which python package do i need to get JsonSchemaDraftIdentifier for azure event hub schema registry? please show me with a python example also for both producer and consumer.

To work with Azure Event Hub Schema Registry and utilize JsonSchemaDraftIdentifier, you need the azure-schemaregistry and azure-eventhub Python packages. Below is an example demonstrating how to produce and consume events using a JSON schema from the Azure Schema Registry.

### **Install the Required Packages**

bash

CopyEdit

pip install azure-schemaregistry azure-eventhub

### **Code Example**

#### **Producer: Sending Messages with JSON Schema**

python

CopyEdit

from azure.eventhub import EventHubProducerClient, EventData  
from azure.schemaregistry import SchemaRegistryClient  
from azure.schemaregistry.serializer.jsonserializer import JsonObjectSerializer  
  
# Configuration  
EVENTHUB\_CONNECTION\_STRING = "<your-event-hub-connection-string>"  
EVENTHUB\_NAME = "<your-event-hub-name>"  
SCHEMA\_REGISTRY\_FQNS = "<your-schema-registry-fully-qualified-namespace>"  
SCHEMA\_GROUP = "<your-schema-group>"  
SCHEMA\_NAME = "<your-schema-name>"  
  
# Initialize Schema Registry client  
schema\_registry\_client = SchemaRegistryClient(  
 fully\_qualified\_namespace=SCHEMA\_REGISTRY\_FQNS,  
 credential=None, # Use DefaultAzureCredential for managed identity or a specific credential  
)  
  
# Get schema from Schema Registry  
schema = schema\_registry\_client.get\_schema(group\_name=SCHEMA\_GROUP, name=SCHEMA\_NAME)  
schema\_content = schema.definition  
  
# Initialize JSON serializer  
serializer = JsonObjectSerializer(schema=schema\_content)  
  
# Initialize Event Hub producer  
producer = EventHubProducerClient.from\_connection\_string(  
 conn\_str=EVENTHUB\_CONNECTION\_STRING,  
 eventhub\_name=EVENTHUB\_NAME,  
)  
  
# Data to send  
payload = {"name": "Alice", "age": 30, "city": "Seattle"}  
  
# Serialize data  
event\_data = EventData(serializer.serialize(payload))  
event\_data.content\_type = "application/json"  
  
# Send event  
with producer:  
 with producer.create\_batch() as batch:  
 batch.add(event\_data)  
 producer.send\_batch(batch)  
  
print("Message sent successfully!")

#### **Consumer: Receiving Messages and Deserializing**

python

CopyEdit

from azure.eventhub import EventHubConsumerClient  
from azure.schemaregistry.serializer.jsonserializer import JsonObjectSerializer  
  
# Configuration  
CONSUMER\_GROUP = "<your-consumer-group>"  
EVENTHUB\_CONNECTION\_STRING = "<your-event-hub-connection-string>"  
EVENTHUB\_NAME = "<your-event-hub-name>"  
SCHEMA\_CONTENT = "<your-schema-content>" # Fetch the schema content from the registry  
  
# Initialize JSON serializer  
serializer = JsonObjectSerializer(schema=SCHEMA\_CONTENT)  
  
# Callback to process events  
def on\_event(partition\_context, event):  
 print(f"Received event from partition: {partition\_context.partition\_id}")  
 try:  
 data = serializer.deserialize(event.body\_as\_str())  
 print(f"Deserialized data: {data}")  
 except Exception as e:  
 print(f"Failed to deserialize event: {e}")  
  
# Initialize Event Hub consumer  
consumer = EventHubConsumerClient.from\_connection\_string(  
 conn\_str=EVENTHUB\_CONNECTION\_STRING,  
 consumer\_group=CONSUMER\_GROUP,  
 eventhub\_name=EVENTHUB\_NAME,  
)  
  
# Receive events  
with consumer:  
 consumer.receive(on\_event=on\_event, starting\_position="-1") # Start from the beginning of the stream

### **Key Points**

1. Replace placeholders (e.g., <your-event-hub-connection-string>) with your actual configuration values.
2. Ensure the schema in the Schema Registry matches the structure of the data being serialized and deserialized.
3. Use the DefaultAzureCredential for authentication in production environments if applicable.

Let me know if you need further assistance!

how to use the JsonSchemaEncoder and JsonSchemaDraftIdentifier for the above example to send the message to azure event hub? pls show a python example.

To use the JsonSchemaEncoder and JsonSchemaDraftIdentifier for sending messages to Azure Event Hub with a schema from the Azure Schema Registry, you'll need to use the azure-schemaregistry and azure-eventhub Python packages.

