **注:** 以前に-a オプションを指定していた場合、このオプションは無視されます。
-u username	SAS Workspace Server のユーザー名を指定します。
-V	致命的なエラーが報告された場合は、アダプタを再始動します。 **javaadapters.config** パラメータ名:restartonerror
-W maxevents	パブリッシュするイベントの最大数を指定します。
-w	構成済みの SAS Workspace Server のパスワードが暗号化されていないことを指定します。デフォルトでは、パスワードは暗号化されています。 **javaadapters.config** パラメータ名:unencryptedpassword
-x password	SAS Workspace Server のパスワードを指定します。-w true が構成されている場合は、このパスワードを暗号化する必要があります。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-L パラメータで指定されたトランスポートの種類によって異なります。 -L solace の場合、デフォルトは**./solace.cfg** です。 -L tervela の場合、デフォルトは**./client.config** です。 -L rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -L kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。
-z bufsize	ソケットバッファサイズを指定します。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

SMTP コネクタとアダプタの使用

SMTP サブスクライバコネクタの使用

SMTP (Simple Mail Transfer Protocol)サブスクライブコネクタを使用して、ウィンドウのイベントブロックまたはアラートや重要なアイテムなどの単一のイベントを電子メールで送信できます。このコネクタはサブスクライブ専用です。SMTP サーバーへの接続はポート 25 を使用します。ユーザー認証は実行されず、プロトコルは暗号化されずに実行されます。

電子メールの送信者と受信者のアドレスは、コネクタに必要な情報です。電子メールの題名には、dfESP://host:port/project/contquery/window という形式の標準イベントストリームプロセッサ URL が含まれ、その後にイベントのキーフィールドのリストが続きます。電子メール本文には、CSV 形式でエンコードされた 1 つ以上のイベントのデータが含まれています。

SMTP コネクタのパラメータは次のとおりです。

表 48 SMTP コネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
smtpserver	SMTP サーバーのホスト名または IP アドレスを指定します。
sourceaddress	電子メールの"差出人"フィールドで使用する電子メールアドレスを指定します。
destaddress	電子メールメッセージを送信する電子メールアドレスを指定します。
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。

表 49 SMTP コネクタのオプションパラメータ

パラメータ	説明
collapse	サブスクライバ出力をパブリッシュ可能にするために UPDATE_BLOCK イベントの変換を有効にします。
emailperevent	true または false を指定します。デフォルトは false です。false の場合、各電子メール本文には完全なイベントブロックが含まれます。true の場合、各メール本文には 1 つのイベントが含まれます。
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバによって受信されたイベントブロックからすべての削除イベントを削除することを指定します。

パラメータ	説明
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

SMTP サブスクライバアダプタの使用

SMTP サブスクライバアダプタはサブスクライバのみです。ソースウィンドウへのパブリッシュはサポートされていません。このアダプタは、サブスクライブされたイベントブロックまたは単一のイベントを構成済みの SMTP サーバーに電子メールメッセージとして転送します。メッセージボディ内のイベントデータは CSV 形式でエンコードされます。

サブスクライバの使用:

dfesp_smtp_adapter -d *destaddress* -h *url* -m *smtpserver* -u *sourceaddress* <-a *dateformat*> <-C [*configfilesection*]> <-E *tokenlocation* > <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-p> <-V> <-y *logconfigfile* ><-Z *transportconfigfile*> <-3 *doubleprecision*>

パラメータ	説明
-a dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-d destaddress	宛先電子メールアドレスを指定します。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します
-g gdconfig	保証された配信構成ファイルを指定します。

パラメータ	説明
-h url	dfESP サブスクライブの標準 URL を "dfESP://host:port/project/continuousquery/window?snapshot=true |false"の形式で指定します。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、以下を追加することができます。 ■ ?collapse=true | false ■ ?rmretdel=true | false
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-m smtpserver	SMTP サーバー名を指定します。
-p	1 つまたは複数のイベントを含む完全なイベントブロックではなく、各電子メールに単一のイベントが含まれることを指定します。
-u sourceaddress	送信元電子メールアドレスを指定します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-y logconfigfile	ログ構成ファイルを指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。
-3 doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

Sniffer コネクタとアダプタの使用

Sniffer パブリッシュコネクタの使用

スニッファコネクタは、プロミスキャスモードでローカルネットワークインターフェイスからパケットを取得し、ソースウィンドウにインジェクトする受信パケットごとにイベントを作成します。コネクタのインスタンスは、インターフェース名、プロトコル、および抽出されてイベントに含まれるフィールドのカンマ区切りリストで構成されます。追加コネクタは、インターフェイス/プロトコル/フィールドの任意の組み合わせで追加のパケットをキャプチャするようインスタンス化され、任意のソースウィンドウにインジェクトされます。

現在、プロトコルのサポートは次のものに限定されています。

- TCP 経由でポート 80 に送信された HTTP パケット。他のポート(8080 など)に送信する HTTP パケットをキャプチャするには、httpports パラメータにポートのリストを構成します。
- Radius Accounting:UDP を介してポート 1813 に送信されたパケットを要求します。1〜190 までの属性値のみがサポートされています。Vendor-Specific 属性の Attribute-Specific フィールドを取り込む場合は、vendorid および vendortype コネクタパラメータを構成する必要があります。
- SSL/TLS クライアント Hello パケットは、TCP または UDP 経由でポート 443 に送信されます。
- 他のポートのトラフィックは、IP 送信元と宛先アドレス、TCP または UDP の送信元と宛先ポート、および verbatim ペイロードがキャプチャされる範囲でのみサポートされます。他のパケットフィールドはキャプチャに使用できません。

イーサネットヘッダーに続くオプションの 802.1Q VLAN タグヘッダーにはサポートが含まれていますが、それ以外の場合は IP ヘッダーがイーサネットヘッダーの直後になければなりません。

コネクタは、ESP に付属していない libpcap ライブラリーを使用します。ターゲットマシンにこれらのライブラリーを個別にインストールする必要があります。これらは http://www.tcpdump.org または Microsoft Windows の場合は http://www.winpcap.org からダウンロードできます。

ほとんどのカーネルは、Raw ソケットを開くアプリケーションから保護します。Linux プラットフォームでは、コネクタは、アプリケーションに未処理のソケットをオープンする許可が与えられていない限り、次のエラーメッセージを記録します。

> “You don't have permission to capture on that device (socket: Operation not permitted)”

コネクタを実行しているサーバーに適切なアクセス許可を付与するには、setcap cap_net_raw,cap_net_admin=eip executable というコマンドを実行します。コマンドを実行するには、root 権限が必要です。

これらのパーミッションを与える副作用は、アプリケーションがシェルの LD_LIBRARY_PATH 環境変数を使用しなくなったことです。Event Stream Processing Server アプリケーションでは、 **$ DFESP_HOME/lib にある**共有オブジェクトを見つける別の方法が必要です。SAS Event Stream Processing サーバーを実行する前に、ldconfig コマンドを使用して共有ライブラリキャッシュを**$DFESP_HOME/lib** ディレクトリに更新します。

packetfields 構成パラメータは、SAS Event Stream Processing スキーマのような構文を使用し、ソースウィンドウスキーマと一致する必要があります。3 つの例外があります。

- ソースウィンドウスキーマには、増加するインデックス値を含み、キーフィールドとして機能する追加の index:int64 フィールドが含まれている必要があります。
- Source ウィンドウのスキーマには、pcap ドライバによって作成されたタイムスタンプを含む frame_time:stamp フィールドも追加する必要があります。
- オプションの addtimestamp コネクタ構成パラメータが指定されている場合、ソースウィンドウスキーマには、タイムスタンプを保持する ts:stamp フィールドも含まれていなければなりません。このフィールドはコネクタによって作成され、現在の時刻を保持します。

index および frame_time フィールドは、ウィンドウスキーマの最初の 2 つのフィールドでなければなりません。ts フィールドは、ウィンドウスキーマの最後のフィールドでなければなりません。

blocksize パラメータが 1 より大きい値で構成されている場合、コネクタは、構成された blocksize 以下のイベントブロックを作成します。pcap ドライババッファがいっぱいになったとき、または pcap ドライバによって開かれたソケットの読み出し timeout が切れたときに、より小さなサイズのイベントブロックを構築することができます。これにより、コネクタは、イベントブロックを埋めるのを待つことなく、インターフェースで受信したパケットから使用可能なすべてのイベントをできるだけ早くインジェクトするようにします。

protocol パラメータが 80,1813、または 443 に設定されておらず、httpports が構成されていない場合、packetfields でサポートされるフィールドは次のとおりです。

- IP ソースと宛先アドレス
- TCP または UDP の送信元ポートと宛先ポート
- payload:string

受信したパケットの形式が誤っているか、無効なパラメータまたは長さが含まれていると、コネクタは一般に info レベルのメッセージを記録し、パケットを無視して処理を続行します。

テストの場合は、ネットワークインターフェイスではなく .pcap キャプチャファイルからパケットを受信できます。このキャプチャファイルの名前は、コネクタインターフェイスのパラメータで 指定できます。コネクタが一致するインターフェース名を見つけることができない場合、コネクタはその名前を.pcapfilename として扱います。

表 50 Sniffer コネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
interface	パケットをキャプチャするローカルマシン上のネットワークインターフェイスの名前を指定します。
protocol	キャプチャするパケットのプロトコルの種類に関連付けられたポート番号を指定します。カンマ区切りのポート番号のリストとして指定できます。

パラメータ	説明
packetfields	キャプチャされたパケットから抽出され、パブリッシュされたイベントに含まれるパケットフィールドを指定します。SAS Event Stream Processing スキーマ構文を使用します。この値には index:int64 frame_time:stamp、または ts:stamp フィールドは含まれません。 ネストされたフィールドをアンダースコアで区切ります。Wireshark オープンソースパケットアナライザで表示されるフィールド名と標準名を一致させます。フィールド名の整合性を維持するには、スペースとハイフンを 削除する必要があります。 Radius の例は次のとおりです。 radius_AcctStatusType:int32,radius_EventTimestamp:stamp,radius_FramedIPAddress:string,radius_UserName:string HTTP の例は次のとおりです。 ip_Source:string,ip_Destination:string,http_Host:string,http_Referer:string,http_UserAgent:string,http_GET_RequestURI:string SSL/TLS の例は次の通りです。 ssltls_Handshake_ClientHello_GMTUnixTime:date,ssltls_Handshake_ClientHello_SessionID:string,ssltls_Handshake_ClientHello_CypherSuites:string,ssltls_Handshake_ClientHello_CompressionMethods:string,ssltls_Handshake_ClientHello_Extensions_ServerName:string protocol が 80,1813、または 443 に設定されておらず、httpports が構成されていない場合、サポートされている値は次のとおりです。 ■ IP ソースと宛先アドレス ■ TCP または UDP の送信元ポートと宛先ポート ■ payload:string

表 51 Sniffer コネクタのオプションパラメータ

パラメータ	説明
transactional	イベントブロックの種類をトランザクションに設定します。デフォルト値は“normal”です。
blocksize	パブリッシュされたイベントブロックに含めるイベントの数を指定します。デフォルト値は 1 です。
addtimestamp	各公開イベントに ESP_TIMESTAMP フィールドを追加することを指定します。フィールド値は、パケットがコネクタによって受信された現在の時刻です。このフィールドは、ソースウィンドウスキーマに存在する必要がありますが、コネクタ packetfields パラメータで必須ではありません。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
vendorid	Radius Accounting-Request パケット内の Vendor-Specific アトリビュートの Attribute-Specific フィールドをキャプチャするときに一致させる vendor-Id フィールドを指定します。

パラメータ	説明
vendortype	Radius Accounting-Request パケット内の Vendor-Specific の属性の Attribute-Specific フィールドを取り込むときに一致させる vendor-Type フィールドを指定します。
indexfieldname	ソースウィンドウスキーマの index:int64 フィールドに index の代わりに使用する名前を指定します。
publishwithupsert	opcode=Insert ではなく opcode=Upsert でイベントを構築することを指定します。
pcapfilter	pcap のマニュアルで定義されているフィルタ式を指定します。コネクタによって受信されたパケットをフィルタリングするために pcap ドライバに渡されます。
httpports	カンマ区切りの宛先ポートのリストを指定します。指定されたポートを含むスニッフィングパケットはすべて、HTTP GET パラメータ用に解析されます。デフォルト値は 80 です。
ignorenopayloadpackets	パケットサイズから TCP または UDP ヘッダーサイズを差し引いて計算した、ペイロードのないパケットを無視するかどうかを指定します。デフォルト値は “false”です。
maxevents	パブリッシュするイベントの最大数を指定します。

Sniffer パブリッシャアダプタの使用

スニファアダプタは、プロミスキャスモードでローカルネットワークインターフェイスからキャプチャされたパケットから作成されるイベントのパブリッシュ操作をサポートします。このアダプタを使用するには、libpcap ランタイムライブラリをインストールする必要があります。

注: 使用する前に、このコネクタへの参照をファイル**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.excluded** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.excluded** (Windows)から削除します。

パブリッシャの使用:

dfesp_sniffer_adapter -f *packetfields* -h *url* -i *interface* -p *protocol* <-b *blocksize* > <-C *configfilesection* > <-d *vendorID* > <-E *tokenlocation* > <-e > <-F *pcapfilter* > <-g *gdconfig* > <-I> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxevents*> <-n *indexfieldname* > <-o *httpports* > <-Q> <-R> <-t> <-V> <-v *vendorType* > <-y *logconfigfile* > <-Z *transportconfigfile*>

パラメータ	説明
-b blocksize	イベントブロックサイズを指定します。デフォルト値は 1 です。
-C configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。

パラメータ	説明
-d vendorID	Radius Accounting-Request パケット内の Vendor-Specific 属性の Attribute-Specific フィールドをキャプチャするときに一致させるベンダー ID フィールドを指定します。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e	イベントがトランザクションであることを指定します。
-F pcapfilter	pcap のマニュアルで定義されているフィルタ式を指定します。この値は、アダプタが受信したパケットをフィルタするために pcap ドライバに渡されます。
-f packetfields	キャプチャされたパケットから抽出されるフィールドのリストを指定します。プロトコルの種類が 80（http）または 1813（radius）でない場合は、"payload:string"を指定する必要があります。
-g gdconfig	保証された配信構成ファイルの名前を指定します。
-h url	dfESP://host:port/project/contquery/window"の形式で dfESP パブリッシュ標準を指定します。
-I	ペイロードがないパケットを無視するかどうかを指定します。パケットサイズから TCP または UDP ヘッダーサイズを差し引いて計算します。
-i interface	パケットをキャプチャするローカルマシン上のネットワークインターフェイスの名前を指定します。
-j trace | debug | info | warn | error | fatal | off	ロギングレベルを指定します。デフォルトは“warn”です。
-l native | solace | tervela | rabbitmq | kafka	パブリッシュ/サブスクライブトランスポートを指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用してください。
-m maxevents	パブリッシュするイベントの最大数を指定します。
-n indexfieldname	ソースウィンドウスキーマの index:int65 フィールドに index の代わりに使用する名前を指定します。
-o httpports	カンマ区切りの宛先ポートのリストを指定します。指定されたポートを含むスニッフィングパケットはすべて、HTTP GET パラメータ用に解析されます。デフォルト値は 80 です。
-p protocol	キャプチャするパケットのプロトコルの種類に関連付けられたポート番号を指定します。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-R	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-t	各公開イベントに ESP_TIMESTAMP フィールドを追加します。

パラメータ	説明
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-v vendorType	Radius Accounting-Request パケット内のベンダー固有属性の Attribute-Specific フィールドを取り込む際に一致させるベンダーの種類を指定します。
-y logconfigfile	ロギング構成ファイルの名前を指定します。デフォルトでは、何もありません。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、"Kafka によるパブリッシャアダプタのフェイルオーバー" (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

