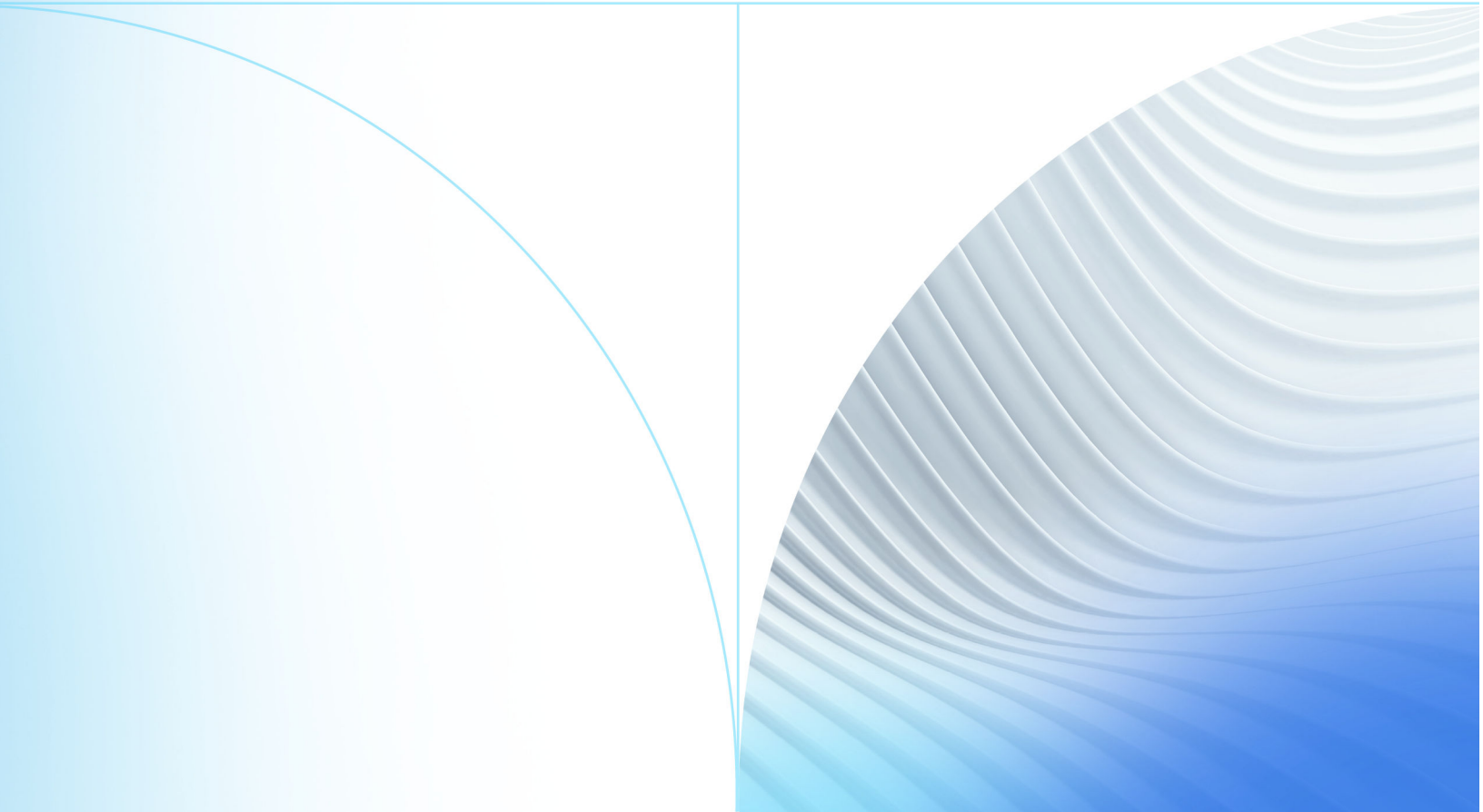
jack henry™

Operational Enhancement Solutions
Enterprise Event System

Release 2023

EES Direct Connect Event Delivery

07/10/2023



Enterprise Event System (EES) Direct Connect Event Delivery

The Direct Connect delivery type is a subscription type that allows the subscriber to connect directly to the Kafka® subscription topic and retrieve events at will.

