



User Guide | PUBLIC
2022-02-25

SAP Event Mesh

Content

1	What Is SAP Event Mesh?	4
2	What's New for Event Mesh	7
2.1	2020 What's New for Enterprise Messaging (Archive)	8
2.2	2019 Enterprise Messaging (Archive)	10
2.3	2018 What's New for Enterprise Messaging (Archive)	12
3	Scope and Limitations	14
4	SAP Event Mesh Plans	16
5	Concepts	18
5.1	Messaging Protocols and Libraries	21
	REST APIs for Messaging	23
	REST APIs for Events	26
5.2	Syntax for Naming Queues, Topics, and Topic Patterns	30
5.3	Syntax for Service Descriptor	32
5.4	Event Mesh User Interface	38
6	Initial Setup	40
6.1	Get Started with the Event Mesh Default Plan	40
	Setting Up the SAP Event Mesh Default Plan	40
	Creating an Event Mesh Service Instance	42
	Creating an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface	44
	Binding an Application to an Event Mesh Service Instance	46
6.2	Get Started with the Event Mesh User Interface	49
	Setting Up an Event Mesh Standard Application Plan	50
	Assigning Roles to Users	51
	Subscribing to the Event Mesh User Interface	52
7	Development	54
7.1	REST APIs for Development	54
	Use REST APIs to Send and Receive Messages	55
	Use REST APIs to Manage Queues and Queue Subscriptions	56
	Use REST APIs to Send and Receive Events	57
8	Using Event Mesh Default Plan	59
8.1	View Rules	59

8.2	Manage Queues.	60
8.3	Manage Webhooks.	62
8.4	View Event Catalog for a Message Client.	64
8.5	View Service Descriptor.	64
8.6	View Event Catalog for a Subaccount.	65
8.7	Monitor Resources.	66
8.8	Test Publishing or Consuming Messages.	67
8.9	Edit Resource Units for a Subaccount.	68
9	Working with Event-Driven Integrations.	70
9.1	SAP-Managed Subscriptions.	70
9.2	SAP-Managed Subscription Monitoring.	71
9.3	Enabling SAP Event Subscriptions.	72
9.4	Registering Event Mesh with SAP Cloud ALM.	74
10	Security.	76
10.1	Technical System Landscape.	76
10.2	User Roles for Event Mesh.	77
10.3	Authentication and Authorization.	79
10.4	Transport Encryption.	80
10.5	Data Protection and Privacy.	80
10.6	Auditing and Logging Information.	82
11	Monitoring and Troubleshooting.	84
12	Glossary.	85
13	Event Mesh Lite Service Plan (Deprecated).	86
13.1	Concepts (Deprecated Lite Service Plan).	86
	Messaging Protocols and Libraries.	87
13.2	Initial Setup (Deprecated Lite Service Plan).	89
	Set up a Subaccount.	90
	Create an Event Mesh Service Instance.	91
	Create an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface	93
	Bind an Application to an Event Mesh Service Instance.	94
	Subscribe to the Event Mesh Business Application.	96
	Create User Groups, Role Collection and Assign Role Collection.	97
	Integrating the Service with SAP S/4HANA (Deprecated Lite Service Plan).	97
13.3	Using Event Mesh (Deprecated Lite Service Plan).	99
	Messaging Administration.	99
	Events Administration.	106

1 What Is SAP Event Mesh?

Learn more about the Event Mesh service for SAP BTP.

SAP Event Mesh is a fully managed cloud service that allows applications to communicate through asynchronous events. Experience greater agility and scalability when you create responsive applications that work independently and participate in event-driven business processes across your business ecosystem.

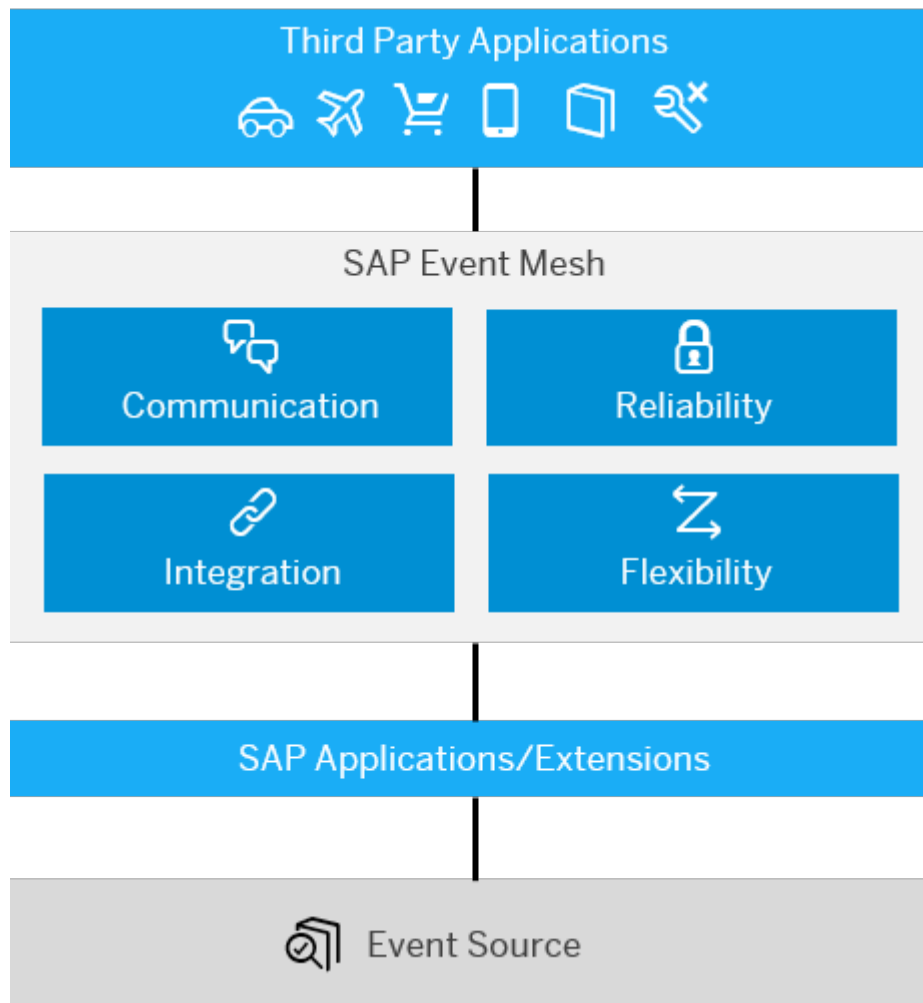
Environment

This service runs in the Cloud Foundry environment.