Solace コネクタとアダプタの使用

Solace Systems コネクタの使用

Solace Systems コネクタは、パブリッシュおよびサブスクライブ操作のためのハードウェアベースの Solace ファブリックと通信します。

Solace Systems のサブスクライバコネクタは、イベントブロックを受信し、この Solace トピックにパブリッシュします。

host:port/projectname/queryname/windowname/O

Solace Systems のパブリッシャコネクタは、次の Solace トピックからイベントブロックを読み取ります

host:port/projectname/queryname/windowname/I

対応するソースウィンドウにインジェクトします。

コネクタによって提供されるバス接続の結果として、エンジンは個々のパブリッシュ/サブスクライブ接続を管理する必要はありません。単一の Event Stream Processing エンジンへの同時パブリッシュ/サブスクライブ接続の大容量が達成されます。

Solace Systems ランタイムライブラリは、実行中のコネクタインスタンスをホストするプラットフォームにインストールする必要があります。ランタイム環境では、それらのライブラリへのパスを定義する必要があります (たとえば、Linux プラットフォームでは **LD_LIBRARY_PATH** を指定します)。

注: 使用する前に、このコネクタへの参照をファイル**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.excluded** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.excluded** (Windows)から削除します。

Solace Systems コネクタは Solace クライアントとして動作します。すべての Solace 接続パラメータは、コネクタ構成パラメータとして必要です。

コネクタを接続する Solace アプライアンスで、次の項目を構成する必要があります。

- コネクタの soluserid および solpassword 構成パラメータと一致するクライアントのユーザー名とパスワード
- コネクタの solvpn 構成パラメータと一致するメッセージ VPN
- メッセージ VPN で、"サブスクリプションイベントメッセージのパブリッシュ」を使用可能にする必要があります。
- メッセージ VPN では、"SEMP over Message Bus"の下に "Client Commands"と "Show Commands"を有効にする必要があります。
- メッセージ VPN では、非ゼロの"最大スプール使用量"を構成する必要があります。
- サブスクライバコネクタでホットフェイルオーバーが有効になっている場合は、メッセージ VPN に "active_esp"という単一の排他キューを作成する必要があります。キュー所有者を適切なクライアントユーザー名に設定します。このキューに正常にバインドされたサブスクライバコネクタがアクティブコネクタになります。
- buspersistence が有効になっている場合は、メッセージ VPN に"クライアントイベントメッセージのパブリッシュ"を有効にする必要があります。
- buspersistence が有効になっている場合は、すべてのサブスクライブするクライアントに 対して排他キューを作成する必要があります。キュー名は、パブリッシャコネクタ("/ I"トピックの場合)またはクライアントサブスクライバ("/ O"トピックの場合)で構成されたキューで構成された buspersistenceque キューと等しくなければなりません。各構成済みのキューに対応するトピックを追加します。
- サブスクライバコネクタで buspersistence が有効になっているか、またはホットフェイルオーバーが有効になっている場合は、クライアントプロファイルで"保証付きエンドポイントの作成"、"保証付きメッセージ送信の許可"、および"保証付きメッセージ受信の許可"を有効にする必要があります。

コネクタが開始すると、トピック urlhostport/M (ここで、*urlhostport* はコネクタ構成パラメータ)がサブスクライブされます。これにより、コネクタは、*そのホスト:ポート*の組み合わせに関連付けられた ESP エンジン内のウィンドウにパブリッシュまたはサブスクライブするクライアントからのメタデータ要求を受け取ることができます。メタデータ応答は、コネクタに関連付けられたウィンドウのプロジェクト名、照会名、およびウィンドウ名のいくつかの組み合わせと、ウィンドウのシリアライズされたスキーマから構成されます。

Solace Systems サブスクライバコネクタは、ホットフェイルオーバーモードをサポートしています。コネクタのアクティブ/スタンバイ状態はファブリックと調整され、アクティブコネクタに 障害が発生したときにスタンバイコネクタがアクティブになります。スイッチオーバーを実現するには、いくつかの条件を満たす必要があります。

- 関連するすべてのコネクタは、同じトピックのセットでアクティブでなければなりません。
- 関係するすべてのコネクタは、同時にメッセージフローを開始する必要があります。これは、すべてのコネクタでメッセージ ID を同期させる必要があるためです。
- これらのバイナリメッセージには使用可能なメッセージ ID が含まれていないため、Google プロトコルバッファのサポートを有効にしてはいけません。

新しいサブスクライバコネクタがアクティブになると 、スタンバイコネクタによるメッセージのバッファリングと、ファブリックとの再開されたフローの調整によって、発信メッセージフローが同期されたままになります。このメッセージバッファのサイズは、サブスクライバコネクタの必須パラメータです。

デフォルトのダイレクトメッセージングモードではなく、永続的なメッセージングモードを使用するように Solace Systems コネクタを構成できます。（buspersistence 構成パラメータの説明を参照してください。）このモードでは、永続化されたメッセージを期限切れにするための他の自動化されたメカニズムがない場合、管理者が永続データを定期的にパージする必要があります。固定モードでは、ファブリックの最大スループットは低下しますが、他のコネクタがすでにデータを処理した後で、パブリッシャコネクタがファブリックに接続できるようになります。ファブリックは、永続メッセージでコネクタを更新し、ホットフェイルオーバーグループ内の他のエンジンとウィンドウの状態を同期させます。

Solace Systems サブスクライバコネクタは、Solace クライアントがコネクタのトピックにサブスクライブしたときに通知を受けることができる特別なトピックにサブスクライブしています。スナップショットを有効にし

てコネクタを構成すると、そのクライアントにウィンドウコンテンツのカスタムスナップショットが送信されます。これにより、遅れたサブスクライバは接続に追いつくことができます。

sol...という名前の Solace Systems コネクタ構成パラメータは、コネクタによって変更されずに Solace API に渡されます。これらのパラメータの詳細については、Solace のマニュアルを参照してください。

必要に応じて、solpassword パラメータを暗号化されたパスワードで構成できます。暗号化されたパスワードのバージョンは、システムにインストールする必要がある OpenSSL を使用して生成することができます。SAS Event Stream Processing システムの暗号化と認証のオーバーレイをインストールするときは、付属の OpenSSL 実行可能ファイルをインストールします。コンソールで次のコマンドを使用して、OpenSSL を起動し、暗号化されたパスワードを表示します。

echo "*solpassword*" | openssl enc -e -aes-256-cbc -a -salt -pass pass:"SASespSOLconnectorUsedByUser=*soluserid*"

暗号化されたパスワードを solpassword パラメータにコピーし、solpasswordencrypted パラメータを有効にします。

Solace パブリッシャコネクタインスタンスは、本文ではなく宛先属性からメッセージペイロードをコピーするように構成できます。ソースウィンドウスキーマは、コネクタによって書き込まれるインデックスとして機能する int64 キーフィールドで始まる必要があります。宛先属性からメッセージを解析するために、スラッシュで区切られた各エントリからフィールドデータが順次プルされ、同じ順序でインデックスフィールドに続くソースウィンドウフィールドにコピーされます。宛先属性からコピーするように構成されたコネクタインスタンスによって、対応するトピックで受信されたすべてのメッセージは、この方法で処理されます。依然本文が含まれているメッセージは、別のコネクタインスタンスによって 別のトピックで受信する必要があります。

Solace Systems コネクタで次のパラメータを使用します。

表 52 Solace Systems サブスクライバコネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
soluserid	アプライアンスとのコネクタのセッションを 認証するために必要なユーザー名を指定します。
solpassword	soluserid に関連付けられたパスワードを指定します。
solhostport	接続するアプライアンスを host:port の形式で指定します。
solvpn	セッションが接続するクライアントを割り当てるアプライアンスメッセージ VPN を指定します。
soltopic	パブリッシュ先の Solace 宛先トピックを指定します。
urlhostport	メタデータ要求を処理するために起動時にサブスクライブされているメタデータトピックの host:port フィールドを指定します。
numbufferedmsgs	スタンバイサブスクライバコネクタによってバッファされるメッセージの 最大数を指定します。超えた場合、最も古いメッセージは破棄されます。コネクタがアクティブになると、バッファがフラッシュされ、メッセージ ID シーケンスを維持するためにバッファされたメッセージが必要に応じてファブリックに送信されます。
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。

表 53 Solace Systems パブリッシャコネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
soluserid	アプライアンスとのコネクタのセッションを 認証するために必要なユーザー名を指定します。
solpassword	soluserid に関連付けられたパスワードを指定します。
solhostport	接続するアプライアンスを host:port の形式で指定します。
solvpn	セッションが接続するクライアントを割り当てるアプライアンスメッセージ VPN を指定します。
soltopic	サブスクライブする Solace トピックを指定します。
urlhostport	メタデータ要求を処理するために起動時にサブスクライブされているメタデータトピックの host:port フィールドを指定します。

表 54 Solace Systems サブスクライバコネクタのオプションパラメータ

パラメータ	説明
collapse	サブスクライバ出力をパブリッシュ可能にするために UPDATE_BLOCK イベントの変換を有効にします。
hotfailover	ホットフェイルオーバーモードを有効にします。
buspersistence	Solace メッセージ配信モードを保証メッセージングに設定します。デフォルトの配信モードはダイレクトメッセージングです。
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバによって受信されたイベントブロックからすべての 削除イベントを削除することを指定します。
protofile	イベントブロックを protobuf メッセージに変換するために使用される Google プロトコルバッファメッセージ定義を含む.proto ファイルを指定します。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した.proto ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
json	JSON メッセージとしてエンコードされたイベントブロックの転送を有効にします。

パラメータ	説明
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
solpasswordencrypted	solpassword が暗号化されることを指定します。

表 55 Solace Systems パブリッシャコネクタのオプションパラメータ

パラメータ	説明
buspersistence	アプライアンスにプロビジョニングされたトピックエンドポイントにバインドされる保証されたメッセージフローを作成し、パブリッシュされた保証されたメッセージが配信され、スプールされるようにします。ダイレクトメッセージングを使用してパブリッシュされたメッセージを受信する必要がないため、このフローはデフォルトで無効になっています。
buspersistencequeue	保証メッセージフローがバインドするキューの名前を指定します。
protofile	イベントブロックを protobuf メッセージに変換するために使用される Google プロトコルバッファメッセージ定義を含む.proto ファイルを指定します。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した.proto ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
json	JSON メッセージとしてエンコードされたイベントブロックの転送を有効にします。
publishwithupsert	opcode = Insert ではなく opcode = Upsert でビルドすることを指定します。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
solpasswordencrypted	solpassword が暗号化されることを指定します。
getmsgfromdestattr	ペイロードをメッセージ本文ではなく宛先属性から抽出します。

パラメータ	説明
transactional	getmsgfromdestattr が有効な場合は、イベントブロックの種類を transactional に設定します。デフォルトのイベントブロックの種類は標準です。
blocksize	getmsgfromdestattr が有効な場合は、パブリッシュされたイベントブロックに含めるイベントの数を指定します。デフォルト値は 1 です。
maxevents	パブリッシュするイベントの最大数を指定します。

Solace Systems アダプタの使用

Solace Systems アダプタは、ハードウェアベースの Solace ファブリック上のパブリッシュおよびサブスクライブ操作をサポートします。アダプタを使用するには、Solace ランタイムライブラリをインストールする必要があります。

サブスクライバの使用:

dfesp_sol_adapter -h *url* -k sub -n *numbufferedmsgs* -o *urlhostport* -p *solpassword* -t *soltopic* -u *soluserid* -v *solvpn* <-b > <-C [*configfilesection*]> <-d *dateformat* > <-E *tokenlocation* ><-f *protofile* > <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *protomsg* > <-V> <-X> <-y *logconfigfile* > <-Z *transportconfigfile*>

パブリッシャ:

dfesp_sol_adapter -h *url* -k pub -o *urlhostport* -p *solpassword* -t *soltopic* -u *soluserid* -v *solvpn* <-B *blocksize*> <-b > <-C [*configfilesection*]> <-D> <-d *dateformat* > <-E *tokenlocation* > <-e> <-f *protofile* > <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *protomsg* > <-Q> <-q *buspersistencequeue* > <-R> <-V><-X> <-Y maxevents> <-y *logconfigfile* > <-Z *transportconfigfile*>

パラメータ	説明
-B blocksize	getmsgfromdestattr が有効な場合は、パブリッシュされたイベントブロックに含めるイベントの数を指定します。デフォルト値は 1 です。
-b	Solace 保証メッセージングを使用します。デフォルトでは、Solace Direct Messaging が使用されます。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-D	ペイロードをメッセージ本文ではなく宛先属性から抽出します。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。

パラメータ	説明
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します
-e	getmsgfromdestattr が有効な場合は、イベントブロックの種類を transactional に設定します。デフォルトのイベントブロックの種類は normal です。
-f protofile	Google プロトコルバッファーのサポートに使用されるメッセージを含む**.proto** ファイルを指定します。
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP パブリッシュおよびサブスクライブ標準 URL を "dfESP://host:port/project/continuousquery/window"の形式で指定します。 サブスクライバに?snapshot= true | false を追加します。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに以下を追加します。 ■ ?collapse= true | false ■ ?rmretdel=true | false
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit() パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-k	サブスクライバ用には "sub"、 サブスクライバ用およびパブリッシャ向けの pub
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-m protomsg	protofile パラメータで指定された**.proto** ファイル内のメッセージ自体を指定します。
-n numbufferedmsgs	スタンバイサブスクライバコネクタによってバッファされるメッセージの 最大数を指定します。
-o urlhostport	コネクタがサブスクライブしているメタデータトピックの *host*:*port* フィールドを指定します。
-p solpassword	Solace のパスワードを指定します。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-q buspersistencequeue	Solace 保証メッセージングパブリッシャが使用するキュー名を指定します。
-R	opcode=Insert の代わりに opcode=Upsert を使用してイベントを作成します。

パラメータ	説明
-s solhostport	Solace*host*:*port* を指定します。
-t soltopic	Solace トピックを指定します。
-u soluserid	Solace ユーザー名を指定します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-v solvpn	Solace VPN の名前を指定します。
-X	solpassword が暗号化されることを指定します。
-Y maxevents	パブリッシュするイベントの最大数を指定します。
-y logconfigfile	ログ構成ファイルを指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

Teradata コネクタとアダプタの使用

Teradata コネクタの使用

Teradata コネクタは、Teradata Parallel Transporter (TPT) API を使用して、Teradata サーバーに対するサブスクライブ操作をサポートします。

Teradata Tools and Utilities (TTU)パッケージは、コネクタを実行しているプラットフォームにインストールする必要があります。実行時環境では、TTU 内の Teradata クライアントライブラリへのパスを定義する必要があります。ロード演算子を使用して Teradata ODBC をサポートするには、TTU 内の Teradata TeraGSS ライブラリへのパスも定義する必要があります。Teradata メッセージのロギングをサポートするには、NLSPATH 環境変数を適切に設定します。たとえば、/opt/teradata に TTU がインストールされている場合、NLSPATH を/opt/teradata/client/version/tbuild/msg64/%N と定義します。コネクタは TTU バージョン 14.10 を使用して認証されています。

デバッグのために、オプションの PC ベースの Teradata クライアントツールをインストールして、Teradata サーバー上のターゲットデータベーステーブルにアクセスすることができます。これらのツールには、両方とも ODBC を使用する GUI SQL ワークベンチ(Teradata SQL Assistant)と DBA/Admin ツール(Teradata Administrator)が含まれます。