A Kafka® client connector is used to retrieve these events. The security model on the Kafka® subscription topics requires a certificate and access control lists (ACLs).

Security

When configuring the direct connect subscription, along with settings applicable to the other types of subscriptions, you must provide the certificate common name that is used to connect to the Kafka® topic.

The subscriber uses the referenced certificate with a valid consumer group ID and topic name to retrieve events from the topic. This process becomes the principal of the Kafka® ACL. You can also designate how many partitions to use. This number cannot be reduced.

To connect, pass the path to the certificate in the SslKeystoreLocation of the consumer configuration element.

The user interface generates the subscription topic name and provides it to the subscriber. As a subscriber, you must implement Kafka® IConsumer.

There are references to the Confluent IO website concerning authorization and consumer clients at the end of this document.

Consumer Group ID

For subscribers to retrieve events from the subscription topic, they specify the subscription ID as the consumer group ID.

This information along with the specified certificate, allows for a specific set of consumer group IDs to access a single topic. All the consumer group IDs must use the same prefix, which matches the subscription ID of the subscription.

For example, if the subscription ID is "malwork_financial", a subscriber can specify it as their consumer group ID and access the subscription's topic. When more than one subscriber wants to read from the same topic, variations of the consumer group ID are necessary, but all must use "malwork_financial" as the prefix.

These examples are all valid variations of "malwork_financial":

- malwork_financial1
- malwork_financial_TEST
- malwork_financial_abc

These examples are invalid variations since they do not use "malwork_financial" as their prefix:

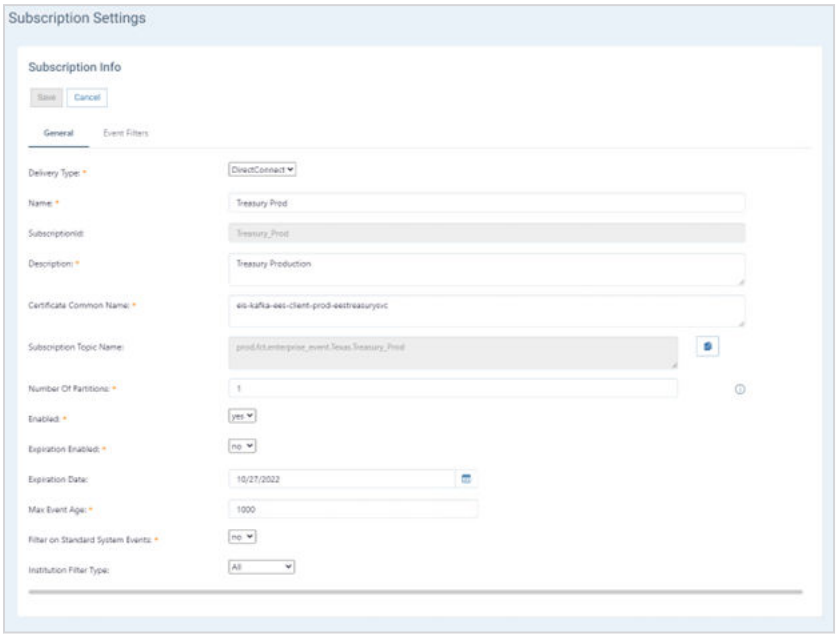
- 1malwork_financial
- malwork_TEST_financial
- abc_malwork_financial

User Interface for Direct Connect

The EI&S Installation team sets up the subscription configuration so it is not something the subscriber is required to set up.

This information has been included in this document to show subscribers the two fields that are used for connecting to the subscription topic directly: **Certificate Common Name** and **Subscription Topic Name**.

This image shows the Direct Connect subscription settings and configuration.



Subscription Topic Schema

The following information is included in the subscription topic schema.