Features

Publish business events	Publish business events from SAP and non-SAP sources across hybrid landscapes from the digital core to extension applications through event-driven architecture.
Consume business events	Consume business events from SAP and non-SAP sources throughout SAP's event-driven ecosystem including SAP Extension Suite, SAP Integration Suite, and selected inbound enabled SAP backends.
Connect seamlessly	Achieve reliable data transmission for extension and integration scenarios through decoupled communication.

Overview



Tools

Tools	Description
SAP BTP cockpit	The central point for managing all activities associated with your subaccount.
Event Mesh Business Application	A web user interface that allows you to manage messaging clients and event catalogs.

Scope and Limitations

Ensure that you understand the scope and limitations associated with the service before you use it for your business scenarios. See [Scope and Limitations \[page 14\]](#).

Information

The lite service plan for SAP Event Mesh is being discontinued. It's recommended that you discontinue using service instances based on the lite service plan and create new service instances with the default service plan. By migrating to the default service plan, you benefit from an optimized messaging infrastructure, which fully supports scalable and versatile API or event-based business scenarios. See [Initial Setup](#).

Related Information

[Feature Scope Description](#)


[About Services](#)

[SAP Discovery Center](#) 

2 What's New for Event Mesh

2021

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Event Mesh	Integration Suite	Cloud Foundry	xs-security object for Service Instance and Service Binding	<p>The xs-security object is used to configure XSUAA-related properties, for example, the supported credential-types for an instance. See:</p> <ul style="list-style-type: none"> Syntax for Service Descriptor Create an Event Mesh Service Instance Create an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface Bind an Application to an Event Mesh Service Instance <p>Action: If you have XSUAA managed certificates, you can add the xs-security object to your service descriptor or update an existing service descriptor with the object. An example of the code snippet to be added is provided here.</p>	Recommended	Changed	2021-10-14
Event Mesh	Integration Suite	Cloud Foundry	Event-Driven Integrations for SAP applications	<p>Two new Event Mesh features are automatically available when the registered SAP applications with event catalogs in your landscape use event-driven integrations. Those SAP applications can use Event Mesh to distribute events between publishing SAP applications and subscribing SAP applications. Event-driven integrations are configured on the SAP side and then you control the flow of events in your landscape.</p> <ul style="list-style-type: none"> Initiate the flow of events to subscribing SAP applications by enabling preconfigured event subscriptions using the Event Mesh UI. For more information, see Enabling SAP Event Subscriptions [page 72]. Register Event Mesh with SAP Cloud ALM to see errors that occur when events are distributed between SAP applications. For more information, see Registering Event Mesh with SAP Cloud ALM [page 74]. 	Info only	New	2021-10-06

Technical Component	Capability	Environment	Title	Description	Action	Type	Available as of
Event Mesh	Integration Suite	Cloud Foundry	Retiring the name SAP Enterprise Messaging	SAP Enterprise Messaging has been renamed to SAP Event Mesh. We're in the process of changing all our documentation, enablement materials, and tools accordingly. See Blog  .	Info only	Announcement	2021-02-22
Enterprise Messaging	Integration Suite	Cloud Foundry	Resource Units	<p>You can now specify resource units at the message client level using the service descriptor and at the subaccount level using the user interface. See</p> <ul style="list-style-type: none"> Scope and Limitations Syntax for Service Descriptor Edit Resource Units for a Subaccount <p>Action: You can define the resource units required for your service instance in the service descriptor. An example of the code snippet to be added is provided here.</p>	Recommended	New	2021-01-20

2.1 2020 What's New for Enterprise Messaging (Archive)

2020

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Enterprise Messaging	Integration Suite	Cloud Foundry	UI Enhancement - Test Tab	The user interface for the Test tab has been enhanced with a split view.	Changed	2020-10-22

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Enterprise Messaging	Integration Suite	Cloud Foundry	REST APIs for Sending and Receiving Events	The service supports the use of REST APIs for publishing and consuming events that are compliant with the CloudEvents specification. Applications can publish and consume events with REST APIs using a REST client tool. See REST APIs for Events .	New	2020-10-22
Enterprise Messaging	Integration Suite	Cloud Foundry	Set Default Content-type	You can now set the default content-type while creating a webhook subscription. The webhook receives the message with the default content-type when the content-type for the message is not available or has the value application/octet-stream. See Manage Webhooks .	New	2020-09-09
Enterprise Messaging	Integration Suite	Cloud Foundry	Download Binary Messages	When you test consuming a message, if the message is in binary format, it's automatically downloaded as a file to your local machine. See Use a Message Client to Publish or Consume Messages .	Changed	2020-08-12
Enterprise Messaging	Integration Suite	Cloud Foundry	Message Properties	While testing message consumption in the user interface, you can now view details on the message headers in the Message Properties tab. See Use a Message Client to Publish or Consume Messages .	Changed	2020-08-12
Enterprise Messaging	Integration Suite	Cloud Foundry	Purge Messages	You can now delete all the messages in a queue at one go using the Purge Messages button under Actions. See Manage Queues .	New	2020-08-12
Enterprise Messaging	Integration Suite	Cloud Foundry	Monitor Resources	You can now view the number of messaging resources you've used in your subaccount. See Monitor Resources .	New	2020-08-12
Enterprise Messaging	Integration Suite	Cloud Foundry	Test Publish Messages to a Topic	You can now test publishing messages to a topic. See Use a Message Client to Publish or Consume Messages .	Changed	2020-04-21