3 つの TPT 演算子の 1 つを使用して、サーバー上のテーブル（stream、update、または load）にデータを書き込む必要があります。

オペレーター	説明
stream	標準的な ESP データベースサブスクライバコネクタのように動作しますが、TPT を使用してスループットを向上させました。イベントの挿入、更新、および削除をサポートします。サブスクライブされたウィンドウからイベントを受け取ると、それらをターゲットテーブルに書き込みます。必要な tdatainsertonly 構成パラメータを false に設定すると、TPT でシリアル化が自動的に有効になり、複数のセッションにわたって行データの順序が正しく維持されます。
update	挿入/更新/削除イベントをサポートしますが、それらをバッチモードでターゲットテーブルに書き込みます。バッチ期間は、必要なコネクタ構成パラメータです。待機時間は長くなりますが、この演算子はバッチ期間がより長い(たとえば、数秒ではなく数分)、より優れたスループットを提供します。
load	挿入イベントをサポートします。空のターゲットテーブルが必要です。最適化されたスループットを提供します。中間ステージングテーブルのペアを介してデータをずらします。これらのテーブル名と接続パラメータは､追加の必須コネクタ構成パラメータです。さらに、ステージングテーブルから最終ターゲットテーブルへの書き込みは、データベースコネクタで使用される汎用 ODBC ドライバを使用します。したがって、関連する接続文字列構成と odbc.ini ファイル指定が必要です。ステージングテーブルは、コネクタによって自動的に作成されます。コネクタの開始時にステージングテーブルおよび関連するエラーおよびログテーブルがすでに存在する場合、ステージングテーブルおよび始動時にそれらを自動的にドロップします。

必要に応じて、tdatauserpwd パラメータを暗号化されたパスワードで構成できます。暗号化されたパスワードのバージョンは、システムにインストールする必要がある OpenSSL を使用して生成することができます。SAS Event Stream Processing システム暗号化および認証オーバーレイをインストールしている場合は、付属の OpenSSL 実行可能ファイルを使用できます。コンソールで次のコマンドを使用して、OpenSSL を起動し、暗号化されたパスワードを表示します。

```
echo "tdatauserpwd" | openssl enc -e -aes-256-cbc -a -salt
-pass pass:"SASespTDATAconnectorUsedByUser=tdatausername"
```

暗号化されたパスワードを tdatauserpwd パラメータにコピーし、tdatauserpwdencrypted パラメータを有効にします。

表 56 Teradata コネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
desttablename	ターゲットテーブル名を指定します。
tdatausername	ターゲット Teradata サーバー上のユーザーアカウントのユーザー名を指定します。
tdatauserpwd	ターゲット Teradata サーバー上のユーザーアカウントのユーザーパスワードを指定します。
tdatatdpid	ターゲットの Teradata サーバー名を指定します。
tdatamaxsessions	TPT によって Teradata サーバーに作成されるセッションの最大数を指定します。
tdataminsessions	TPT によって Teradata サーバーに作成されるセッションの最小数を指定します。

パラメータ	説明
tdatadriver	演算子: stream、update、または load を指定します。
tdatainsertonly	サブスクライバイベントストリーム処理ウィンドウ内のイベントを挿入するかどうかを指定します。ロード演算子を使用する場合は true でなければなりません。
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。

表 57 Teradata コネクタのオプションパラメータ

パラメータ	説明
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバが受信したイベントブロックからすべての 削除イベントを削除します。
tdatabatchperiod	バッチ期間を秒単位で指定します。update 演算子を使用する場合は必須です。それ以外の場合は無視されます。
stage1tablename	最初のステージングテーブルを指定します。load 演算子を使用する場合に必要です。それ以外の場合は無視されます。
stage2tablename	2 番目のステージングテーブルを指定します。load 演算子を使用する場合に必要です。それ以外の場合は無視されます。
connectstring	ターゲットテーブルとステージングテーブルにアクセスするために 使用する接続文字列を指定します。 Teradata connectstring の要件については、“データベースコネクタの使用”を参照してください。
tdatatracelevel	現行作業ディレクトリの TRACE ファイルに書き込まれた Teradata メッセージのトレースレベルを指定します。TRACE ファイルの名前は *operator* 1.txt です。デフォルトは 1 (TD_OFF)です。有効な値は、2 (TD_OPER)、3 (TD_OPER_CLI)、4 (TD_OPER_OPCOMMON)、5 (TD_OPER_SPECIAL)、6 (TD_OPER_ALL)、7 (TD_GENERAL)、および 8 (TD_ROW)です。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
tdatauserpwdencrypted	tdatauserpwd が暗号化されることを指定します。

Teradata サブスクライバアダプタの使用

Teradata アダプタは、パフォーマンスを向上させるために Teradata Parallel Transporter を使用して、Teradata サーバーに対するサブスクライブ操作をサポートします。このアダプタを使用するには、Teradata Tools and Utilities (TTU)をインストールする必要があります。

注: 使用する前に、このコネクタへの参照をファイル**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.excluded** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.excluded** (Windows)から削除します。

注: 別のデータベースアダプタでは、Teradata プラットフォーム用の汎用 ODBC ドライバが使用されます。詳細については、“データベースアダプタの使用”を参照してください。

使用法:

dfesp_tdata_adapter -a *maxsessions* -d stream | update | load -h *url* -i true | false -n *minsessions* -s *servername* -t *tablename* -u *username* -x *userpwd* <-b *batchperiod* > <-C [*configfilesection*]> <-c *connectstring* > <-E *tokenlocation* > <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-q *stage1table* > <-r *stage2table* > <-V> <-X> <-y *logconfigfile* > <-Z *transportconfigfile*> <-z *tracelevel* >

パラメータ	説明
-a maxsessions	Teradata サーバー上の最大セッション数を指定します。Teradata セッションは、アプリケーションと Teradata Database 間の論理接続です。これにより、アプリケーションは Teradata Database との間でリクエストを送信したり、Teradata Database から応答を受け取ることができます。Teradata Database がユーザーのユーザー名とパスワードを受け入れると、セッションが確立されます。
-b batchperiod	バッチ期間を秒単位で指定します。このパラメータは、-d が更新演算子を指定する場合に必要です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-c connectstring	Teradata サーバー上のステージングテーブルとターゲットテーブルにアクセスするために ODBC ドライバが使用する接続文字列を指定します。Teradata connectstring の要件については、“データベースコネクタの使用”を参照してください。このパラメータは、-d load が指定されている場合に必要です。
-d stream | update | load	Teradata 演算子を指定します。これらの演算子の詳細については、Teradata ツールとユーティリティに付属のマニュアルを参照してください。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP サブスクライブ標準 URL を dfESP://host:port/project/continuousquery/window?snapshot=true |false 形式で指定します。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて ?collapse=true | false を追加します。 必要に応じて?rmretdel=true | false を追加します。
-i	サブスクライブされたウィンドウに挿入専用データが含まれることを指定します。

パラメータ	説明
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。C_dfESPpubsubInit()パブリッシュ/サブスクライブ API コールまたは engine initialize()コールで設定できるロギングレベルと同じ範囲のロギングレベルを使用します。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-n minsessions	Teradata サーバー上のセッションの最小数を指定します。
-q stage1table	Teradata サーバー上の最初のステージングテーブルの名前を指定します。このパラメータは、-d がロード演算子を指定する場合に必要です。
-r stage2table	Teradata サーバー上の 2 番目のステージングテーブルの名前を指定します。このパラメータは、-d がロード演算子を指定する場合に必要です。
-s servername	ターゲットの Teradata サーバーの名前を指定します。
-t tablename	Teradata サーバー上のターゲットテーブルの名前を指定します。
-u username	Teradata ユーザー名を指定します。
-V	致命的なエラーが報告された場合は、アダプタを再始動します。
-X	tdatauserpwd が暗号化されることを指定します。
-x userpwd	Teradata ユーザーのパスワードを指定します。
-y logconfigfile	ログ構成ファイルを指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。
-z tracelevel	現行作業ディレクトリのトレースファイルに 書き込まれた Teradata メッセージのトレースレベルを指定します。TRACE ファイルの名前は *operator* 1.txt です。デフォルト値は 1 (TD_OFF)です。有効な値は、2 (TD_OPER)、3 (TD_OPER_CLI)、4 (TD_OPER_OPCOMMON)、5 (TD_OPER_SPECIAL)、6 (TD_OPER_ALL)、7 (TD_GENERAL)、および 8 (TD_ROW)です。

Teradata Listener コネクタとアダプタの使用

Teradata Listener コネクタの使用

Teradata Listener コネクタは、SAS Event Stream Processing から Teradata Listener アプリケーションにデータを送信します。Teradata Listener は大量のリアルタイムデータストリームを取り込み、それらのストリームからのデータを Teradata、Aster、または Hadoop に持続させます。リスナーは、ターゲットシステムが使用可能になるまでデータをバッファリングすることにより、ターゲットシステムのサービスの中断に耐えます。

イベントのブロックは、REST API を使用して Teradata Listener インジェストサービスに送信されます。Teradata Listener コネクタは、リスナーインジェスト REST API (バージョン 1)をサポートしています。

データがリスナーインジェストサービスに配信された後、リスナーはデータをキューに入れ、そのリスナーソースストリームにサブスクライブしているすべてのターゲットにデータを 送信します。ターゲットは、1 つ以上の Teradata、Aster、または Hadoop システム、およびリスナーストリームサービスを使用して同じリスナーソースから読み取る他のアプリケーションにすることができます。

Teradata Listener コネクタは、SAS Event Stream Processing からイベントを収集し、contentType パラメータで指定されている JSON またはプレーンテキスト形式に変換します。リスナーインジェストサービスへの接続は、ingestUrl パラメータと SSLCert パラメータで定義されます。データが配信されるリスナーソース(データストリーム)は、sourceKey パラメータによって 識別されます。

Teradata Listener コネクタはサブスクライブ専用です。Teradata Listener から SAS Event Stream Processing にデータを送信するには、contentType=JSON の WebSocket コネクタを使用します。

注: Teradata Listener コネクタは SSL を使用して Teradata Listener Ingest サービスへの暗号化接続を確立するため、ランタイム環境で SSL ライブラリへのパスを定義する必要があります(例: Linux プラットフォームで LD_LIBRARY_PATH を指定)。SAS Event Stream Processing で使用される SSL ライブラリの詳細については、*SAS Event Stream Processing: セキュリティ*を参照してください。

表 58 Teradata Listener コネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
ingestUrl	リスナーインジェスト REST API (バージョン 1)の URL を指定します。
SSLCert	SSL 証明書を含むファイルへのパスを、リスナーインジェストサービスに安全に接続するように指定します。リスナーは TLS 1.2 を使用します。
sourceKey	SAS Event Stream Processing がデータを送信するリスナーソースフィードを識別するリスナーソースシークレットキーを指定します。Teradata Listener 管理者に連絡して、ソースの秘密鍵を入手してください。
snapshot	スナップショットデータを送信するかどうかを指定します。true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。

表 59 Teradata Listener コネクタのオプションパラメータ

パラメータ	説明
ingestBlocksize	1 つのリスナーインジェストメッセージで送信するデータ行の最大数を指定します。コネクタブロックサイズをソースウィンドウのブロックサイズに 合わせることをお勧めします。デフォルトのブロックサイズは 256 です。
contentType	SAS Event Stream Processing からリスナーに送信される、JSON または plaintext (カンマ区切り)のいずれかのデータの形式を指定します。デフォルトは“JSON”です。
ingestDelim	SAS Event Stream Processing からリスナーインジェスト REST API への複数行メッセージのデータ行を区切る文字を指定します。区切り文字は JSON の句読文字であってはなりません。デフォルトはチルダ（~）です。
-3 doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

Teradata Listener アダプタの使用

Teradata Listener サブスクライバアダプタは、いくつかのオプションのアダプタ専用パラメータに加えて、Teradata Listener コネクタと同じ機能を提供します。

使用法:

dfesp_tdlistener_adapter -h *url* -i *ingestUrl* -k *sourceKey* -s *SSLCert* <-b *ingestBlocksize*> <-C *configfilesection*> <-c JSON | plaintext> <-d *ingestDelim*> <-E *tokenlocation*> <-g *gdconfig*> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-V> <-y *logconfigfile*> <-Z *transportconfigfile*> <-3 *doubleprecision*>

パラメータ	説明
-b ingestBlocksize	1 つのリスナーインジェストメッセージで送信するデータ行の最大数を指定します。デフォルトは 256 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-c JSON | plaintext	SAS Event Stream Processing からリスナーに送信されるデータの形式を指定します。デフォルトは“JSON”です。
-d ingestDelim	SAS Event Stream Processing からリスナーインジェスト REST API への複数行メッセージのデータ行を区切る文字を指定します。区切り文字は JSON の句読文字であってはなりません。デフォルトはティルダ（~）です。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-g gdconfig	保証された配信構成ファイルを指定します。

パラメータ	説明
-h url	dfESP パブリッシュおよびサブスクライブ標準 URL を "dfESP://host:port/project/continuousquery/window"の形式で指定します。 サブスクライバに?snapshot= true | false を追加します。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて次の行を追加します。 ?rmretdel=true | false
-i ingestUrl	リスナーインジェスト REST API (バージョン 1)の URL を指定します。
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-k sourceKey	SAS Event Stream Processing がデータを送信するリスナソースフィードを識別するリスナソースシークレットキーを指定します。ソースキーを取得するには、Teradata Listener 管理者に連絡してください。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-n numbufferedmsgs	スタンバイサブスクライバコネクタによってバッファされるメッセージの 最大数を指定します。
-s SSLCert	リスナーインジェストサービスに安全に接続するための SSL 証明書を含むファイルへのパスを指定します。リスナーは TLS 1.2 を使用します。
-V	致命的なエラーが報告された場合は、アダプタを再始動します。
-y logconfigfile	ログ構成ファイルを指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。
-3 doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

Tervela コネクタとアダプタの使用

Tervela データファブリックコネクタの使用

Tervela データファブリックコネクタは、パブリッシュおよびサブスクライブ操作のために、ソフトウェアまたはハードウェアベースの Tervela データファブリックと通信します。

Tervela サブスクライバコネクタは、イベントブロックを受信し、次の Tervela トピック "SAS.ENGINES.*enginename.projectname.queryname.windowname*.OUT"にパブリッシュします。

Tervela パブリッシャコネクタは、次の Tervela のトピック "SAS.ENGINES.*enginename.projectname.queryname.windowname*.IN"からイベントブロックを読み取り、ソースウィンドウにインジェクトします。

Tervela データファブリックコネクタによって 提供されるバス接続の結果、エンジンは個々のパブリッシュ/サブスクライブ接続を管理する必要はありません。単一の ESP エンジンへの同時パブリッシュ/サブスクライブ接続の大容量が達成されます。

実行中のコネクタインスタンスをホストするプラットフォームに Tervela ランタイムライブラリをインストールする必要があります。ランタイム環境では、それらのライブラリへのパスを定義する必要があります(たとえば、Linux プラットフォームでは **LD_LIBRARY_PATH** を指定します)。

注: 使用する前に、このコネクタへの参照をファイル**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.excluded** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.excluded** (Windows)から削除します。

Tervela データファブリックコネクタの特徴は次のとおりです。

- バイナリイベントブロックで動作します。他のイベントブロック形式はサポートされていません。
- Tervela クライアントとして動作します。すべての Tervela データファブリック接続パラメータは、コネクタ構成パラメータとして必要です。

Tervela データファブリックコネクタを使用する前に、Tervela TPM Provisioning and Management System で次の項目を構成する必要があります。

- コネクタの tvauserid および tvapassword 構成パラメータと一致するクライアントのユーザー名とパスワード
- インバウンドとアウトバウンドのトピック文字列と関連するスキーマ
- クライアントユーザー名に関連付けられたエンタイトルメント権限をパブリッシュまたはサブスクライブします