EESEventRecord

EventUniqueId (string)

BusCorrelId (string)

Custom (string)

EESIdNumHistInfo (string)

EventAppId (string)

EventNum (int)

EventProcId (string)

EventProd (string)

EventThreadId (string)

EventTimeDt (DateTimeOffset)

InstId (string)

ProdEnv (string)

SystemId (string)

WorkflowCorrelId (string)

EventUsrId (string)

EventWsId (string)

ReceivedDateTime (DateTimeOffset)

IsTest boolean

AddedByUser (string)

jXLogTrackingId (string)

EESEventDatas (List<EESEventData>)

EventSource (string)

EESEventData

NAME (string)

VAL (string)

PREVVAL (string)

> **NOTE**
> Records are returned from Kafka® in Avro format as generic records. There is a reference to the Avro generic records specification at the end of this document.

The following example code shows how Avro generic records are mapped to `EESEventAdd` records.

```
public static AggregationEESHistory CreateFromGenericRecord(GenericRecord
val)
{
var target = new AggregationEESHistory();
            for (var x = 0; x <= 27; x++)
            {
                target.Put(x, val.GetValue(x));
}

            return target;
}

public void Put(int fieldPos, object fieldValue)
{
switch (fieldPos)
            {
                case 0:  this.EESEventRecordEventUniqueId =
(System.String)fieldValue; break;
                    case 1:  this.EESEventRecordEventNum = (int)fieldValue;
break;
                    case 2:  this.EESEventRecordEventProd =
(System.String)fieldValue; break;
                    case 3:  this.EESEventRecordEventTimeDt =
DateTimeOffset.FromUnixTimeMilliseconds((long)fieldValue); break;
                    case 4:  this.EESEventRecordProdEnv =
(System.String)fieldValue; break;
                    case 5:  this.EESEventSubscriptionDelivered =
(bool)fieldValue; break;
                     case 6:  this.EESEventSubscriptionDeliveredDateTime =
DateTimeOffset.FromUnixTimeMilliseconds((long)fieldValue); break;
                     case 7:  this.EESEventSubscriptionReceivedDateTime =
DateTimeOffset.FromUnixTimeMilliseconds((long)fieldValue); break;
                     case 8:  this.InstallationId =
(System.String)fieldValue; break;
                     case 9:  this.InstitutionAba =
(System.String)fieldValue; break;
                     case 10: this.InstitutionCity =
(System.String)fieldValue; break;
                     case 11: this.InstitutionEnvironment =
(System.String)fieldValue; break;
                     case 12: this.InstitutionName =
(System.String)fieldValue; break;
                     case 13: this.InstitutionState =
(System.String)fieldValue; break;
                     case 14: this.JobId = (System.String)fieldValue; break;
                     case 15: this.RequestDate =
DateTimeOffset.FromUnixTimeMilliseconds((long)fieldValue); break;
                     case 16: this.SubscriberDescription =
```

```
(System.String)fieldValue; break;
                    case 17: this.SubscriberId =
(System.String)fieldValue; break;
                    case 18: this.SubscriberName =
(System.String)fieldValue; break;
                    case 19: this.SubscriberValidatedConsumerName =
(System.String)fieldValue; break;
                    case 20: this.SubscriberValidatedConsumerProduct =
(System.String)fieldValue; break;
                    case 21: this.SubscriptionDeliveryType =
(System.String)fieldValue; break;
                    case 22: this.SubscriptionDescription =
(System.String)fieldValue; break;
                    case 23: this.SubscriptionEnabled = (bool)fieldValue;
break;
                    case 24: this.SubscriptionExpirationDateTime =
DateTimeOffset.FromUnixTimeMilliseconds((long)fieldValue); break;
                    case 25: this.SubscriptionId =
(System.String)fieldValue; break;
                    case 26: this.SubscriptionMaxEventAge =
(int)fieldValue; break;
                    case 27: this.SubscriptionName =
(System.String)fieldValue; break;
                    default: throw new AvroRuntimeException("Bad index " +
fieldPos + " in Put()");
};
}
```

References

The following references were used to create this document.

https://docs.confluent.io/platform/current/kafka/authorization.html.

https://docs.confluent.io/platform/current/clients/confluent-kafka-dotnet/_site/api/Confluent.Kafka.IConsumer-2.html

https://avro.apache.org/docs/1.11.1/api/csharp/html/classAvro_1_1Generic_1_1GenericRecord.html

How To Work With Avro Data Type In .NET Environment (c-sharpcorner.com)