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Enterprise Messaging	Integration Suite	Cloud Foundry	Displaying Additional Properties	You can now view additional properties for a queue under Actions . See Manage Queues .	Changed	2020-04-21

2.2 2019 Enterprise Messaging (Archive)

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Enterprise Messaging	Extension Suite - Digital Experience	Cloud Foundry	Deprecation of the lite service plan	<p>The lite service plan for SAP Event Mesh is being discontinued. It's recommended that you discontinue using service instances based on the lite service plan and create new service instances based on the default service plan. By migrating to the default service plan, you benefit from an optimized messaging infrastructure, which fully supports scalable and versatile API or event-based extension scenarios, see Initial Setup.</p> <p>The revised version of the service guide provides you with information specific to the default service plan.</p>	Changed	2019-10-10
Enterprise Messaging	Extension Suite - Digital Experience	Cloud Foundry	Publish and Consume Messages	You can now publish and consume message with a message client, see Test Publishing or Consuming Messages [page 67] .	New	2019-10-10

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Enterprise Messaging	Extension Suite - Digital Experience	Cloud Foundry	Syntax for Service Descriptor	You can now provide a version for the service descriptor. The attributes for the connection rules have changed, see Syntax for Service Descriptor [page 32] .	Changed	2019-10-10
Enterprise Messaging	Extension Suite - Digital Experience	Cloud Foundry	Support for Message Headers	You can now use message headers while sending and receiving messages using REST APIs, see REST APIs for Messaging [page 23] .	Changed	2019-09-03
Enterprise Messaging	Extension Suite - Digital Experience	Cloud Foundry	Default Plan	<p>The default plan provides a feature-rich user interface that allows you to:</p> <ul style="list-style-type: none"> • Create message clients with different user credentials • Communicate between different message clients within a subaccount using the message bus • Provide access rules for each message client • Create a message client with an event catalog • Use the event catalog API in a subaccount with business user credentials • Manage messaging and eventing tasks in a unified user interface <p>For more information, see</p> <ul style="list-style-type: none"> • Concepts [page 18] • Creating an Event Mesh Service Instance [page 42] • Syntax for Service Descriptor [page 32] 	New	2019-05-20

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Enterprise Messaging	Extension Suite - Digital Experience	Cloud Foundry	Messaging using REST APIs	<p>You can now:</p> <ul style="list-style-type: none"> Authenticate webhook URLs with OAuth client credentials Pause and resume webhook subscriptions Provide handshake exemption for a webhook URL <p>For more information, see REST APIs for Messaging [page 23].</p>	New	2019-05-20

2.3 2018 What's New for Enterprise Messaging (Archive)

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Enterprise Messaging	Integration	Cloud Foundry	Messaging Using REST API	<p>For messaging using REST APIs, you can now:</p> <ul style="list-style-type: none"> Create, Read, and Delete subscriptions in the dashboard Whitelist the subscription's WebHook URLs for handshake exemption in the dashboard APIs to initiate handshake for existing subscriptions Initiate handshake action for existing subscriptions in the dashboard 	Changed	2018-09-27
Enterprise Messaging	Integration	Cloud Foundry	REST APIs to Send and Receive Messages	You can now use REST APIs to send and receive messages. For more information, see Messaging using REST APIs [page 23] .	New	2018-09-13

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Enterprise Messaging	Integration	Cloud Foundry	Released Features	<p>You can:</p> <ul style="list-style-type: none"> Create, view, and delete webhook subscriptions in the dashboard Exempt handshake for the subscription's webhook URL in the dashboard Initiate handshake for existing webhook subscriptions in the dashboard Use REST APIs to initiate handshake for existing webhook subscriptions View the service descriptor JSON you provided during service instance creation in the dashboard 	Changed	2018-10-04

3 Scope and Limitations

Find information on the service availability and technical limitations for SAP Event Mesh, Default plan.

Regional Availability

Get an overview on the availability of Event Mesh according to region, infrastructure provider, and release status in the [Service Catalog](#) under [Service Plan](#).

i Note

The following limitations and limits are applicable for the default service plan.

Limitations

When applications poll for messages from queues using a REST API call, ensure that there's a delay of 5 seconds between consecutive polling requests on the same queue.

Allowed Limits

Message Size

- The maximum message size is 1 MB for all messaging protocols.
-
- The maximum storage space for all messages in all the queues per subaccount is 10 GB. If messages are above 1 MB, the AMQP 1.0 over WebSocket and MQTT 3.1.1 over WebSocket connections are closed. It's applicable for applications running on Cloud Foundry or on other platforms.
- The maximum message throughput per subaccount is 250 KB/s.

Messaging Resources

If messages are above 1 MB, the AMQP 1.0 over WebSocket and MQTT 3.1.1The service supports the following messaging resources:

- Connection - Represents an AMQP or MQTT connection.
- Endpoint - Represents either a queue or a topic endpoint. Topic endpoints are created when messages are directly consumed from a topic.
- Consumer - A consumer needs to be created to consume messages from an endpoint with AMQP or MQTT.
- Producer - A producer needs to be created to publish messages to a queue or topic with AMQP or MQTT.
- Queue Subscription – A queue subscription is created when a queue subscribes to a topic or topic pattern.

Resource Unit

The service uses a Resource Unit to group messaging resources. The following table explains resources that are part of 1 Resource Unit:

Messaging Resource	Allocation
Connections	1

Messaging Resource	Allocation
Endpoints	3
Consumers	3
Producers	3
Queue Subscriptions	15

Limits on Resource Units

The default resource unit allocation is 200 for newly created and existing subaccounts. The same allocation is applicable if you have a subaccount and you're using the service for the first time in the subaccount.