コネクタが始動すると、トピック SAS.META.*tvaclientname* (ここで、*tvaclientname* はコネクタ構成パラメータ)にメッセージがパブリッシュされます。このメッセージには、次の情報が含まれています。

- ESP エンジン名の、ESP パブリッシュ/サブスクライブクライアントによって 潜在的に使用される *host*:*port* フィールドへのマッピング。*ホスト*:*port* ストリングは必須の urlhostport コネクタ構成パラメータであり、ファブリック上で使用されるトピック文字列のエンジン名で置き換えられます。
- コネクタに関連付けられたウィンドウのプロジェクト名、照会名、およびウィンドウ名、およびウィンドウのシリアライズされたスキーマ。

Tervela コネクタによって 実行されるすべてのメッセージングは、Tervela 保証配信モードを使用します。メッセージは Tervela TPE アプライアンスに保存されます。パブリッシャコネクタは、ファブリックに接続すると、最近の期間にわたって購読トピックに既にパブリッシュされたメッセージを受信します。既定では、パブリッシャコネクタはこの期間を 8 時間に設定します。これにより、パブリッシャは 1 日分のメッセージに追いつくことができます。このモードを使用するには、永続化されたメッセージを期限切れにするための他の自動化されたメカニズムがない場合、管理者による永続化データの定期的なパージが必要です。

Tervela サブスクライバコネクタはホットフェイルオーバーモードをサポートしています。コネクタのアクティブ/スタンバイ状態はファブリックと調整され、アクティブコネクタに障害が発生したときにスタンバイコネクタがアクティブになります。スイッチオーバーを実現するには、いくつかの条件を満たす必要があります。

- 関連するコネクタを実行している ESP エンジンのエンジン名はすべて同じでなければなりません。この一連の ESP エンジンはフェイルオーバーグループと呼ばれます。
- 関連するすべてのコネクタは、同じトピックのセットでアクティブでなければなりません。
- 関連するすべてのサブスクライバコネクタは、同じ **tvaclientname** で構成する必要があります。
- 関係するすべてのコネクタは、同時にメッセージフローを開始する必要があり、TPE は関連トピックのすべてのメッセージをパージします。これは、すべてのコネクタでメッセージ ID を同期させる必要があるためです。
- イベントブロックのインジェクタによってモデルに設定されるメッセージ ID は、順次であり、他のスタンバイコネクタで使用される ID と同期している必要があります。インジェクタが Tervela パブリッシャコネクタの場合、そのコネクタは ID = 1 で始まるすべてのインジェクトイベントブロックにメッセージ ID を設定します。

新しいサブスクライバコネクタがアクティブになると、スタンバイコネクタによるメッセージのバッファリングと、ファブリックとの再開されたフローの調整によって、発信メッセージフローが同期されたままになります。このメッセージバッファのサイズは、サブスクライバコネクタの必須パラメータです。

tva...という名前の Tervela コネクタ構成パラメータは、コネクタによって変更されずに Tervela API に渡されます。これらのパラメータの詳細については、Tervela のマニュアルを参照してください。

必要に応じて、tvapassword パラメータを暗号化されたパスワードで構成できます。暗号化されたパスワードのバージョンは、システムにインストールする必要がある OpenSSL を使用して生成することができます。SAS Event Stream Processing システム暗号化および認証オーバーレイをインストールしている場合は、付属の OpenSSL 実行可能ファイルを使用できます。コンソールで次のコマンドを使用して、OpenSSL を起動し、暗号化されたパスワードを表示します。

echo "*tvapassword*" | openssl enc -e -aes-256-cbc -a -salt -pass pass:"SASespTVAconnectorUsedByUser=*tvauserid*"

暗号化されたパスワードを tvapassword パラメータにコピーし、tvapasswordencrypted パラメータを有効にします。

Tervela コネクタでは以下のパラメータを使用してください。

表 60 Tervela サブスクライバコネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
tvauserid	Tervela TPM で定義されているユーザー名を指定します。パブリッシュトピックエンタイトルメント権限は、このユーザー名に関連付けられている必要があります。
tvapassword	tvauserid に関連付けられたパスワードを指定します。
tvaprimarytmx	プライマリ TMX のホスト名または IP アドレスを指定します。
tvatopic	サブスクライブするトピックのトピック名を指定します。このトピックは、GD サービス用の TPM で構成する必要があります。tvauserid には、TPM のこのトピックに対する保証された配信/サブスクライブ権を割り当てる必要があります。

パラメータ	説明
tvaclientname	Tervela 保証配信コンテキストに関連付けられたクライアント名を指定します。ホットフェールオーバーが有効な場合、この名前はフェイルオーバーグループ内の他のサブスクライバコネクタの tvaclientname と一致する必要があります。それ以外の場合、名前は Tervela コネクタのすべてのインスタンスで一意である必要があります。
tvamaxoutstand	Tervela ファブリックにパブリッシュできる承認されていないメッセージの最大数を指定します(事実上パブリケーションキャッシュのサイズ)。予想される送信速度の 2 倍にする必要があります。
numbufferedmsgs	スタンバイサブスクライバコネクタによってバッファされるメッセージの 最大数を指定します。超過すると、最も古いメッセージは破棄されます。コネクタがアクティブになると、バッファがフラッシュされ、メッセージ ID シーケンスを維持するためにバッファされたメッセージが必要に応じてファブリックに送信されます。
urlhostport	開始時にトピック SAS.META.*tvaclientname* のコネクタによってパブリッシュされたメタデータメッセージで送信される host/port 文字列を指定します。
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。

表 61 Tervela パブリッシャコネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
tvauserid	Tervela TPM で定義されているユーザー名を指定します。サブスクリプション - トピック権限付与権は、このユーザー名に関連付けられている必要があります。
tvapassword	tvauserid に関連付けられたパスワードを指定します。
tvaprimarytmx	プライマリ TMX のホスト名または IP アドレスを指定します。
tvatopic	パブリッシュするトピックのトピック名を指定します。このトピックは、GD サービスの TPM で構成する必要があります。
tvaclientname	Tervela 保証配信コンテキストに関連付けられたクライアント名を指定します。Tervela コネクタのすべてのインスタンスで 一意である必要があります。
tvasubname	作成される保証配信サブスクリプションに割り当てられた名前を指定します。この名前と tvaclientname の組み合わせは、ファブリックによって最後のサブスクリプション状態を再生するために使用されます。サブスクリプション状態が検出された場合は、そのサブスクリプション状態を以前の状態から再開するために使用されます。そうでない場合は、過去 8 時間以内に受信したメッセージの再生から開始して、新規サブスクライブが開始されます。

パラメータ	説明
urlhostport	開始時にトピック SAS.META.*tvaclientname* のコネクタによってパブリッシュされたメタデータメッセージで送信される host:port 文字列を指定します。

表 62 Tervela サブスクライバコネクタのオプションパラメータ

パラメータ	説明
collapse	サブスクライバ出力をパブリッシュ可能にするために UPDATE_BLOCK イベントの変換を有効にします。
hotfailover	ホットフェイルオーバーモードを有効にします。
tvasecondarytmx	セカンダリ TMX のホスト名または IP アドレスを指定します。フォールトトレラントペアにログインする場合は必須です。
tvalogfile	コネクタで、syslog (Linux または Solaris の場合)または Tervela.log (Windows の場合)ではなく指定したファイルにログが記録されます。
tvapubbwlimit	ファブリックにパブリッシュされるデータの最大帯域幅(Mbps)を指定します。デフォルトは 100 Mbps です。
tvapubrate	データメッセージがファブリックにパブリッシュされる速度を Kbps 単位で指定します。デフォルトは 30,000 メッセージ/秒です。
tvapubmsgexp	パブリッシュされたメッセージが Tervela API のキャッシュに保持される最大時間を秒単位で指定します。このキャッシュは、チャネルの出力信頼性ウィンドウの一部として使用されます(再送信が必要な場合)。デフォルト値は 1 秒です。
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバによって受信されたイベントブロックからすべての 削除イベントを削除することを指定します。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
protofile	Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。この定義は、イベントブロックを protobuf メッセージに変換するために使用されます。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
json	JSON メッセージとしてエンコードされたイベントブロックの転送を有効にします。

パラメータ	説明
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
tvapasswordencrypted	tvapassword が暗号化されることを指定します。

表 63 Tervela パブリッシャコネクタのオプションパラメータ

パラメータ	説明
tvasecondarytmx	セカンダリ TMX のホスト名または IP アドレスを指定します。フォールトトレラントペアにログインする場合に必要です。
tvalogfile	コネクタを syslog (Linux または Solaris の場合)または Tervela.log (Windows の場合)ではなく指定したファイルに記録します。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
protofile	Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。この定義は、イベントブロックを protobuf メッセージに変換するために使用されます。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
json	JSON メッセージとしてエンコードされたイベントブロックの転送を有効にします。
publishwithupsert	opcode = Insert ではなく opcode = Upsert でイベントを構築することを指定します。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
tvapasswordencrypted	tvapassword が暗号化されることを指定します。
maxevents	パブリッシュするイベントの最大数を指定します。

Tervela データファブリックアダプタの使用

Tervela アダプタは、ハードウェアベースまたはソフトウェアベースの Tervela ファブリック上のパブリッシュおよびサブスクライブ操作をサポートします。このアダプタを使用するには、Tervela ランタイムライブラリをインストールする必要があります。

サブスクライバの使用:

dfesp_tva_adapter -b *numbufferedmsgs* -c *tvaclientname* -f *tvatopic* -h *url* -k sub -m *tvamaxoutstand* -o *urlhostport* -p *tvapassword* -t *tvaprimarytmx* -u *tvauserid* <-a *protofile* > <-C [*configfilesection*]> <-D *dateformat* > <-d *protomsg* > <-E *tokenlocation* > <-e *tvapubmsgexp* > <-g *gdconfig* > <-i *tvalogfile* > <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-r *tvapubrate* > <-s *tvasecondarytmx* > <-V> <-w *tvapubbwlimit* > <-X> <-y *logconfigfile* > <-Z *transportconfigfile*>

パブリッシャ:

dfesp_tva_adapter -c *tvaclientname* -f *tvatopic* -h *url* -k pub -n *tvasubname* -o *urlhostport* -p *tvapassword* -t *tvaprimarytmx* -u *tvauserid* <-a *protofile* > <-C [*configfilesection*]> <-D *dateformat* > <-E *tokenlocation* > <-g *gdconfig* > <-i *tvalogfile* > <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *protomsg* > <-Q> <-R> <-s *tvasecondarytmx* > <-V> <-X> <-Y *maxevents*> <-y *logconfigfile* > <-Z *transportconfigfile*>

パラメータ	説明
-a protofile	Google プロトコルバッファーのサポートに使用されるメッセージを含む**.proto** ファイルを指定します。
-b numbufferedmsgs	スタンバイサブスクライバコネクタによってバッファされるメッセージの 最大数を指定します。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-c tvaclientname	Tervela クライアント名を指定します。
-D dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-d protomsg	protofile パラメータで指定された**.proto** ファイル内のメッセージ自体を指定します。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e tvapubmsgexp	パブリッシュされたメッセージをキャッシュするための Tervela の最大時間(秒)を指定します。デフォルト値は 1 です。
-f tvatopic	Tervela トピックを指定します。

パラメータ	説明
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP パブリッシュおよびサブスクライブ標準 URL を "dfESP://host:port/project/continuousquery/window"の形式で指定します。 サブスクライバに次の行を追加します。?snapshot=true | false ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに以下を追加します。 ■ ?collapse=true | false ■ ?rmretdel=true | false
-i tvalogfile	Tervela ログファイルを指定します。デフォルトは“syslog”です。
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-k	サブスクライバの使用の sub とパブリッシャの使用のための pub を指定します。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-m tvamaxoutstand	Tervela の未確認メッセージの最大数を指定します。
-o urlhostport	コネクタメタデータメッセージで 送信される *host*:*port* 文字列を指定します。
-p tvapassword	Tervela のパスワードを指定します。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-R	opcode=Insert ではなく opcode=Upsert でイベントを作成します。
-r tvapubrate	Tervela パブリッシュレート(Kbps)を指定します。デフォルト値は 30 です。
-s tvasecondarytmx	Tervela セカンダリ TMX を指定します。
-t tvaprimarytmx	Tervela プライマリ TMX を指定します。
-u tvauserid	Tervela ユーザー名を指定します。
-V	致命的なエラーが報告された場合にアダプタを再起動することを指定します。
-w tvapubbwlimit	パブリッシュされたデータの Tervela 最大帯域幅(Mbps)を指定します。デフォルト値は 100 です。

パラメータ	説明
-X	tvapassword が暗号化されることを指定します。
-Y maxevents	パブリッシュするイベントの最大数を指定します。
-y logconfigfile	ログ構成ファイルを指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

Tibco Rendezvous コネクタとアダプタの使用

Tibco Rendezvous (RV)コネクタの使用

Tibco Rendezvous (RV)コネクタは、Tibco RV デーモンを介してパブリッシュおよびサブスクライブ操作用に Tibco RV API をサポートします。サブスクライバは、イベントブロックを受信し、それらを Tibco RV 題名にパブリッシュします。パブリッシャは、受信したイベントブロックをソースウィンドウにインジェクトする Tibco RV サブスクライバです。

Tibco RV ランタイムライブラリは、コネクタの実行中のインスタンスをホストするプラットフォームにインストールする必要があります。ランタイム環境では、それらのライブラリへのパスを定義する必要があります(たとえば、Linux プラットフォームでは **LD_LIBRARY_PATH** を指定します)。

注: 使用する前に、このコネクタへの参照をファイル**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.excluded** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.excluded** (Windows)から削除します。

コネクタが RVD デーモンを実行できるように、システムパスは **Tibco/RV/bin** ディレクトリを指している必要があります。

Tibco RV コネクタで使用される題名は、必要なコネクタパラメータです。Tibco RV サブスクライバには、Tibco RV にイベントをパブリッシュするために使用されるメッセージ形式を定義するパラメータも必要です。Tibco RV パブリッシャは、Tibco RV サブスクライバが作成したメッセージの種類をすべて使用できます。

デフォルトでは、Tibco RV コネクタは、Tibco RV デーモンがコネクターと同じプラットフォームで実行されていることを前提としています。また、コネクタ tibrvdaemon 構成パラメータを指定して、リモートデーモンを使用することもできます。

同様に、オプションの tibrvservice および tibrvnetwork パラメータを指定して、コネクタで使用される Rendezvous サービスおよびネットワークインタフェースを制御することもできます。詳細については、Tibco RV のマニュアルを参照してください。

Tibco RV コネクタは、メッセージ配信用のデフォルトのマルチキャストプロトコルに依存しています。Tibco RV デーモンとの間で送受信されるメッセージの信頼性間隔は、デーモンが使用している値から継承されます。

Tibco RV コネクタで次のパラメータを使用します。

表 64 Tibco RV サブスクライバコネクタの必須パラメータサブスクライバコネクタ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
tibrvsubject	Tibco RV の題名を指定します。
tibrvtype	**binary**、**CSV**、**JSON**、またはサブスクライブされたウィンドウスキーマ内の文字列フィールドの名前を指定します。
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。

表 65 Tibco RV パブリッシャコネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
tibrvsubject	Tibco RV の題名を指定します。
tibrvtype	**binary**、**CSV**、**JSON**、または **opaquestring** を指定します。**opaquestring** の場合、ソースウィンドウスキーマは index:int64,message:string とみなされます。