Here's how you can integrate these classes into your producer code:

### **Install Required Packages**

bash

CopyEdit

pip install azure-schemaregistry azure-eventhub azure-identity

### **Code Example with JsonSchemaEncoder and JsonSchemaDraftIdentifier**

#### **Producer: Sending Messages with JSON Schema**

python

CopyEdit

from azure.eventhub import EventHubProducerClient, EventData  
from azure.schemaregistry import SchemaRegistryClient  
from azure.schemaregistry.encoder.avroencoder import JsonSchemaEncoder  
from azure.identity import DefaultAzureCredential  
from azure.schemaregistry.encoder.avroencoder.\_draft import JsonSchemaDraftIdentifier  
  
# Configuration  
EVENTHUB\_CONNECTION\_STRING = "<your-event-hub-connection-string>"  
EVENTHUB\_NAME = "<your-event-hub-name>"  
SCHEMA\_REGISTRY\_FQNS = "<your-schema-registry-fully-qualified-namespace>"  
SCHEMA\_GROUP = "<your-schema-group>"  
SCHEMA\_NAME = "<your-schema-name>"  
  
# Initialize Schema Registry client  
schema\_registry\_client = SchemaRegistryClient(  
 fully\_qualified\_namespace=SCHEMA\_REGISTRY\_FQNS,  
 credential=DefaultAzureCredential(),  
)  
  
# Initialize JSON Schema Encoder  
json\_schema\_encoder = JsonSchemaEncoder(  
 client=schema\_registry\_client,  
 group\_name=SCHEMA\_GROUP,  
)  
  
# Initialize Event Hub producer  
producer = EventHubProducerClient.from\_connection\_string(  
 conn\_str=EVENTHUB\_CONNECTION\_STRING,  
 eventhub\_name=EVENTHUB\_NAME,  
)  
  
# Data to send  
payload = {"name": "Alice", "age": 30, "city": "Seattle"}  
  
# Specify the schema version or draft identifier  
draft\_identifier = JsonSchemaDraftIdentifier(schema\_name=SCHEMA\_NAME)  
  
# Serialize data  
encoded\_message = json\_schema\_encoder.encode(payload, schema\_id=draft\_identifier)  
  
# Send the event  
event\_data = EventData(body=encoded\_message)  
event\_data.content\_type = json\_schema\_encoder.get\_content\_type(draft\_identifier)  
  
with producer:  
 with producer.create\_batch() as batch:  
 batch.add(event\_data)  
 producer.send\_batch(batch)  
  
print("Message sent successfully!")

### **Explanation**

1. **Schema Registry Client**:
   1. Connects to your Azure Schema Registry to fetch the schema.
   2. Uses DefaultAzureCredential for authentication (suitable for Azure-hosted environments like Azure VMs, App Services, etc.).
2. **JsonSchemaEncoder**:
   1. Handles serialization of your data using the schema fetched from the registry.
   2. The JsonSchemaDraftIdentifier specifies the schema version or draft identifier to use.
3. **Event Hub Producer**:
   1. Sends the serialized message to Azure Event Hub.
4. **Content Type**:
   1. The content\_type is set using the encoder to ensure the consumer can correctly interpret the data.

### **Consumer Code**

For the consumer, you can deserialize the received messages using the same schema and the JsonSchemaEncoder. Here's a snippet:

python

CopyEdit

from azure.eventhub import EventHubConsumerClient  
from azure.schemaregistry.encoder.avroencoder import JsonSchemaEncoder  
from azure.identity import DefaultAzureCredential  
  
# Configuration  
CONSUMER\_GROUP = "<your-consumer-group>"  
EVENTHUB\_CONNECTION\_STRING = "<your-event-hub-connection-string>"  
EVENTHUB\_NAME = "<your-event-hub-name>"  
  
# Initialize Schema Registry client  
schema\_registry\_client = SchemaRegistryClient(  
 fully\_qualified\_namespace="<your-schema-registry-fully-qualified-namespace>",  
 credential=DefaultAzureCredential(),  
)  
  
# Initialize JSON Schema Encoder  
json\_schema\_encoder = JsonSchemaEncoder(client=schema\_registry\_client)  
  
# Callback to process events  
def on\_event(partition\_context, event):  
 print(f"Received event from partition: {partition\_context.partition\_id}")  
 try:  
 decoded\_data = json\_schema\_encoder.decode(event)  
 print(f"Decoded data: {decoded\_data}")  
 except Exception as e:  
 print(f"Failed to decode event: {e}")  
  
# Initialize Event Hub consumer  
consumer = EventHubConsumerClient.from\_connection\_string(  
 conn\_str=EVENTHUB\_CONNECTION\_STRING,  
 consumer\_group=CONSUMER\_GROUP,  
 eventhub\_name=EVENTHUB\_NAME,  
)  
  
# Receive events  
with consumer:  
 consumer.receive(on\_event=on\_event, starting\_position="-1") # Start from the beginning of the stream

### **Key Notes**

1. **Schema Management**:
   1. Ensure your schema is registered in Azure Schema Registry.
   2. Use the correct schema group and schema name.
2. **Authentication**:
   1. Use DefaultAzureCredential for seamless integration with Azure-managed identities.
   2. Alternatively, provide explicit credentials (e.g., ClientSecretCredential).
3. **Error Handling**:
   1. Implement robust error handling for schema fetch, serialization, and deserialization.