Create a ticket to increase the allocation for a subaccount up to 400 resource units through the business application to 1000 units. To learn more, see [Edit Resource Units for a Subaccount](#). The following table explains the default and maximum allocation for messaging resources in a subaccount:

Messaging Resource	Default	Maximum
Connections	400	1000
Endpoints	1200	3000
Consumers	1200	3000
Producers	1200	3000
Queue Subscriptions	6000	15000

Messaging Resources for REST APIs

- When webhooks are used to consume messages, each webhook gets 3 consumers.
- When messages are consumed with REST APIs by polling for messages, 3 consumers are used per queue or topic. These consumers are closed if there's no active publishing of messages for a period of 30 minutes.
- The following limits are applicable when only REST APIs are used:

Messaging Resource	Default	Maximum
Webhooks	400	1000
Queue Subscriptions	6000	15000

Related Information

[Messaging Protocols and Libraries \[page 21\]](#)

[Setting Up the SAP Event Mesh Default Plan \[page 40\]](#)

4 SAP Event Mesh Plans

Learn about the plans that make up the SAP Event Mesh service.

SAP Event Mesh Plans

Scenario	Capability	Advantages	Event Mesh Plan	Type	Event Errors Monitored in SAP Cloud ALM	Initial Setup
Manage a single event or message broker from the Business Technology Platform	Eventing and Messaging	<ul style="list-style-type: none"> • Pay for what you use • Let SAP maintain the hardware and software that hosts the event or message broker • Deploy event or message brokers in the BTP Cloud Foundry environment • Use the event or message broker as a service 	Default plan	External (for customers)	No	Initial Setup [page 40]
Enable the SAP-managed subscriptions that allow SAP applications in your customer landscape to share CloudEvents.	Eventing	<ul style="list-style-type: none"> • Benefit from managed event handling that simplifies information exchange between your SAP applications 	Internal service plans	Internal with some customer-facing features in the Event Mesh UI	Yes	SAP-managed setup

Scenario	Capability	Advantages	Event Mesh Plan	Type	Event Errors Monitored in SAP Cloud ALM	Initial Setup
Access the Event Mesh UI	Eventing and Messaging	<ul style="list-style-type: none"> Manage event and message capabilities from an intuitive user interface 	Standard plan	External (for customers)	No	Subscribing to the Event Mesh User Interface [page 52]

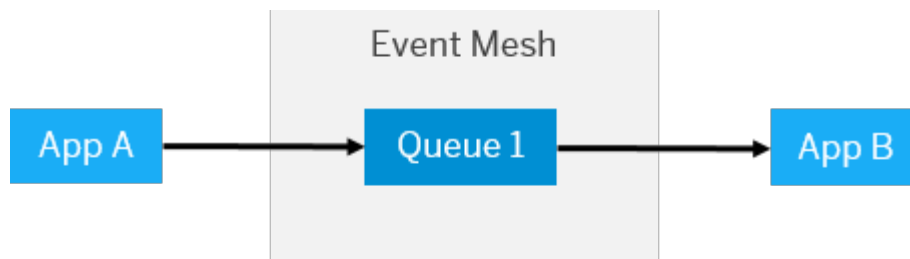
5 Concepts

The SAP Event Mesh Default plan employs a centralized event-oriented architecture. The service decouples communication between the sending and receiving applications and ensures the delivery of events and messages between them.

The Default service plan supports the following messaging and event-enabling concepts:

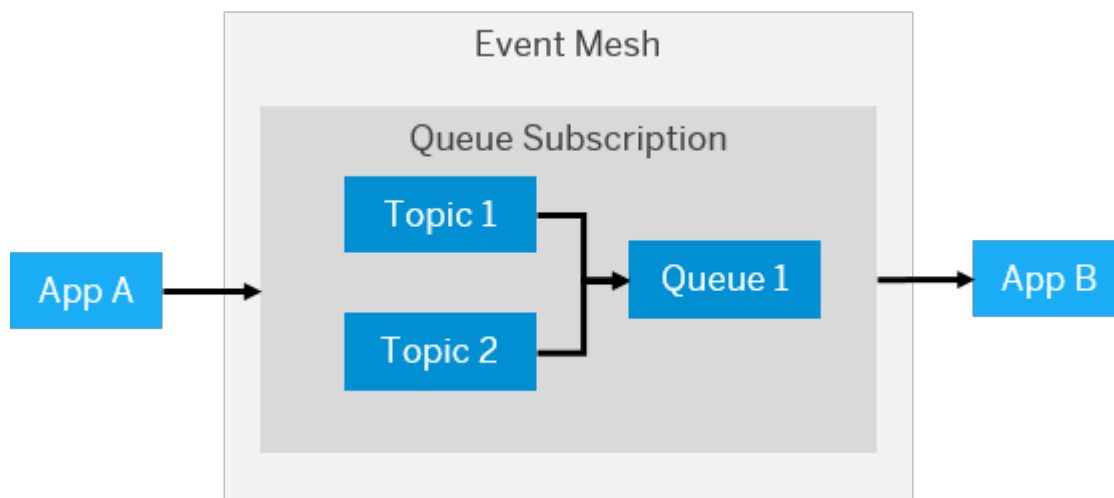
Queues

The service enables applications to communicate with each other through message queues. A sending application sends a message to a named queue. There's a one-to-one correspondence between a receiving application and its queue. The message queue retains messages until the receiving application consumes them. You can manage these queues using the service.



Queue Subscriptions

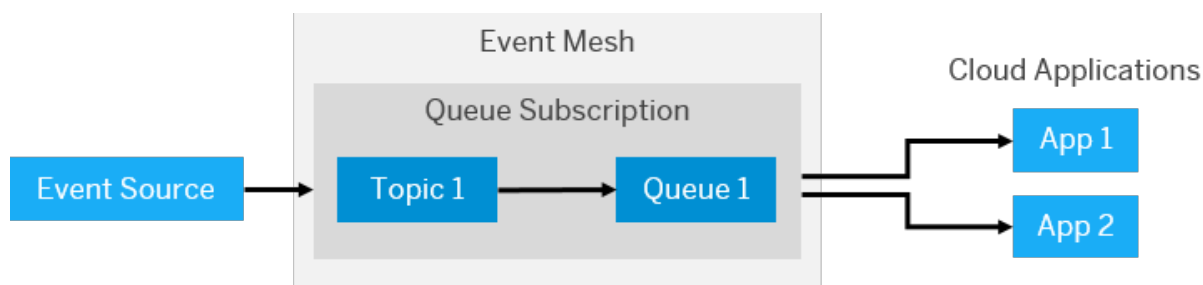
The service enables a sending application to publish messages and events to a topic. Topics don't retain messages but, it can be used when each message needs to be consumed by a number of receiving applications. Topics must be managed through queue subscriptions. In queue subscriptions, the service enables a sending application to publish messages to a topic that directly sends the message to a queue to which it's bound. For example, events from an SAP S/4HANA system can only be sent to a topic. A queue subscription ensures that a message sent to a topic is retained until it's consumed by the receiving application.