表 66 Tibco RV サブスクライバコネクタのオプションパラメータ

パラメータ	説明
collapse	サブスクライバ出力をパブリッシュ可能にするために UPDATE_BLOCK イベントの変換を有効にします。
tibrvservice	コネクタによって作成された Tibco RV トランスポートで使用される Rendezvous サービスを指定します。デフォルトのサービス名は“rendezvous”です。
tibrvnetwork	コネクタによって作成された Tibco RV トランスポートで使用されるネットワークインターフェイスを指定します。デフォルトのネットワークは、コネクタが使用するデーモンの種類によって異なります。
tibrvdaemon	コネクタで使用される Rendezvous デーモンを指定します。デフォルトは、ローカルデーモンによって作成されるデフォルトのソケットです。

パラメータ	説明
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバによって受信されたイベントブロックからすべての削除イベントを削除することを指定します。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config**(Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config**(Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
protofile	Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。この定義は、イベントブロックを protobuf メッセージに変換するために使用されます。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
csvmsgperevent	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
csvmsgpereventblock	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

表 67 Tibco RV パブリッシャコネクタのオプションパラメータ

パラメータ	説明
blocksize	パブリッシュされたイベントブロックに含めるイベントの数を指定します。デフォルト値は 1 です。
transactional	イベントブロックの種類をトランザクションに設定します。デフォルトのイベントブロックの種類は標準です。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。

パラメータ	説明
tibrvservice	コネクタによって作成された Tibco RV トランスポートで使用される Rendezvous サービスを指定します。デフォルトのサービス名は rendezvous です。
tibrvnetwork	コネクタによって作成された Tibco RV トランスポートで使用されるネットワークインターフェイスを指定します。デフォルトのネットワークは、コネクタが使用するデーモンの種類によって異なります。
tibrvdaemon	コネクタで使用される Rendezvous デーモンを指定します。デフォルトは、ローカルデーモンによって作成されるデフォルトのソケットです。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
ignorecsvparseerrors	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。
protofile	Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。この定義は、イベントブロックを protobuf メッセージに変換するために使用されます。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。
noautogenfield	入力イベントに、ソースウィンドウによって自動生成されるキーフィールドがないことを指定します。
publishwithupsert	opcode=Insert ではなく opcode=Upsert でイベントを構築することを指定します。
addcsvopcode	CSV イベントを入力するための演算コードとカンマを先頭に付けます。演算コードは、publishwithupsert が有効になっていない場合は挿入です。
addcsvflags	入力 CSV イベントに挿入するイベントの種類を指定します(カンマ)。有効な値は normal と partialupdate です。
maxevents	パブリッシュするイベントの最大数を指定します。

Tibco Rendezvous (RV)アダプタの使用

Tibco RV アダプタは、Tibco RV デーモンを使用してパブリッシュおよびサブスクライブ操作をサポートします。このアダプタを使用するには、Tibco RV ランタイムライブラリをインストールする必要があります。

サブスクライバの使用:

dfesp_tibrv_adapter -f *tibrvsubject* -h *url* -k sub -s *tibrvservice* -t *tibrvtype* <-A> <-a *protofile* > <-b *protomsg* > <-C [*configfilesection*]> <-D *csvfielddelimiter* > <-d *dateformat* > <-E *tokenlocation* > <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal | off> <-L> <-l native | solace | tervela | rabbitmq | kafka> <-M> <-m *tibrvdaemon* > <-n *tibrvnetwork* > <-R> <-V> <-y *logconfigfile* > <-Z *transportconfigfile*> <-3 *doubleprecision*>

パブリッシャ:

dfesp_tibrv_adapter -f *tibrvsubject* -h *url* -k pub -s *tibrvservice* -t *tibrvtype* <-a *protofile* > <-b *protomsg* > <-C [*configfilesection*]> <-d *dateformat* > <-E *tokenlocation* > <-e> <-F *eventtype*> <-g *gdconfig* > <-I> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *tibrvdaemon* > <-n *tibrvnetwork* > <-O> <-Q> <-r *blocksize* > <-V> <-Y *maxevents*> <-y *logconfigfile* ><-Z *transportconfigfile*>

パラメータ	説明
-A	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。
-a protofile	Google プロトコルバッファーのサポートに使用されるメッセージを含む**.proto** ファイルを指定します。
-b blocksize	ブロックサイズを指定します。デフォルト値は 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-D csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e	イベントがトランザクションであることを指定します。
-F eventtype	入力 CSV イベントに挿入するイベントの種類をカンマで指定します。有効な値は "normal"と"partialupdate"です。
-f tibrvsubject	Tibco RV の題名を指定します。
-g gdconfig	保証された配信構成ファイルを指定します。

パラメータ	説明
-h url	dfESP パブリッシュおよびサブスクライブ標準 URL を "dfESP://host:port/project/continuousquery/window”の形式で指定します。 サブスクライバに"?snapshot= true | false "を追加します。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに以下を追加します。 ■ ?collapse= true | false ■ ?rmretdel= true | false
-I	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-k	サブスクライバの使用の sub を指定し、パブリッシャを使用するための pub を指定します。
-L	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の**ネイティブ**トランスポートの代わりに **solace**、**tervela**、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-M	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
-m tibrvdaemon	Tibco RV デーモンを指定します。
-n tibrvnetwork	Tibco RV ネットワークを指定します。
-O	CSV イベントを入力するための演算コードとカンマを先頭に付けます。演算コードは、-R が有効になっていない場合は挿入されます。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-R	挿入の代わりに opcode=Upsert を使用してイベントを作成します。
-r protomsg	protofile パラメータで指定された**.proto** ファイル内のメッセージ自体を指定します。
-s tibrvservice	Tibco RV サービスを指定します。

パラメータ	説明
-t tibrvtype	binary、CSV、または JSON を指定します。opaquestring はパブリッシャでサポートされています。opaquestring の場合、ソースウィンドウスキーマは "index:int64,message:string"とみなされます。サブスクライバの場合、サブスクライブされたウィンドウスキーマ内の文字列フィールドの名前もサポートされます。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-Y maxevents	パブリッシュするイベントの最大数を指定します。
-y logconfigfile	ログ構成ファイルを指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。
-3 doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

タイマコネクタとアダプタの使用

タイマパブリッシャコネクタの使用

タイマパブリッシャコネクタは、一定の間隔を置いてトリガイベントを生成してパブリッシュします。間隔の終了ごとにスキーマ<id*:int64,time:stamp,label:string>のイベントが生成されます。コネクタのソースウィンドウは、このスキーマに従う必要があります。

タイマ間隔は、コネクタパラメータで指定されているように、次のように定義されます。

- 開始時間。
- 間隔。任意の数の秒、分、時間、日、週、月、年を指定できます。
- 複数のタイマコネクタインスタンスがある場合に、タイマソースを識別するラベル文字列。指定しない場合、デフォルトはコネクタの名前です。

開始時間は、将来または過去の日付にすることができます。将来の場合、これは開始時間です。過去の場合は、基準時間の意味合いが強くなります。

時間が 2017-08-01 11:25:32 に設定され、間隔が 1 時間に設定されているとします。モデルの開始する現在時刻が 2017-09-15 01:12:31 の場合、最初のイベントは 2017-09-15 01:25:32、2 番目は 2017-09-15 02:25:32、というように生成されていきます。

表 68 タイマコネクタの必須パラメータ

パラメータ	説明
type	パブリッシュまたはサブスクライブを指定します。
basetime	timeformat パラメータで定義された形式で開始時間を指定します。
interval	unit パラメータで定義された単位で間隔長を指定します。
unit	interval パラメータの単位を指定します。単位には second | minute | hour | day | week | month | year が含まれます。

表 69 タイマコネクタのオプションパラメータ

パラメータ	説明
label	ソースウィンドウの“ラベル”フィールドに書き込む文字列を指定します。デフォルト値はコネクタ名です。
timeformat	basetime パラメータの形式を指定します。デフォルト値は“%Y-%m-%d %H:%M:%S”です。
transactional	イベントブロックの種類を transactional に設定します。デフォルト値は normal です。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
publishwithupsert	opcode = Insert の代わりに opcode = Upsert を使用してイベントを作成します。
maxevents	パブリッシュするイベントの最大数を指定します。

タイマパブリッシャアダプタの使用

タイマパブリッシャアダプタでは、タイマパブリッシャコネクタと同じ機能が提供され、アダプタのみのオプションパラメータがいくつか追加されています。

使用法:

dfesp_timer_adapter -b *basetime* -h *url* -i *interval* -u second | minute | hour | day | week | month | year <-C *configfilesection*> <-E *tokenlocation*> <-e> <-f *timeformat*> <-g *gdconfig*> <-j trace | debug | info | warn | error | fatal | off> <-L *label*> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxevents*> <-Q> <-R> <-V> <-y *logconfigfile*> <-Z *transportconfigfile*>

使用法:

パラメータ	説明
-b basetime	timeformat パラメータで定義された形式で開始時間を指定します。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e	イベントが transactional であることを指定します。
-f timeformat	basetime パラメータの形式を指定します。デフォルト値は%Y-%m-%d %H:%M:%S です。
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP のパブリッシュとサブスクライブの標準 URL を dfESP://host:port/project/continuousquery/window の形式で指定します。
-i interval	unit パラメータで定義された単位で間隔長を指定します。
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-L label	ソースウィンドウの label フィールドに書き込み文字列を指定します。デフォルト値はコネクタ名です。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-m maxevents	パブリッシュするイベントの最大数を指定します。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-R	opcode=Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-u second | minute | hour | day | week | month | year	interval パラメータの単位を指定します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-y logconfigfile	ログ構成ファイルを指定します。

パラメータ	説明
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。

Twitter パブリッシャアダプタの使用

Twitter アダプタはパブリッシャのみです。Twitter のストリームを消費し、イベントブロックをエンジンのソースウィンドウにインジェクトします。Twitter アダプタは以下の機能を提供します:

- Twitter のストリームをサブスクライブし、ツイートするとすぐにツイートの流れを続けます。
- Twitter が提供する次の標準機能を使用して、入力ストリームをフィルタリングします。
  - 特定のユーザーをフォローします。
  - 特定のキーワードを追跡します。
  - 特定の地理的な場所をフィルタリングします。
  - 特定の言語をフィルタリングします。
  - ランダムに生成されたツイートを受信します。
  - RT ツイートのみを受信します。これには、特定の Twitter アカウントの承認が必要です。
  - リンクを含むツイートのみを受信します。これには、特定の Twitter アカウントの承認が必要です。
  - 完全な "消防ホース"を受け取ります。これには、特定の Twitter アカウントの承認が必要です。
- 受け取ったツイートをソースウィンドウにパブリッシュして、モデル内で処理します。

使用法:

dfesp_twitter_publisher -m sample | filter | links | firehose | retweet -u *url* <-a *languages* > < -b *blocksize* > <-C *prevcount* > <-c [*configfilesection*]> <-D *dumpfile* > <-d *dateformat* > <-f *followlist* > < -g *gdconfigfile* > <-k *tracklist* > < -l native | solace | tervela | rabbitmq | kafka> <-O *tokenlocation* > <-o severe | warning | info> < -p *locations* > <-s> <-T *twitterpropfile* > <-t> <-V> <-W *maxevents*> <-Z *transportconfigfile*>

パラメータ	説明
-a languagelist	指定された言語で書かれていると検出されたツイートのみを返す BCP 47 言語識別子のコンマ区切りリストを指定します。たとえば、"en、fr"で接続すると、英語またはフランス語で検出されたツイートだけがストリーミングされます。 **注:** このパラメータは、フィルタアクセス方式のフィルタ述語の少なくとも 1 つと組み合わせて使用する必要があります。-a languagelist パラメータは、-m filter 引数を使用してコマンドラインで指定したフィルタアクセス方式でのみサポートされています。フィルタアクセス方式には、サポートされているフィルタ述語のうち少なくとも 1 つが必要です。詳細は、表 70 を参照してください。
-b blocksize	イベントブロックごとのイベント数を指定します。
-C prevcount	フィルタ、リンク、またはリトウェットメソッドで使用されます。ライブストリームに移行する前にストリーミングする前のステータスの数を示します。デフォルトは 0 です。指定できる値は、1〜150000 または-1〜-150000 の整数です。正の数を指定すると、バックフィルがクライアントに配信された後にストリームがライブ値に移行します。負の数を指定すると、バックフィルがクライアントに配信された後にストリームが切断されます。これはデバッグに役立つ可能性があります。このパラメータを使用するにはアクセス権が必要です。 詳細については、ストリーミング API 要求パラメータを参照してください。
-c [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/javaadapters.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\javaadapters.config** (Windows)にセクションの名前を指定します。
-D dumpfile	再生のためにバイナリイベントブロックをダンプするファイルの絶対パスを指定します。イベントブロックを再生するには、dfESP_fs_adapter を使用します。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-f followlist	"user1、user2、user3"の形式でフィルタメソッドに従うユーザー ID のリストを指定します。デフォルトのアクセスレベルでは、最大 400 のフォロワーユーザー ID が許可されます。詳細については、ストリーミング API 要求パラメータを参照してください。
-g gdconfigfile	クライアントの保証された配信構成ファイルを指定します。

パラメータ	説明
-k tracklist	フィルタメソッドでトラッキングするキーワードのリストを "keyword1、keyword2、keyword3"の形式で指定します。デフォルトのアクセスレベルでは、最大 200 のトラックのキーワードが許可されます。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、*"Java クライアント用の代替トランスポートライブラリの使用" (SAS Event Stream Processing: パブリッシュ/サブスクライブ API)*で指定された必要なクライアント構成ファイルを使用します。
-m sample | filter | links | firehose | retweet	Twitter ストリームをサブスクライブするためのメソッドの種類を指定します。次の表を参照してください。
-O tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します
-o severe | warning | info	アプリケーションのログレベルを指定します。
-p locationlist	カンマ区切りの経度、緯度のペアのリスト形式で、フィルタメソッドで追跡する場所のリストを指定します。各ペアは、境界ボックスのセットを指定します。境界ボックスの南西の角が最初に来ます。サンプル:サンフランシスコの場合は"-122.75,36.8、-121.75,37.8"です。境界ボックスは、他のフィルタパラメータのフィルタとして機能しないことに注意してください。デフォルトのアクセスレベルでは、1 度のロケーションボックスを最大 10 個まで使用できます。
-s	Insert の代わりに opcode=Upsert を使用してイベントを作成します。 **javaadapters.config** パラメータ名:publishwithupsert
-T twitterpropfile	Twitter 構成パラメータを解析するための構成ディレクトリ内のファイルの名前を指定します。デフォルトは **twitterpublisher.properties** です。
-t	イベントブロックがトランザクションであることを指定します。デフォルトのイベントブロックの種類は標準です。 **javaadapters.config** パラメータ名: transactional
-u url	パブリッシュする標準 URL を " **dfESP://*host*:*port*/*project*/*continuousquery*/*window***"の形式で指定します。
-V	致命的なエラーが報告された場合は、アダプタを再始動します。 j**avaadapters.config** パラメータ名:restartonerror
-W maxevents	パブリッシュするイベントの最大数を指定します。

パラメータ	説明
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、*"Kafka によるパブリッシャアダプタのフェイルオーバー" (SAS Event Stream Processing: 高度なトピック)*を参照してください。