Let me know if you need more details!

how to use the JSON schema instead of using Avro schema for the above example?

To use JSON Schema instead of Avro Schema in the above example with Azure Event Hub and Schema Registry, you can use the JsonSchemaEncoder specifically designed for JSON serialization. Here's how you can adjust the producer and consumer examples:

### **Producer: Sending Messages with JSON Schema**

python

CopyEdit

from azure.eventhub import EventHubProducerClient, EventData  
from azure.schemaregistry import SchemaRegistryClient  
from azure.schemaregistry.encoder.jsonencoder import JsonSchemaEncoder  
from azure.identity import DefaultAzureCredential  
  
# Configuration  
EVENTHUB\_CONNECTION\_STRING = "<your-event-hub-connection-string>"  
EVENTHUB\_NAME = "<your-event-hub-name>"  
SCHEMA\_REGISTRY\_FQNS = "<your-schema-registry-fully-qualified-namespace>"  
SCHEMA\_GROUP = "<your-schema-group>"  
SCHEMA\_NAME = "<your-schema-name>"  
  
# Initialize Schema Registry client  
schema\_registry\_client = SchemaRegistryClient(  
 fully\_qualified\_namespace=SCHEMA\_REGISTRY\_FQNS,  
 credential=DefaultAzureCredential(),  
)  
  
# Initialize JSON Schema Encoder  
json\_schema\_encoder = JsonSchemaEncoder(client=schema\_registry\_client, group\_name=SCHEMA\_GROUP)  
  
# Initialize Event Hub producer  
producer = EventHubProducerClient.from\_connection\_string(  
 conn\_str=EVENTHUB\_CONNECTION\_STRING,  
 eventhub\_name=EVENTHUB\_NAME,  
)  
  
# Data to send  
payload = {"name": "Alice", "age": 30, "city": "Seattle"}  
  
# Serialize data using JSON Schema  
encoded\_message = json\_schema\_encoder.encode(payload, schema\_name=SCHEMA\_NAME)  
  
# Prepare EventData  
event\_data = EventData(body=encoded\_message)  
event\_data.content\_type = json\_schema\_encoder.get\_content\_type(schema\_name=SCHEMA\_NAME)  
  
# Send the event  
with producer:  
 with producer.create\_batch() as batch:  
 batch.add(event\_data)  
 producer.send\_batch(batch)  
  
print("Message sent successfully!")

### **Consumer: Receiving and Deserializing JSON Schema Messages**

python

CopyEdit

from azure.eventhub import EventHubConsumerClient  
from azure.schemaregistry.encoder.jsonencoder import JsonSchemaEncoder  
from azure.identity import DefaultAzureCredential  
  
# Configuration  
CONSUMER\_GROUP = "<your-consumer-group>"  
EVENTHUB\_CONNECTION\_STRING = "<your-event-hub-connection-string>"  
EVENTHUB\_NAME = "<your-event-hub-name>"  
SCHEMA\_REGISTRY\_FQNS = "<your-schema-registry-fully-qualified-namespace>"  
  
# Initialize Schema Registry client  
schema\_registry\_client = SchemaRegistryClient(  
 fully\_qualified\_namespace=SCHEMA\_REGISTRY\_FQNS,  
 credential=DefaultAzureCredential(),  
)  
  
# Initialize JSON Schema Encoder  
json\_schema\_encoder = JsonSchemaEncoder(client=schema\_registry\_client)  
  
# Callback to process events  
def on\_event(partition\_context, event):  
 print(f"Received event from partition: {partition\_context.partition\_id}")  
 try:  
 decoded\_data = json\_schema\_encoder.decode(event)  
 print(f"Decoded data: {decoded\_data}")  
 except Exception as e:  
 print(f"Failed to decode event: {e}")  
  
# Initialize Event Hub consumer  
consumer = EventHubConsumerClient.from\_connection\_string(  
 conn\_str=EVENTHUB\_CONNECTION\_STRING,  
 consumer\_group=CONSUMER\_GROUP,  
 eventhub\_name=EVENTHUB\_NAME,  
)  
  
# Receive events  
with consumer:  
 consumer.receive(on\_event=on\_event, starting\_position="-1") # Start from the beginning of the stream

### **Key Differences for JSON Schema**

1. **Encoder**:
   1. Use JsonSchemaEncoder from the azure-schemaregistry.encoder.jsonencoder module instead of Avro-specific encoders.
   2. JSON Schema is used for serialization and deserialization.
2. **Schema Name**:
   1. Pass the schema name (schema\_name=SCHEMA\_NAME) to specify the schema in the registry.
3. **Content Type**:
   1. The content\_type is set automatically by the encoder to ensure the consumer knows how to deserialize the data.
4. **Payload**:
   1. The payload should conform to the JSON schema registered in the Schema Registry.

### **Schema Registration**

Ensure that the JSON schema is registered in Azure Schema Registry under the specified schema group and name. You can use the Azure portal, Azure CLI, or SDKs to register the schema.

Let me know if you need help with schema registration or any additional clarification!