Events

An event is defined as a significant change in state, which indicates that something happened in the publishing application. Within Event Mesh, the actual event isn't distributed, as events occur in the publishing application. It's the event description that is distributed. To keep things simple, we use the terms event and event description to mean the same thing. When we say that events are distributed, we mean that the information about a change that occurred in the publishing application is distributed to subscribing applications.

An event source is the system or publishing application where the event originated. Different event sources can publish events to the service. A subscribing application needs to subscribe with Event Mesh to receive events.



Asynchronous Messaging

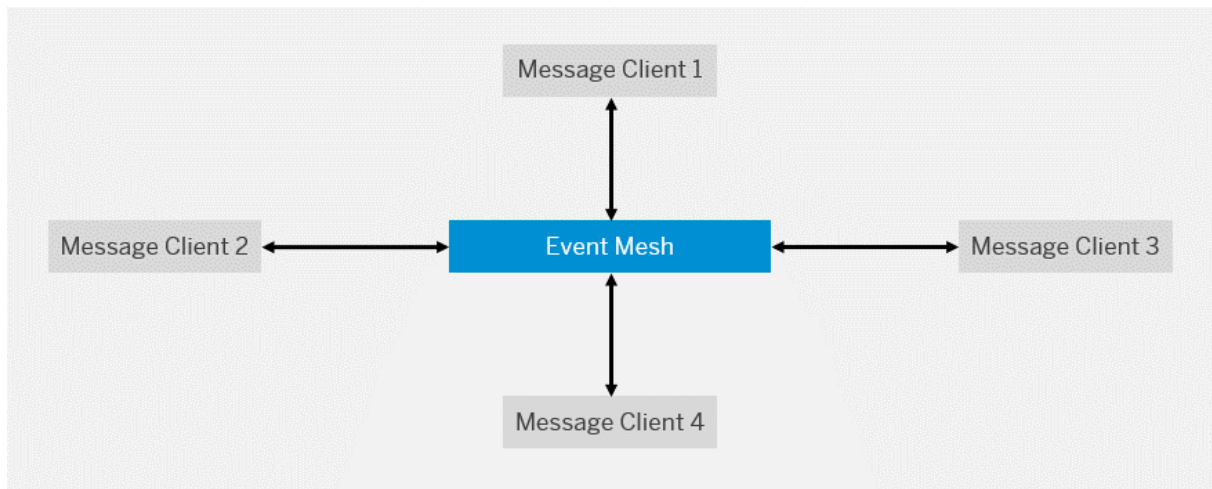
Asynchronous messaging allows communication between a sending and receiving application or system. Messages are sent to a queue where they're stored until the receiving application acknowledges them. The service provides capabilities for storing and transmitting messages through queues and queue subscriptions.

Message Client

A message client allows you to connect to the service through its unique credentials to send and receive messages. The message client can run within SAP BTP or outside it. You can create multiple message clients that can be distinguished with a set of credentials that consist of a namespace and connection rules. Also, the credentials must define the list of queues or topics to which the message client can send or receive messages. The credentials are stored in the service descriptor that you define while creating a service instance.

The namespace is a unique prefix that defines all the queues or topics that have been created in the context of a particular message client. When you manage queues or topics in the service, providing the namespace allows message clients to identify the queues or topics to which communication must be established. The connection rules specify the queue or topic to which a message client must publish or consume messages.

The default service plan supports a connection between different message clients in a subaccount through its unique credentials. Message clients can communicate with each other using the service. Each message client has a set of queues and topics associated with it. All these queues and topics belonging to 1 message client are exposed to other message clients through the unique credentials in the service descriptor.

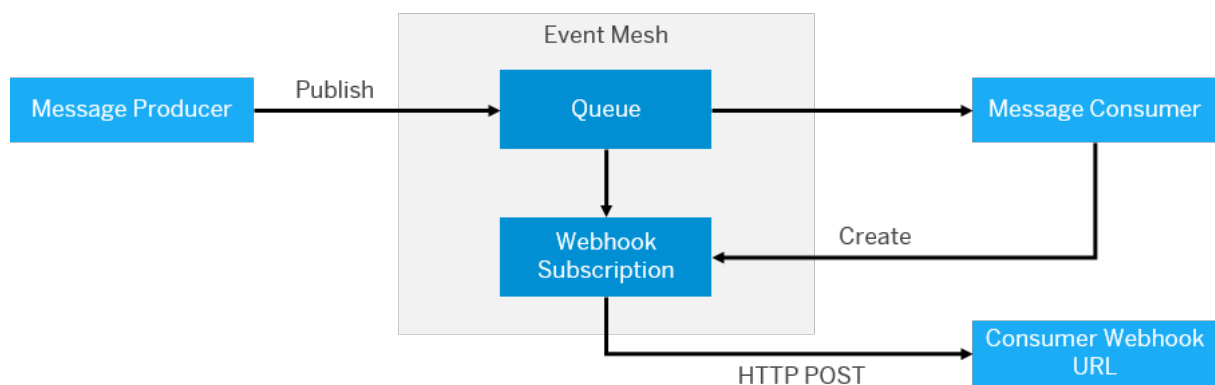


Event Catalog

An event source can have a list of events that can be published or consumed by the service. This list of events is known as event catalog. The event catalog contains events that can be either published or consumed. Each event in a catalog has a payload schema. An event source can provide an event catalog endpoint that provides the list of events to be published or consumed by the message client. The event catalog must adhere to the AsyncAPI specification, see [AsyncAPI Specification](#) .