Twitter はストリームにアクセスするための制限を設定し、ユーザーアカウントのアクセスレベルに応じていくつかの方法を使用します。

表 70 Twitter ストリームアクセス方式

アクセス方式	制限事項
sample	すべてのパブリッシュステータスの無作為サンプルでリスニングを開始します。デフォルトのアクセスレベルは、firehose のほんの一部です。"ardenhose"アクセスレベルは、統計的に重要なサンプルを必要とするデータマイニングおよびリサーチアプリケーションに適した割合を提供します。
filter	1 つまたは複数のフィルタ述語に一致するパブリッシュ状態を消費し始めます。少なくとも 1 つの述語パラメータ、フォロー、位置、または追跡を指定する必要があります。ほとんどのクライアントがストリーミング API への単一の接続を使用できるようにする複数のパラメータを指定できます。長いパラメータを URL に配置すると、過剰な URL の長さに対して要求が拒否される可能性があります。デフォルトのアクセスレベルでは、最大 200 のトラックキーワード、400 人のフォロワーユーザー ID、10 の 1 度ロケーションボックスが可能です。アクセスレベルを上げると、次のことが可能になります。 ■ 80,000 人のフォロワーユーザー ID ("シャドー"役割) ■ 400,000 人のフォロワーユーザー ID ("birddog"役割) ■ 10,000 トラックのキーワード("制限されたトラック"の役割) ■ 200,000 トラックのキーワード("パートナートラック"の役割) ■ 200 の 10 度ロケーションボックス("locRestricted"役割) トラックアクセスレベルを上げると、ストリームを制限する前にステータスの割合が高くなります。パラメータ指定の詳細については、https://developer.twitter.com/en/docs を参照してください。
firehose	すべてのパブリッシュ状態でリッスンを開始します。Twitter で承認された当事者のみ利用可能で、アクセスするには署名付きの同意書が必要です。
retweet	すべてのリツイートでリスニングを開始します。リツイートストリームは一般的に利用可能なリソースではありません。このレベルのアクセスが必要なアプリケーションはほとんどありません。他のリソースとさまざまなアクセスレベルの組み合わせを創造的に使用することで、ほぼすべてのアプリケーションのユースケースを 満たすことができます。
link	リンクを含むパブリッシュ状態をすべてリッスンします。Twitter で承認された当事者のみ利用可能で、アクセスするには署名付きの同意書が必要です

Twitter アダプタを実行する前に、twitter4j.org から twitter4j-core と twitter4j-stream の JAR ファイルをダウンロードし、**$DFESP_HOME/lib** にコピーします。バージョン 4.0.2 および 4.0.4 が認定されています。

Twitter アダプタによる Twitter ユーザーアカウントの使用を承認するには、SAS Twitter アプリ ID のかわりに Twitter ユーザーアカウントへのアクセスを許可する必要があります。SAS Event Stream Processing 構成ディレクトリにある Twitter プロパティファイルには、ユーザーアカウントがアダプタにアクセスするための暗号化アクセストークンが含まれている必要があります。デフォルトでは、Twitter プロパティファイルの名前は **twitterpublisher.properties** で、アクセストークンは含まれません。暗号化アクセストークンを生成してプロパティファイルに書き込むには、**$DFESP_HOME/bin** にある dfesp_twitter_auth ユーティリティを実行します。トークンがプロパティファイルに含まれたら、アダプタが正常に実行されることを確認します。同じ Twitter アカウントに対して dfesp_twitter_auth ユーティリティを複数回実行する必要はありません。

プロキシサーバー、API ストリーム URL、またはトレース設定などのその他の設定を追加する必要がある場合は、http://twitter4j.org/en/configuration.html#Streaming を参照して、Twitter のプロパティファイルを適宜編集してください。

Twitter アダプタがパブリッシュするモデルのソースウィンドウには、次のスキーマが必要です。

```
tw_ID*:int64,tw_AuthorScreenName:string,tw_AuthorId:int64,tw_AuthorDescrip
tion:string,tw_AuthorFavouritesCount:int32,tw_AuthorFollowersCount:int32,tw
_AuthorFriendsCount:int32,tw_AuthorProfileImageURL:string,tw_AuthorLang:str
ing,tw_AuthorLocation:string,tw_AuthorName:string,tw_AuthorTimeZone:string,
tw_AuthorURL:string,tw_AuthorStatusesCount:int32,tw_AuthorListedCount:int32
,tw_Text:string,tw_CreatedAt:stamp,tw_CurrentUserRetweetId:int64,tw_Favorit
eCount:int32,tw_GeolocLatitude:double,tw_GeolocLongitude:double,tw_InReplyT
oScreenName:string,tw_InReplyToStatusId:int64,tw_InReplyToUserId:int64,tw_L
ang:string,tw_RetweetCount:int32,tw_RetweetedStatusId:int64,tw_SourceText:s
tring,tw_Source:string,tw_isFavorited:int32,tw_isPossiblySensitive:int32,tw
_isRetweet:int32,tw_isRetweeted:int32,tw_isTruncated:int32,tw_PlaceFullName
:string,tw_MentionedUserNames:string,tw_MentionedUserScreenNames:string,tw_
MentionedUserIds:string,tw_StatusLinks:string,tw_StatusTags:string,f_method
:string,f_followed_IDs:string,f_tracked_terms:string,f_GeoLocations:string,
f_Languages:string
```

フィールド	種類	説明
tw_ID	int64	ステータスの ID
tw_AuthorScreenName	string	ステータスに関連付けられたユーザー ID の画面名
tw_AuthorId	int64	ステータスに関連付けられたユーザー ID。
tw_AuthorDescription	string	オーサーの説明
tw_AuthorFavouritesCount	int32	オーサーのお気に入りの数
tw_AuthorFollowersCount	int32	オーサーのフォロワーの数
tw_AuthorFriendsCount	int32	オーサーの友人の数
tw_AuthorProfileImageURL	string	オーサーのプロフィール画像の URL
tw_AuthorLang	string	オーサーの優先言語
tw_AuthorLocation	string	オーサーの所在地

フィールド	種類	説明
tw_AuthorName	string	オーサーの名前
tw_AuthorTimeZone	string	オーサーのタイムゾーン
tw_AuthorURL	string	オーサーの URL
tw_AuthorStatusesCount	int32	オーサーのステータスカウント
tw_AuthorListedCount	int32	オーサーがリストされているパブリッシュリストの数。カウントが利用できない場合は-1。
tw_Text	string	ステータスのテキスト
tw_CreatedAt		ステータスの作成日
tw_CurrentUserRetweetId	int64	このツイートの認証ユーザーのリツイート ID、またはこの機能が有効になる前にツイートが作成された場合は 1L。
tw_FavoriteCount	int32	このツイートが Twitter ユーザーによって"お気に入り"にされた回数。
tw_GeolocLatitude	double	このツイートが参照する場所(使用可能な場合)(ヌルでもかまいません)。
tw_GeolocLongitude	double	このツイートが参照する場所(使用可能な場合)(ヌルでもかまいません)。
tw_InReplyToScreenName	string	ステータスのスクリーン名。このステータスは、以下に返信しています。
tw_InReplyToStatusId	int64	ステータスの ID。このステータスは返信中です。
tw_InReplyToUserId	int64	ユーザーの ID。このステータスは返信しています。
tw_Lang	string	2 文字の ISO 言語コード(存在する場合)。
tw_RetweetCount	int32	このツイートが retweet された回数、またはこの機能が有効になる前にツイートが作成されたときは-1 です。ツイートを受信すると、リアルタイムでこの値は prevcount パラメータが指定されている場合を除いて常に 0 です。
tw_RetweetedStatusId	int64	ステータスの ID です。このステータスは、以下のリツイートです。
tw_SourceText	string	ソース HTML タグの形式で、このステータスがパブリッシュされたソース。
tw_Source	string	このステータスが発行されたソースデバイスまたはアプリケーション。
tw_isFavorited	int32	ステータスがお気に入りの場合は true。

フィールド	種類	説明
tw_isPossiblySensitive	int32	ステータスに機密として識別されるリンクが含まれている場合は true です。
tw_isRetweet	int32	ステータスが retweet の場合は true。
tw_isRetweeted	int32	ステータスが retweeted の場合は true。
tw_isTruncated	int32	ステータスが切り捨てられた場合は true。
tw_PlaceFullName	string	このステータスに付けられた場所。
tw_MentionedUserNames	string	ツイートに記載されているユーザーの名前のリスト。
tw_MentionedUserScreenNames	string	ツイートに記載されているユーザーの画面名のリスト。
tw_MentionedUserIds	string	ツイートに記載されているユーザーの ID のリスト。
tw_StatusLinks	string	ツイートに記載されているリンクのリスト。
tw_StatusTags	string	ハッシュタグのリスト #
f_method	string	Twitter ストリームをクエリするために使用されるアダプタの followlist パラメータ(source、filter、retweet、firehose、links)で指定されたメソッド。
f_followed_IDs	string	アダプタの followlist パラメータで指定されたユーザー ID のリストで、Twitter ストリームをフィルタリングします。
f_tracked_terms	string	アダプタの tracklist パラメータで指定されたキーワードのリストで、Twitter ストリームをフィルタリングします。
f_GeoLocations	string	Twitter ストリームをフィルタリングするためにアダプタの locations パラメータで指定されたジオロケーションのリスト。
f_Languages	string	Twitter ストリームをフィルタリングするために、アダプタの languages パラメータで指定された言語のリスト。

Twitter Gnip パブリッシャアダプタの使用

アダプタの使用

Twitter Gnip アダプタは、SAS Event Stream Processing C パブリッシュ/サブスクライブと JSON ライブラリを呼び出す Python スクリプトです。アダプタは、Twitter Gnip の Firehose ストリームからイベントブロックを

作成し、それらをソースウィンドウにパブリッシュします。この Twitter ストリームへのアクセスは、Twitter で承認された関係者に限定されています。アクセスには署名付き契約が必要です。

アダプタは、構成された URL へのキープアライブ接続を通じて Gnip ストリームをタップします。接続は、構成されたユーザー ID とパスワードのクレデンシャルを使用します。それ以降のすべての JSON レスポンスは、*"JSON メッセージングのパブリッシュ/サブスクライブ API サポート" (SAS Event Stream Processing: パブリッシュ/サブスクライブ API)*で説明する JSON 解析ルールを使用するイベントブロックに変換されます。対応するソースウィンドウには、アダプタパラメータ esp-ignoremissingschemafields を指定しない限り、受信したすべての JSON キーに対応するフィールドが含まれている必要があります。アップサートを使用するようにアダプタを構成しないかぎり、イベントは挿入演算コードで作成されます。必要に応じて、esp-matchsubstrings パラメータを使用して JSON フィルタリング関数を有効にすることができます。

ソースウィンドウでキーフィールドを 割り当てます。キーフィールドとして JSON キーを使用することが現実的でない場合は、次のキーフィールドのいずれかまたは両方をソースウィンドウスキーマに追加できます。

```
eventindex*:int64
adapterindex*:string
```

eventindex キーフィールドは、入力 JSON から構築されたすべてのイベントに対して増分されます。adapterindex key フィールドには、アダプタのすべてのインスタンスで 一意の定数 GUID 値が格納されます。これらの 2 つのキーフィールドを組み合わせることで、アダプタの複数のインスタンスがキーを複製せずに同じソースウィンドウにイベントをパブリッシュすることができます。

使用法:

dfesp_twittergnip_publisher -esp-url *url* -streampassword *streampassword* -streamurl *streamurl* -streamuserid *streamuserid* < <-esp-dateformat *dateformat* > > <-esp-ignoremissingschemafields > <-esp-logconfigfile *logconfigfile* > < <-esp-loglevel *loglevel* > > <-esp-matchsubstrings *substrings* > <-esp-publishwithupsert> <-gnipkeepalive *gnipkeepalive*><--maxrecords *maxrecords* >

パラメータ	説明
-esp-dateformat dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-esp-ignoremissingschemafields	スキーマで JSON キーが見つからない場合に無視して続行するように指定します。デフォルトでは、エラーが報告され、キーがないときにプロセスは終了します。
-esp-logconfigfile logconfigfile	ログ構成ファイルの名前を指定します。
-esp-loglevel loglevel	ロギングレベルを指定します。指定できる値は、TRACE、DEBUG、INFO、WARN、ERROR、または FATAL です。デフォルト値は INFO です。
-esp-matchsubstrings substrings	コンマで区切られた fieldname:value のペアのリストを指定します。ここで、fieldname 文字列フィールドに value の部分文字列を含まない入力イベントはドロップされます。比較は大文字と小文字を区別しません。
-esp-publishwithupsert	opcode=Upsert でイベントをパブリッシュします。デフォルトでは、イベントは opcode=Insert でパブリッシュされます。

パラメータ	説明
-esp-url url	パブリッシャの標準 URL を "dfESP://host:port/project/continuousquery/window"の形式で指定します。
-gnipkeepalive gnipkeepalive	Gnip 接続のキープアライブ値を指定します。デフォルトは 30 秒です。
-maxrecords maxrecords	処理するレコードの最大数を設定します。デフォルトでは、最大値はありません。
-streampassword streampassword	Gnip パスワードを指定します。
-streamurl streamurl	Gnip ストリーミング URL を https://stream.gnip.com:443/accounts/account/publishers/twitter/streams/stream/label.json の形式で指定します。
-streamuserid streamuserid	Gnip ユーザー名を指定します。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

Twitter Gnip クエリ登録ユーティリティの使用

$DFESP_HOME/lib に GNIPQueryMgr.py という付随する Python スクリプトがあります。パラメータを指定せずにスクリプトを実行すると、使用状況が表示されます。

このスクリプトを使用すると、Gnip で登録された特定のクエリ文字列(特定の Gnip ストリームとラベルに関連付けられている Gnip ルール)を表示、追加、削除できます。

登録されたクエリを表示すると、groupname パラメータを使用してそれらをフィルタリングしない限り、一致するすべてのクエリが表示されます。このパラメータは、rules 配列エントリの value キーに関連付けられている tag キーに対応します。指定した groupname がルールの tag 値の部分文字列として現れると、ルールが表示されます。

クエリを追加または削除するには、指定したクエリ文字列に加えて、対応する groupname パラメータが必要です。

URL コネクタの使用

URL コネクタを使用すると、URL ベース接続を介して SAS Event Stream Processing にデータを持ち込み、データを変換し、イベントをイベントストリーム処理モデルにパブリッシュすることができます。SAS Event Stream Processing に URL コネクタを使用してパブリッシュできるデータの例を次に示します。

- いくつかの RSS ヘッドラインニュースフィードから引き出された XML データ
- 気象サービスからの REST 要求で JSON のライブ天気データを取得
- HTML ページから取得したニュース記事

1 つの URL 要求で多くのイベントが生成される可能性があります。URL コネクタは、イベントループ技術を使用して、イベントとして SAS Event Stream Processing にデータを解析、変換、パブリッシュします。

URL コネクタを使用するイベントストリーム処理モデルを定義する XML は、外部構成ファイルを指します。この構成ファイルには、取得する URL と、各 URL の情報を SAS Event Stream Processing にパブリッシュできる

データに変換する方法に関する情報が含まれています。構成ファイルの例については、SAS Event Stream Processing インストールディレクトリの examples フォルダにある URL コネクタの例を参照してください。

(Linux) **$DFESP_HOME/examples/xml/url_connector_news/config.xml** または **DFESP_HOME/examples/xml/url_connector_weather/config.xml**

(Windows) **DFESP_HOME\examples\xml\url_connector_news\config.xml** または **DFESP_HOME\examples\xml\url_connector_weather\config.xml**

URL コネクタを使用すると、関連するソースウィンドウに任意の数のパブリッシャを指定できます。各パブリッシャには、イベントストリーム処理のパブリッシング用のデータを準備するための独自の変換ルールがあります。各パブリッシャは、パブリッシャに対して任意の数の要求を定義できます。同じデータ準備ルールを使用してイベントを作成する場合、要求が同じパブリッシャに存在することがあります。たとえば、1 つのパブリッシャが複数のニュースフィードに REST 要求を送信し、1 つのパブリッシャと複数のリクエスト定義を使用して SAS Event Stream Processing にパブリッシュすることができます。

URL コネクタは、イベントをパブリッシュするためにのみ使用できます。

表 71 パブリッシャ URL コネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
configUrl	コネクタ構成の URL を指定します。