Webhooks

Webhooks allow you to set up an integration to send real-time data from 1 application to another when an event occurs. A webhook subscription notifies the receiving application when an event is triggered. The data is sent through HTTP using a POST request to the receiving application that handles the data. The exchange of data is done through a webhook URL. A webhook URL is configured by the receiving application. The data sent to the webhook URL is known as payload. When an event is triggered, an HTTP POST payload is sent to the webhook URL. See [REST APIs for Messaging \[page 23\]](#).



[Messaging Protocols and Libraries \[page 21\]](#)

The service supports open standard messaging protocols and allows you to use client libraries for Java and Node.js.

[Syntax for Naming Queues, Topics, and Topic Patterns \[page 30\]](#)

The service follows a specific syntax for queue names, topic names, and topic pattern names to allow a stable configuration for business applications when messaging protocols are changed.

[Syntax for Service Descriptor \[page 32\]](#)

You must maintain a JSON file with parameters describing the attributes of the service instance. The JSON file is called service descriptor. In the service descriptor, you need to enter your message client name, the namespace, and maintain the options to define the access channel and rules for your message client.

[Event Mesh User Interface \[page 38\]](#)

The SAP Event Mesh, standard application plan provides access to the user interface that supports Event Mesh.

Related Information

[Scope and Limitations \[page 14\]](#)

[Syntax for Service Descriptor \[page 32\]](#)

[Syntax for Naming Queues, Topics, and Topic Patterns \[page 30\]](#)

[Messaging Protocols and Libraries \[page 21\]](#)


5.1 Messaging Protocols and Libraries

The service supports open standard messaging protocols and allows you to use client libraries for Java and Node.js.


Protocols

The Event Mesh service supports the following messaging protocols:

Advanced Message Queuing Protocol (AMQP) 1.0 over WebSocket

It's an open standard protocol used for messaging between applications or organizations. We recommend that you use AMQP 1.0 over WebSocket for messaging between applications running on Cloud Foundry. For more information, see [Specification for AMQP 1.0 over WebSocket](#) .

Message Queuing Telemetry Transport (MQTT) 3.1.1 over WebSocket

It's a lightweight messaging protocol designed specifically for constrained devices, low bandwidth, high latency, or unreliable devices. We recommend that you use MQTT 3.1.1 over WebSocket for messaging to a service from applications not running on the Cloud, for example, SAP S/4HANA. For more information, see [Specification for MQTT 3.1.1 over WebSocket](#) .

i Note

Quality of Service (QoS) is a feature of MQTT, where the protocol handles retransmission and guarantees that the message is delivered regardless of network reliability. MQTT 3.1.1 over WebSocket supports only QoS 0, QoS 1, and Messaging Gateway. The supported QoS levels are:

- At most once (0) - It guarantees its best effort with delivery. A message isn't acknowledged by the receiver, stored or redelivered by the sender.
- At least once (1) - It guarantees that a message is delivered at least once to the receiver. The message can also be delivered more than once.

REST APIs for Messaging

The service provides REST APIs for messaging. You can use these messaging REST APIs to send and receive messages.

i Note

Quality of Service (QoS) 0 and 1 are supported.

For more information, see [REST APIs for Messaging \[page 23\]](#).

REST APIs for Events

The service provides REST APIs for sending and receiving events. The events can be described using the CloudEvents v1.0 specification, see [CloudEvents Specification](#) .

i Note

Quality of Service (QoS) ATMOST_ONCE and ATLEAST_ONCE are supported.

For more information, see [REST APIs for Events \[page 26\]](#).

i Note

The order of delivery isn't guaranteed for the messaging protocols supported by the service.

Libraries

The Event Mesh service allows you to use the following libraries:

AMQP 1.0 over WebSocket libraries and MQTT 3.1.1 over WebSocket libraries for Node.js

It enables Node.js developers to connect and use the service. We recommend that you use these libraries to develop your application as doing so significantly reduces development effort. For more information, see [Node.js libraries](#).


Protocol Agnostic Libraries

The service supports protocol agnostic libraries that can be used at the application configuration level for Java and Node.js. It allows you to work with messaging applications without getting into the intricacies of a

messaging protocol. You can create an application configuration that defines an input (source from which the application receives messages) and output (destination to which the application sends messages). The application configuration is typically a JSON file with properties such as input and output configuration, quality of service and so on. Once you create the application configuration, you can use it in an environment variable to start the application. Use protocol agnostic libraries in the application code to create stream objects that define the source, destination, quality of service and other technical properties from the application configuration JSON file. The advantages are as follows:

- The messages can be made available through stream object. The application isn't dependent on a messaging protocol for relaying messages.
- The application configuration resides outside the application code. You can use the same application code and change only the application configuration for different landscapes. For example, when you move the application from development to production landscape, the properties in the application configuration change. In this scenario, only the application configuration has to be changed and the application code remains the same.

Apache QPID JMS Client Library

It enables Java developers to connect and use the service. We recommend that you use these libraries to develop your application as doing so significantly reduces development effort, see [Java libraries](#) .

Related Information

[REST APIs for Messaging \[page 23\]](#)

[Development \[page 54\]](#)

5.1.1 REST APIs for Messaging

The service supports the use of REST APIs for publishing and consuming messages. Message client applications with REST-based messaging can implement the messaging functionality using a REST client tool.

Publish Messages to Queues and Topics

The service provides REST APIs to publish messages to queues and topics in a RESTful way. To publish messages to a queue or a topic, the client must make a REST API call specifying the queue or topic name as the path parameter and message data in the request body. In the publish request, the client must provide a mandatory header `Content-Type` which is set as a property for the message. APIs for publishing messages treat the message data as binary data and publish the message as is to a queue or topic without any processing. The APIs for publishing messages publish only one message at a time for each HTTP request.