表 72 パブリッシャ URL コネクタのオプションパラメータ

パラメータ	説明
interval	要求の送信間隔を指定します。デフォルトは 10 seconds です。
properties	構成で使用できるプロパティを指定します。プロパティは、名前と値のペアのセミコロンで区切られたリストとして入力されます。
maxevents	パブリッシュするイベントの最大数を指定します。

UVC コネクタとアダプタの使用

UVC コネクタの使用

UVC コネクタを使用すると、V4L2 互換カメラで撮影した写真を SAS Event Stream Processing にパブリッシュできます。1 つの UVC コネクタは一度にメモリ内で実行できます。カメラのストリーミング速度は、ストリーミング写真の解像度、画像の形式、カメラの品質、frame_rate アダプタ構成パラメータなど、いくつかの要因によって異なります。

モデルの XML 定義の UVC コネクタパラメータで、画像形式、フレーム率、画像サイズ、およびその他の写真プロパティを指定できます。UVC コネクタが開始すると、コンソールは、カメラがサポートする画像形式、フレーム率、および画像サイズのリストを返します。

UVC コネクタでは、MJPG (Motion-JPEG)と YUYV がサポートされています。format_in または format_out コネクタパラメータで MJPG および YUYV 以外の形式が指定されている場合、またはカメラが指定された形式をサポートしていない場合、コネクタは失敗します。width および height パラメータで、カメラでサポートされていない画像サイズが指定されている場合、カメラでは最も近い互換サイズに写真が拡大/縮小され、コンソールではキャプチャされた画像サイズとパブリッシュされた画像サイズの違いについて警告が表示され、プロセスは続行されます。同様に、frame_rate パラメータで、カメラでサポートされていないフレーム率が指定されている場合、コネクタではカメラのキャプチャ率が最も近いサポート率に設定されます。

注: frame_rate パラメータでは、パブリッシュされたフレーム率が指定されます。これは、キャプチャされたフレーム率とは異なります。*キャプチャフレーム率*は、カメラが写真をキャプチャするフレーム率で、frame_rate パラメータで指定されたパブリッシュフレーム率と異なる場合があります。パブリッシュされる画像は常に最新のキャプチャ画像です。

V4L2 互換カメラは、ブロッキングモードと非ブロッキングモードをサポートしています。V4L2 は、出力バッファを使用してユーザーに写真を配信します。出力バッファがデータで満たされておらず使用可能な場合、ブロッキングモードのコネクタは待機し、非ブロッキングモードのコネクタはデータなしですぐに返します。blocking パラメータでブロックを有効にします。

UVC コネクタは、イベントをパブリッシュするためにのみ使用できます。

表 73 UVC パブリッシャコネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。

表 74 UVC パブリッシャコネクタのオプションパラメータ

パラメータ	説明
frame_rate	カメラがストリームする 1 秒あたりのフレーム数を指定します。 ダブルでなければなりません。デフォルト値は 15 です。
format_in	撮影した写真の画像形式を指定します。 デフォルトは mjpg です。解凍されたイメージ形式である yuyv もサポートされています。
format_out	コネクタがパブリッシュするイメージ形式を指定します。 デフォルトは mjpg です。圧縮されていないイメージ形式の yuyv は、format_in が yuyv の場合にのみサポートされます。
width	写真の幅を指定します。 すべての解像度がサポートされているわけではありません。サポートされている解像度のリストについては、カメラメニューを参照するか、v4l2-ctl コマンドラインユーティリティを使用してください。
height	写真の高さを指定します。 すべての解像度がサポートされているわけではありません。サポートされている解像度のリストについては、カメラメニューを参照するか、v4l2-ctl コマンドラインユーティリティを使用してください。

パラメータ	説明
brightness	写真の明るさを指定します。 サポートされている輝度値のリストについては、カメラメニューを参照するか、v4l2-ctl コマンドラインユーティリティを使用してください。
gain	写真のゲインを指定します。 サポートされているゲイン値の一覧については、カメラメニューを参照するか、v4l2-ctl コマンドラインユーティリティを使用してください。
saturation	写真の彩度を指定します。 サポートされている飽和値のリストについては、カメラメニューを参照するか、v4l2-ctl コマンドラインユーティリティを使用してください。
contrast	写真のコントラストを指定します。 サポートされているコントラスト値のリストについては、カメラメニューを参照するか、v4l2-ctl コマンドラインユーティリティを使用してください。
device	Linux オペレーティングシステムでカメラが使用しているデバイス名を指定します。 デバイス名は通常、 **/dev/videoX** です。
blocking	コネクタがブロッキングモードかどうかを指定します。 デフォルトは false (非ブロッキング)です。
predelay	コネクタを起動する際の遅延時間を秒単位で指定します。 環境によっては、カメラのリセットや初期化に時間がかかることがあります。カメラの始動に失敗し、predelay が *n* になっていると、コネクタは開始するまでに *n* 秒間待機します。コネクタがカメラの検出に失敗した場合、ESP サーバーは致命的なエラーで終了します。
maxevents	パブリッシュするイベントの最大数を指定します。

UVC アダプタの使用

UVC アダプタを使用すると、V4L2 互換カメラで撮影した写真を SAS Event Stream Processing にパブリッシュできます。

パブリッシャ:

dfesp_uvc_adapter -h *url* <-B> <-b *blocksize*> <-C *configfilesection*> <-d *device*> <-E *tokenlocation*> <-e> <-g *gdconfig*> <-H *height*> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *maxevents*> <-Q> <-R> <-r *frame_rate*> <-V> <-W *width*> <-y *logconfigfile*> <-Z *transportconfigfile*>

パラメータ	説明
-B true|false	true の場合、アダプタをブロッキングモードに設定します。デフォルトは“false”です。
-b blocksize	イベントブロックサイズを指定します。デフォルトは 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-D delay	カメラを起動できない場合に再試行するまでに遅延する秒数を指定します。デフォルトは 0 です。
-d devicename	デバイス名を指定します。デフォルトは“/dev/video0”です。
-E tokenlocation	OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。このトークンは、パブリッシュ/サブスクライブサーバーによる認証に必要です。
-e	イベントがトランザクションであることを指定します。
-g gdconfig	保証された配信構成ファイルを指定します。
-H height	フレーム解像度の高さを指定します。デフォルトは 480 です。
-h url	ESP パブリッシュ/サブスクライブの標準 URL を dfESP://<host>:<port>/<project>/<contquery>/<window>の形式で指定します。
-j trace | debug | info | warn | error | fatal | off	ロギングレベルを指定します。デフォルトは“warn”です。
-l native | solace | tervela | rabbitmq | kafka	パブリッシュ/サブスクライブトランスポートを指定します。デフォルトは native です。
-m maxevents	パブリッシュするイベントの最大数を指定します。
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-R	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-r framespersecond	1 秒あたりのフレーム数を指定します。デフォルトは 15 です。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。

パラメータ	説明
-W width	フレーム解像度の幅を指定します。デフォルトは 640 です。
-y logconfigfile	ロギング構成ファイルを指定します。デフォルトでは、何もありません。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です。 -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート構成ファイルは必要ありません。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

WebSocket コネクタの使用

概要

WebSocket コネクタを使用すると、WebSocket ベースの接続を介してデータを読み取り、SAS Event Stream Processing にデータをパブリッシュできます。コネクタは、WebSocket 接続を介して XML および JSON をサポートし、データを解析してストリーミングイベントに変換できる完全機能の API を提供します。

WebSocket コネクタは、イベントループ技術を使用して、データを解析し、変換し、SAS Event Stream Processing にパブリッシュします。

XML および JSON の場合、コネクタは適切な種類のオブジェクトを作成できるようになるまで WebSocket データを読み取ります。その後、関数を使用してオブジェクトを処理し、イベントをパブリッシュします。

WebSocket コネクタは、パブリッシュする目的でのみ使用できます。

表 75 WebSocket コネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
url	WebSocket 接続の URL を指定します。

パラメータ	説明
configUrl	コネクタ構成ファイルの URL を指定します。この構成ファイルには、イベントをパブリッシュするために必要な変換ステップに関する情報が含まれています。 構成ファイルの例については、SAS Event Stream Processing インストールディレクトリの examples フォルダにある WebSocket コネクタの例を参照してください。 (Linux) **$DFESP_HOME/examples/xml/ws_connector_json/config.xml** or **DFESP_HOME/examples/xml/ws_connector_xml/config.xml** (Windows) **DFESP_HOME\examples\xml\ws_connector_json\config.xml** or **DFESP_HOME\examples\xml\ws_connector_xml\config.xml**
contentType	WebSocket 接続で受け取ったコンテンツの種類として XML または JSON を指定します。

表 76 WebSocket コネクタのオプションパラメータ

パラメータ	説明
sslCertificate	安全なサーバーに接続するときに使用する SSL 証明書の場所を指定します。
sslPassphrase	SSL 証明書のパスワードを指定します。
requestHeaders	サーバーに送信する要求ヘッダーのコンマ区切りリストを指定します。このリストは、name:value の形式で名前と値のペアで構成されていなければなりません。
maxevents	パブリッシュするイベントの最大数を指定します。

例: WebSocket コネクタを使用して Teradata Listener からパブリッシュする

WebSocket コネクタを使用して、サブスクライブ専用の SAS Event Stream Processing コネクタを持つソースからパブリッシュします。

WebSocket コネクタを使用して、Teradata Listener Stream サービスから SAS Event Stream Processing にデータを送信できます。使用例には、SAS Event Stream Processing を使用して、リスナデータストリームのパターンをフィルタリング、集計、スコアリング、または検出することが含まれます。変換されたデータは、Teradata、Aster、または Hadoop にロードされる別の Listener ソースに再度挿入できます。次の XML 例は、Teradata Listener Stream サービスに接続されている WebSocket コネクタを宣言するための構文に従います。

```
<connectors>
 <connector class='websocket'>
  <properties>
   <property name='type'>pub</property>
   <property name='url'>wss://listener-streamer-services-poc.labs.teradata.com/v1/streamer/
        444186fb-b652-485f-9dc3-c00c60863476?secret=e52378aa-b7aa-4722-a226-ca18225481f9</property>
   <property name='contentType'>json</property>
   <property name='configUrl'>file://ws.xml</property>
```

```
    <property name='sslCertificate'>/etc/pki/tls/certs/listener.crt</property>
  </properties>
 </connector>
</connectors>
```

url WebSocket コネクタパラメータには、ストリーム識別子とストリーム秘密鍵を含むリスナストリーム API (バージョン 1)の URL を指定します。これらの値を取得するには、Teradata Listener 管理者に連絡してください。

SSLCert WebSocket コネクタパラメータの場合、リスナーストリームサービスに安全に接続するための SSL 証明書を含むファイルへのパスを指定します。リスナーは TLS 1.2 を使用します。

IBM WebSphere MQ コネクタとアダプタの使用

IBM WebSphere MQ コネクタの使用

IBM WebSphere MQ コネクタ(MQ)は、パブリッシュおよびサブスクライブ操作用の IBM WebSphere Message Queue インターフェースをサポートします。サブスクライバは、イベントブロックを受信し、それらを MQ キューにパブリッシュします。パブリッシャは MQ サブスクライバであり、受信したイベントブロックをソースウィンドウにインジェクトします。

IBM WebSphere MQ Client ランタイムライブラリは、コネクタの実行中のインスタンスをホスティングするプラットフォームにインストールする必要があります。ランタイム環境では、それらのライブラリへのパスを定義する必要があります(たとえば、Linux プラットフォームでは **LD_LIBRARY_PATH** を指定します)。

注: 使用する前に、このコネクタへの参照をファイル**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.excluded** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.excluded** (Windows)から削除します。

コネクタは MQ クライアントとして動作します。コネクタの MQ 接続パラメータを指定するには、環境変数 **MQSERVER** を定義する必要があります。この変数は、サーバーのチャネル、トランスポートの種類、およびホスト名を指定します。詳細については、WebSphere のマニュアルを参照してください。

MQ コネクタは、mqqueue および mqtopic パラメータの設定に応じて、MQ キューまたはトピックとの間でメッセージを読み書きできます。さらに、MQ サブスクライバには、イベントを MQ にパブリッシュするために使用されるメッセージ形式を定義するパラメータが必要です。MQ パブリッシャは、MQ サブスクライバによって生成されたすべてのメッセージの 種類を使用できます。

具体的には、メッセージペイロードがバイナリのかわりにテキストである場合、メッセージヘッダーの形式名は MQSTR に等しくする必要があります。他の値を指定すると、パブリッシャはペイロードがバイナリであると仮定します。

あるいは、ignoremqmdformat パラメータを構成して、メッセージ形式パラメータを無視することもできます。この場合、パブリッシャコネクタは、その形式が mqtype パラメータ設定と互換性があるとみなします。

MQ サブスクライバまたはパブリッシャは、MQ キューに対して読み取りまたは書き込みを行うには、mqqueue パラメータを構成します。mqtopic パラメータの構成は必須ではありません。Websphere トピックから読み取る場合、MQ パブリッシャは永続サブスクリプションに関連する 2 つの追加パラメータを必要とします。パブリッシャは、保持サブスクリプションを使用して MQ トピックに常にサブスクライブします。これは、パブリッシャが以前のサブスクリプションを再確立し、パブリッシャが切断されている間に関連トピックにパブリッシュされたメッセージを受信できることを意味します。

これらのパラメータは次のとおりです。

- サブスクリプション名。ユーザが提供し、サブスクリプションを一意に識別します。
- パブリッシャによって入力用に開かれた MQ キュー

MQ サブスクライバによって MQ に書き込まれるメッセージの MQ 永続性設定は、常に MQ キューの保持性設定と同じです。

パブリッシャとサブスクライバの両方が **usecorrelid** パラメータをサポートしているため、各 MQ メッセージの相関 ID を SAS Event Stream Processing イベントの **correlid** フィールドにコピーするか correlid フィールドからコピーします。これは、mqtype パラメータが **csv** または **json** の場合にのみ適用されます。**correlid** フィールドはウィンドウスキーマ内の任意の場所にできますが、ESP_UTF8STR 型でなければなりません。複数のイベントを MQ メッセージに書き込むサブスクライバの場合、相関 ID 値はイベントブロックの最初のイベントからコピーされます。パブリッシャが MQ メッセージから複数のイベントを読み取る場合、メッセージから作成されたすべてのイベントに同じ相関 ID 値が書き込まれます。MQ 相関 ID はバイト文字列(文字列ではない)なので、SAS Event Stream Processing イベントの **correlid** フィールドの値は常に base64 でエンコードされます。

MQ コネクタには、次のパラメータを使用します。

表 77 サブスクライブ MQ コネクタの必須パラメータ

パラメータ	説明
type	サブスクライブを指定します。“sub”にしてください。
mqtype	binary、CSV、JSON、XML、またはサブスクライブされたウィンドウスキーマ内の文字列フィールドの名前を指定します。
snapshot	スナップショットデータを送信するかどうかを指定します。 true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。

表 78 パブリッシャ MQ コネクタの必須パラメータ

パラメータ	説明
type	パブリッシュを指定します。“pub”にしてください。
mqtype	binary、CSV、JSON、XML、opaquestring のいずれかを指定します。opaquestring の場合、ソースウィンドウスキーマは index:int64,message:string とみなされます。

表 79 サブスクライブ MQ コネクタのオプションパラメータ

パラメータ	説明
mqtopic	MQ トピック名を指定します。mqqueue が構成されていない場合は必須です。
mqqueue	MQ キュー名を指定します。mqtopic が構成されていない場合は必須です。
collapse	サブスクライバ出力をパブリッシュ可能にするために UPDATE_BLOCK イベントの変換を有効にします。
queuemanager	MQ キューマネージャーを指定します。