Quality of Service (QoS)

The allowed values for specifying QoS are 0 and 1. The API for publishing messages requires a mandatory header `x-qos`. When the client calls the API for publishing messages with QoS 0, the service tries to deliver the

message and returns an HTTP response with code 204 irrespective of whether the message is delivered. The best effort is made to deliver the message but, doesn't guarantee that the message is sent to queue or topic. If a message client calls the API for publishing messages with QoS 1 for guaranteed delivery, then, the service responds with the HTTP response code 204. If the 204 response code isn't received, it's the client's responsibility to retry until the response code 204 is received.

Time to Live

With every publish request, the message client can set a time to live for the message in milliseconds. If the API is called without this header, the default value of 2592000000 (30 days) is used.

Consume Messages from Queues

The service supports receiving a message in the following ways:

- Pull based model – A receiving application makes a REST API call to read a message from a queue. The response body of the HTTP request contains the message data. If the message client calls with QoS 1, then, the service needs an acknowledgment request from the message client with the `message-id`. The message is deleted from the queue only after the acknowledgment is consumed. When QoS 0 is used, the message is deleted from the queue without an acknowledgment from the client.
- Push based model – The service pushes messages consumed on the queue to the webhook associated with the queue subscription. The message client needs to request a create subscription API call. The subscription must contain the subscription name, queue name, details of the webhook, and quality of service.

Webhook Subscriptions

Webhook Authentication

The webhook has to be secured by HTTPS for subscriptions. Webhooks with basic authentication, OAuth 2.0 authentication, and no authentication are supported. The authentication applies to both handshake and message delivery calls. For OAuth 2.0 client credentials protected webhooks, the token URL is called with an HTTP POST method and client credentials are sent in a basic authentication header.

❖ Example

OAuth 2.0 Token Request

```
POST <token-url>?grant_type=client_credentials
Headers:
Authorization: Basic <base-64-encoded-client-ID-and-secret>
```

Handshake

When a client creates a subscription to a queue with a webhook, the service makes a handshake request before pushing the messages to the webhook. The handshake request is an OPTIONS call to the webhook with the HTTP header `WebHook-Request-Origin`. The webhook must respond to a handshake request with the HTTP header `WebHook-Allowed-Origin` and the value of this header has to match the value of the request header

WebHook-Request-Origin. The webhook can also respond with the value '*' for the WebHook-Allowed-Origin header value. After the handshake for the subscription is completed successfully, the service publishes messages to the webhook.

❖ Example

Handshake call where the webhook URL is protected with basic authentication:

```
OPTIONS <webhook-url>
Headers:
Authorization: Basic <base64-encoded-basic-auth-credentials>
WebHook-Request-Origin: <origin-of-enterprise-messaging-service>
```

❖ Example

Handshake call where the webhook URL is protected with OAuth 2.0 authentication:

```
OPTIONS <webhook-url>
Headers:
Authorization: Bearer <access-token-received-from-call-to-token-url>
WebHook-Request-Origin: <origin-of-enterprise-messaging-service>
```

Message Delivery

After you complete the handshake, the service can deliver the messages available in a queue by sending a POST request to the webhook with the header Content-Type set to the message's Content-Type property. The request body contains the message data and the request header X-Subscription-Name contains the subscription name.

❖ Example

Message delivery request where the webhook URL is protected with OAuth 2.0 authentication:

```
POST <webhook-url>
Headers:
Authorization: Bearer <access-token-received-from-call-to-token-url>
Content-Type: <content-type-of-message>
X-Subscription-Name: <webhook-subscription-name>
Request Body:
<message-data>
```

Message Delivery Rate

The service sends one message with one request at a time to the webhook.

Pause or Resume a Webhook Subscription

After you create a webhook you can pause or resume a webhook subscription, for example, for carrying out maintenance activities or upgrade for the consumer application. You can stop receiving messages from a webhook temporarily using pause and consume messages again when you're ready using resume.

Quality of Service (QoS)

- Pull based model: For QoS 0, the message is auto-acknowledged, delivered to the client, and removed from the queue. With QoS 1, an explicit acknowledgment API has to be invoked after which the message is removed from the queue.

- Push based model: Subscriptions can be created with QoS 0 and QoS 1. In the case of QoS 1, when the service sends the POST request to the webhook with a message, it needs an acknowledgment from the webhook as a response to the POST request. All 2XX HTTP response codes are considered as positive acknowledgments by the webhook and the messages are deleted from the queue. For any other response codes with QoS 1, the service tries again to deliver the message to the webhook. When you use QoS 0, the service deletes the message from the queue after the POST request to the webhook for message delivery.

i Note

The webhook URL that you provide must be available on the public internet. Additionally, if the webhook URL is running in an on-premise system, it can be accessed through the cloud connector configuration used to connect to the on-premise system in your subaccount. On-premise webhooks with basic authentication and no authentication are supported.

Related Information

[Use REST APIs to Send and Receive Messages \[page 55\]](#)

[Use REST APIs to Manage Queues and Queue Subscriptions \[page 56\]](#)


[Specification for messaging REST APIs](#)

[Specification for management REST APIs](#)

5.1.2 REST APIs for Events

The service supports the use of REST APIs for publishing and consuming events that are compliant with the CloudEvents specification. Applications can publish and consume events with REST APIs using a REST client tool.

Supported CloudEvents Versions

CloudEvents is a specification for describing event data in common formats to provide interoperability across services, platforms, and systems. [CloudEvents Specification v1.0](#)  of the CloudEvents specification is supported.

The REST APIs for events support publishing of events in structured and binary modes. For the structured mode, only the JSON event format is supported.

Publish Events

The service provides REST APIs to publish events in a RESTful manner. To publish an event, the client must make an HTTP call by specifying the event data and headers in compliance with the HTTP binding of

CloudEvents specification. Along with the event data and headers, an additional header with the Quality of Service can also be specified. If you don't set a header, the default value is AT_LEAST_ONCE.