パラメータ	説明
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
rmretdel	ウィンドウ保持ポリシーによって導入されたサブスクライバによって受信されたイベントブロックからすべての削除イベントを削除することを指定します。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config**(Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config**(Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
protofile	Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。この定義は、イベントブロックを protobuf メッセージに変換するために使用されます。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
usecorrelid	イベントの **correlid** フィールドの値を MQ メッセージ相関 ID にコピーします。
csvmsgperevent	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
csvmsgpereventblock	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

表 80 パブリッシャ MQ コネクタのオプションパラメータ

パラメータ	説明
mqtopic	MQ トピック名を指定します。mqqueue が構成されていない場合は必須です。
mqqueue	MQ キュー名を指定します。mqtopic が構成されていない場合は必須です。
mqsubname	MQ サブスクリプション名を指定します。mqqueue が構成されていない場合は必須です。
blocksize	パブリッシュされたイベントブロックに含めるイベントの数を指定します。デフォルト値は 1 です。

パラメータ	説明
transactional	イベントブロックの種類をトランザクションに設定します。デフォルトのイベントブロックの種類は標準です。
dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
queuemanager	MQ キューマネージャーを指定します。
configfilesection	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config**(Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config**(Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
ignorecsvparseerrors	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。
protofile	Google プロトコルバッファメッセージ定義を含む**.proto** ファイルを指定します。この定義は、イベントブロックを protobuf メッセージに変換するために使用されます。このパラメータを指定する場合は、protomsg パラメータも指定する必要があります。
protomsg	protofile パラメータで指定した**.proto** ファイル内の Google プロトコルバッファメッセージの名前を指定します。イベントブロックはこのメッセージに変換されます。
csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は**,**文字です。
noautogenfield	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。
publishwithupsert	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
addcsvopcode	CSV イベントを入力するための演算コードとカンマを先頭に付けます。演算コードは、publishwithupsert が有効になっていない場合は挿入です。
addcsvflags	入力 CSV イベントに挿入するイベントの種類を指定します(カンマ)。有効な値は **normal** と **partialupdate** です。
usecorrelid	すべての SAS Event Stream Processing イベントの MQ メッセージ相関 ID の値を **correlid** フィールドにコピーします。
ignoremqmdformat	Message Descriptor Format パラメータの値を無視し、メッセージ形式が mqtype パラメータ設定と互換性があると仮定します。
maxevents	パブリッシュするイベントの最大数を指定します。

IBM WebSphere MQ アダプタの使用

IBM WebSphere MQ アダプタは、IBM WebSphere Message Queue システムでのパブリッシュおよびサブスクライブ操作をサポートします。このアダプタを使用するには、IBM WebSphere MQ Client ランタイムライブラリをインストールし、環境変数 MQSERVER を定義してアダプタの MQ 接続パラメータを指定する必要があります。

サブスクライバの使用:

dfesp_mq_adapter -h *url* -k sub -t *mqtype* <-a *protofile* > <-C [*configfilesection*]> <-c> <-d *dateformat* > <-E *tokenlocation* > <-f *mqtopic*> <-g *gdconfig* > <-j trace | debug | info | warn | error | fatal | off> <-L> <-l native | solace | tervela | rabbitmq | kafka> <-M> <-m *protomsg* > <-q *mqqueuemanager* > <-U *mqqueue*> <-V> <-y *logconfigfile* > <-Z *transportconfigfile*> <-3 *doubleprecision*>

パブリッシャ:

dfesp_mq_adapter -h *url* -k pub -t *mqtype* < -A> <-a *protofile* > <-b *blocksize* > <-C [*configfilesection*]> <-c> < -D *csvfielddelimiter* > <-d *dateformat* > <-E *tokenlocation* > <-e> <-F *eventtype*> <-f *mqtopic*> <-g *gdconfig* > <-I> <-j trace | debug | info | warn | error | fatal | off> <-l native | solace | tervela | rabbitmq | kafka> <-m *protomsg* > <-n *mqsubname* > <-O> <-o> <-Q> <-q *mqqueuemanager* > <-R> <-U *mqqueue*> <-V> <-Y *maxevents*> <-y *logconfigfile* > <-Z *transportconfigfile*>

パラメータ	説明
-A	入力イベントに、ソースウィンドウによって 自動生成されるキーフィールドがないことを指定します。
-a protofile	Google プロトコルバッファーのサポートに使用されるメッセージを含む**.proto** ファイルを指定します。
-b blocksize	ブロックサイズを指定します。デフォルト値は 1 です。
-C [configfilesection]	構成パラメータを解析するために**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.config** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.config** (Windows)にセクションの名前を指定します。値を[configfilesection]として指定します。
-c	SAS Event Stream Processing イベントの **correlid** 文字列フィールドに MQ 相関 ID をコピーします。
-D csvfielddelimiter	入力 CSV イベントのフィールドデータの文字区切り文字を指定します。デフォルトの区切り文字は,文字です。
-d dateformat	CSV イベントの ESP_DATETIME フィールドと ESP_TIMESTAMP フィールドの形式を指定します。デフォルトの動作では、これらのフィールドは、エポック以来の整数の秒数(ESP_DATETIME)またはマイクロ秒数(ESP_TIMESTAMP)として解釈されます。dateformat パラメータでは、UNIX の strftime 関数でサポートされている任意の時間形式が受け入れられます。
-E tokenlocation	パブリッシュ/サブスクライブサーバーによる認証に必要な OAuth トークンを含むローカルファイルシステム内のファイルの場所を指定します。
-e	イベントがトランザクションであることを指定します。

パラメータ	説明
-F eventtype	入力 CSV イベントに挿入するイベントの種類をカンマで指定します。有効な値は "normal"と "partialupdate"です。
-f mqtopic	MQ トピック名を指定します。mqtopic が構成されていない場合は必須です。
-g gdconfig	保証された配信構成ファイルを指定します。
-h url	dfESP パブリッシュおよびサブスクライブ標準 URL を "dfESP://host:port/project/continuousquery/window”."の形式で指定します。 サブスクライバに次の行を追加します。?snapshot=true | false です。 ?snapshot=true の場合、サブスクライバは、その時点でウィンドウに含まれている挿入イベントのコレクションを受信します。次いで、サブスクライバは、スナップショットの時間から生成されたイベントのストリームを受信します。後続のイベントは、挿入、更新、または削除です。 必要に応じて、サブスクライバに以下を追加します。 ■ ?collapse=true | false ■ ?rmretdel=true | false
-I	入力 CSV イベントのフィールドを解析できない場合、イベントが破棄され、エラーが記録され、パブリッシュが続行されるように指定します。
-j trace | debug | info | warn | error | fatal | off	アダプタのロギングレベルを設定します。これは、C_dfESPpubsubInit()パブリッシュ/サブスクライブ API 呼び出しとエンジンの initialize()呼び出しで設定できるロギングレベルの範囲と同じです。
-k	サブスクライバの使用の sub とパブリッシャの使用のための pub を指定します。
-L	CSV の場合、イベントブロックごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
-l native | solace | tervela | rabbitmq | kafka	トランスポートの種類を指定します。既定の native トランスポートの代わりに solace、tervela、rabbitmq、または kafka トランスポートを指定する場合は、C ++ C_dfESPpubsubSetPubsubLib()API 呼び出しの説明で指定されたクライアント構成ファイルを使用します。
-M	CSV の場合、イベントごとに 1 つのメッセージを送信するように指定します。デフォルトは、トランザクションイベントブロックごとに 1 つのメッセージ、またはイベントごとに 1 つのメッセージです。
-m protomsg	protofile パラメータで指定された**.proto** ファイル内のメッセージ自体を指定します。
-n mqsubname	MQ サブスクリプション名を指定します。mqqueue が構成されていない場合は必須です。
-O	CSV イベントを入力するための演算コードとカンマを先頭に付けます。オペコードは、-R が有効になっていない場合は挿入されます。
-o	Message Descriptor Format パラメータの値を無視し、メッセージ形式が mqtype パラメータ設定と互換性があると仮定します。

パラメータ	説明
-Q	*maxevents* を構成すると、すべてのイベントがソースウィンドウにインジェクトされた後でプロジェクトが休止します。
-q mqqueuemanager	MQ キューマネージャー名を指定します。
-R	Insert の代わりに opcode=Upsert を使用してイベントを作成します。
-t mqtype	**binary**、**CSV**、**XML**、または **JSON** を指定します。パブリッシャにとっては、 **opaquestring** もサポートされています。**opaquestring** の場合、ソースウィンドウスキーマは "index:int64,message:string"とみなされます。サブスクライバの場合、サブスクライブされたウィンドウスキーマ内の文字列フィールドの名前もサポートされます。
-U mqqueue	MQ キュー名を指定します。
-V	致命的なエラーが報告された場合は、アダプタを再起動します。
-Y maxevents	パブリッシュするイベントの最大数を指定します。
-y logconfigfile	ログ構成ファイルを指定します。
-Z transportconfigfile	パブリッシュ/サブスクライブトランスポート構成ファイルを指定します。デフォルト値は、-l パラメータで指定されたトランスポートの種類に依存します。 -l solace の場合、デフォルトは**./solace.cfg** です。 -l tervela の場合、デフォルトは**./client.config** です。 -l rabbitmq の場合、デフォルトは**./rabbitmq.cfg** です -l kafka の場合、デフォルトは**./kafka.cfg** です。 **注:** ネイティブトランスポートにはトランスポート 構成ファイルは必要ありません。
-3 doubleprecision	倍精度の ASCII 表現の小数点以下の桁数を指定します。デフォルト値は 6 です。

パブリッシャアダプタのホットフェイルオーバーの実装については、“Kafka によるパブリッシャアダプタのフェイルオーバー” (*SAS Event Stream Processing: 高度なトピック*)を参照してください。

カスタムコネクタの作成と統合

カスタムコネクタの作成

独自のコネクタを作成する場合、コネクタクラスは基本クラス dfESPconnector から継承する必要があります。

コネクタ構成は、すべてのキーと 値がテキスト文字列であるキーまたは値のペアのセットで維持されます。コネクタは、getParameter()を呼び出してキー文字列を渡すことにより、いつでも構成項目の値を取得できます。無効な要求は、空の文字列を返します。

コネクタは、ウィンドウによって生成されたイベントを受け取るサブスクライバ、またはイベントをウィンドウにインジェクトするパブリッシャを実装できます。ただし、コネクタの 1 つのインスタンスは同時にパブリッシュおよびサブスクライブできません。

サブスクライバコネクタは、エンジンが所有するスレッドで呼び出されるコネクタクラスで定義されたコールバックメソッドを使用してイベントを受信します。パブリッシャコネクタは、通常、ソースからイベントを読み取る専用のスレッドを作成します。次に、これらのイベントをソースウィンドウにインジェクトし、その後のコネクタの呼び出しのためにメインコネクタスレッドを残します。

コネクタは、以下の静的データ構造を定義する必要があります。

静的データ構造	説明
dfESPconnectorInfo	コネクタ名、パブリッシュ/サブスクライブの種類、初期化関数ポインター、および構成データのポインタを指定します。
subRequiredConfig	サブスクライバに必要な構成パラメータをリストした dfESPconnectorParmInfo_t エントリの配列を指定します。
sizeofSubRequiredConfig	subRequiredConfig 内のエントリ数を指定します。
pubRequiredConfig	パブリッシャに必要な構成パラメータをリストした dfESPconnectorParmInfo_t エントリの配列を指定します。
sizeofPubRequiredConfig	pubRequiredConfig のエントリ数を指定します。
subOptionalConfig	サブスクライバのオプションの構成パラメータをリストする dfESPconnectorParmInfo_t エントリの配列を指定します。
sizeofSubOptionalConfig	subOptionalConfig 内のエントリ数を指定します。
pubOptionalConfig	パブリッシャのオプションの構成パラメータをリストする dfESPconnectorParmInfo_t エントリの配列を指定します。
sizeofPubOptionalConfig	pubOptionalConfig のエントリ数を指定します。

コネクタは、以下の静的メソッドを定義する必要があります。

静的メソッド	説明
dfESPconnector *initialize(dfESPengine *engine, dfESPpsLib_t psLib)	コネクタのインスタンスを返します。
dfESPconnectorInfo *getConnectorInfo()	dfESPconnectorInfo 構造体を返します。

コネクタのインスタンスを作成する前に、これらの静的メソッドを呼び出すことができます。

コネクタは、以下の仮想メソッドを定義する必要があります。

仮想メソッド	説明
start()	コネクタを起動します。開始する前にコネクタの構成を検証するために、基本クラスのメソッド checkConfig()を呼び出さなければなりません。また、基本クラスのメソッド start()を呼び出す必要があります。成功すると、variable _started = true を設定する必要があります。
stop()	コネクタを停止します。stop()を呼び出さなければなりません。コネクタは、コネクタを再起動するために start()を呼び出すことができる状態にしておく必要があります。
callbackFunction()	エンジンが呼び出すメソッドを指定して、接続先のウィンドウによって生成されたイベントブロックを渡します。
errorCallbackFunction()	エンジンによって検出されたエラーを報告するために、エンジンによって呼び出されるメソッドを指定します。ゼロ以外の場合は、ユーザーコールバック関数_errorCallback を呼び出す必要があります。
setupCallbackFunction()	ソースウィンドウスキーマを必要とするコネクタを設定するために、エンジンによって呼び出されるメソッドを指定します。

コネクタは、コネクタのオーケストレーターが使用するために実行状態を設定する必要があります。これは、dfESPconnector::setState()メソッドを呼び出すことで行います。関連する 2 つの状態は **state_RUNNING** と **state_FINISHED** です。すべてのコネクタは、起動時に **state_RUNNING** を設定する必要があります。データの転送を実際に完了したコネクタだけが **state_FINISHED** を設定する必要があります。通常、このように状態を設定することは、有限数のイベントブロックをパブリッシュするパブリッシャコネクタにのみ関係します。

最後に、派生コネクタは、アプリケーションから呼び出すことができるユーザー定義メソッドを最大 10 個実装できます。コネクタは実行時にロードされるプラグインなので、ユーザーアプリケーションはクラスメソッドを直接呼び出すことはできません。コネクタには接続されていません。

基本コネクタクラスは、仮想メソッド userFunction_01～userFunction_10 を定義し、派生コネクタは必要に応じてこれらのメソッドを実装します。たとえば、

```
void * myConnector::userFunction_01(void *myData) {
```

アプリケーションは、次のようにメソッドを呼び出します。

```
myRC = myConnector->userFunction_01((void *)myData);
```

カスタムコネクタの統合

すべてのコネクタは、グローバルコネクタマネージャによって 管理されます。SAS Event Stream Processing に同梱されているデフォルトのコネクタは、製品の初期化時にコネクタマネージャによって 自動的にロードされます。**/opt/sas/viya/config/etc/SASEventStreamProcessingEngine/default/connectors.excluded** (Linux)または**%ProgramData%\SAS\Viya\SASEventStreamProcessingEngine\default\connectors.excluded** (Windows)にリストされているものを除き、ライブラリとしてビルドされて**$DFESP_HOME/lib/plugins** に配置されたカスタムコネクタも初期化時にロードされます。

初期化後、コネクタは、アプリケーションで定義されたイベントストリームプロセッサウィンドウで使用できるようになります。任意のコネクタと同様に、getConnector()ウィンドウを呼び出してそのユーザー定義メソッドを渡すことで、そのインスタンスを取得できます。プロジェクトを開始する前に、setParameter()を使用してコネクターを構成することができます。

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © 2018, SAS Institute Inc., Cary, NC, USA. All Rights Reserved. November 2018 5.2-P1:espca

sas
THE POWER TO KNOW®