When you publish a CloudEvent, the event is routed to a topic address constructed from the source and type attributes of the CloudEvent as shown in the following example:

❖ Example

topic := <three-segment source w/o leading slash>/ce/<type with dots replaced by slashes>

In the following sample CloudEvent; topic := default/org.my.crm/1/ce/org/my/crm/Customer/Added/v1

```
{
  "specversion": "1.0",
  "id": "b9976b88-b446-493a-84eb-432cda902349",
  "source": "/default/org.my.crm/1",
  "type": "org.my.crm.Customer.Added/v1",
  "subject": "customerId:123",
  "time": "2020-09-17T09:56:24Z",
  "data": {
    "customerId": "123"
  }
}
```

Quality of Service with Event Publish (QoS)

The allowed values for specifying the Quality Of Service header are AT_LEAST_ONCE and AT_MOST_ONCE. When a client calls the service API for publishing events with AT_LEAST_ONCE, the service tries to deliver the event and returns an HTTP response with code 204 irrespective of whether the event is published or not. The best effort is made to publish the event but, it doesn't guarantee that the event is published to the service. If a client calls the API for publishing events with AT_MOST_ONCE then, the service responds with the HTTP response code 204 only if the event is published. If the 204 response code isn't received, it's the client's responsibility to retry until the response code 204 is received. The default value for QoS is AT_LEAST_ONCE.

Time to Live

With every publish request, the client can set a time to live for the message in milliseconds. If the API is called without this header, the default value 2592000000 is used.

Consume Events with Webhook Subscriptions

The service supports consuming events through webhook subscriptions. The service pushes events to the webhook through the client that requests a create subscription API call. The webhook subscription must contain the subscription name, event source, event type, webhook authentication details, and quality of service.

When you want to consume CloudEvents using HTTP, a webhook subscription must be created as shown in the following example:

❖ Example

```
POST /sap/ems/v1/events/subscriptions
Host: https://enterprise-messaging-pubsub.cfapps.eu10.hana.ondemand.com
Authorization: Bearer ...
```

```
Content-Type: application/json
{
  "name": "crm-ext-webhook",
  "events": [{
    "source": "/default/org.my.crm/1",
    "type": "org.my.crm.Customer.Added.v1"
  },
  {
    "source": "/default/org.my.crm/1",
    "type": "org.my.crm.Customer.Deleted.v1"
  }
],
  "webhookUrl": "<URL of my-crm-ext app>"
}
```

The webhook subscription internally creates a queue whose lifecycle depends on the webhook subscription, that is, the queue is deleted when the subscription is deleted.

The queue that gets created is named <namespace of message client>/ce/webhook/<webhook subscription name>. In this example, the name of the queue is default/org.my.crm.ext/1/ce/webhook/crm-ext-webhook.

The queue is configured to get events from all topics based on the source/type pairs defined in the webhook subscription request. In this example, the queue subscribes to the following topics:

- default/org.my.crm/1/ce/org/my/crm/Deleted/Added/v1
- default/org.my.crm/1/ce/org/my/crm/Customer/Deleted/v1

Webhook Authentication

The webhook has to be secured by HTTPS for subscriptions. Webhooks with basic authentication, OAuth 2.0 authentication, and no authentication are supported. The authentication applies to both handshake and message delivery calls. For OAuth 2.0 client credentials protected webhooks, the token URL is called with an HTTP POST method and client credentials are sent as basic authentication header. Delivery of events to a webhook can be done only in the binary mode.

❖ Example

OAuth 2.0 Token Request

```
POST <token-url>?grant_type=client_credentials
Headers:
Authorization: Basic <base-64-encoded-client-ID-and-secret>
```

Handshake

When a client creates a subscription to an event with a webhook, the service makes a handshake request before pushing events to the webhook. The handshake request is an OPTIONS call to the webhook with the HTTP header WebHook-Request-Origin. The webhook must respond to a handshake request with the HTTP header WebHook-Allowed-Origin and the value of this header has to match the value of the request header WebHook-Request-Origin. The webhook can also respond with the value '*' for the WebHook-Allowed-Origin header value. After the handshake for the subscription is completed successfully, the service publishes messages to the webhook.

❖ Example

Handshake call where the webhook URL is protected with basic authentication:

```
OPTIONS <webhook-url>
```

Headers :

```
Authorization: Basic <base64-encoded-basic-auth-credentials>
WebHook-Request-Origin: <origin-of-enterprise-messaging-service>
```

❖ Example

Handshake call where the webhook URL is protected with OAuth 2.0 authentication:

```
OPTIONS <webhook-url>
```

Headers :

```
Authorization: Bearer <access-token-received-from-call-to-token-url>
WebHook-Request-Origin: <origin-of-enterprise-messaging-service>
```

Message Delivery

After you complete the handshake, the service can deliver the events published by making a POST request to the webhook. The delivery of events done in the binary mode.

❖ Example

Event delivery request where the webhook URL is protected with OAuth 2.0 authentication:

```
POST <webhook-url>
```

Headers :

```
Authorization: Bearer <access-token-received-from-call-to-token-url>
WebHook-Request-Origin: <origin-of-enterprise-messaging-service>Content-Type:
<content-type-of-event>
X-Subscription-Name: <webhook-subscription-name>
//Additional headers for binary mode of events complying to the HTTP binding
in CloudEvents specification.//
```

Request Body:

```
<event-data>
```

Event Delivery Rate

The service sends one event with one request at a time to the webhook.

Pause or Resume a Webhook Subscription

After creating a webhook, you can pause or resume it. For example, to carry out maintenance activities or upgrade for a consumer application. You can stop receiving messages from a webhook temporarily using pause and consume messages again when you're ready using resume.

Quality of Service with Webhooks

Webhook subscriptions can be created with QoS AT_LEAST_ONCE and AT_MOST_ONCE. In the case of Quality of Service AT_LEAST_ONCE, the service sends a POST request to the webhook with an event. The service expects an acknowledgment from the webhook as a response to the POST request. All 2XX HTTP response codes are considered as positive acknowledgments by the webhook and the events are deleted from the queue. For any other response code with QoS AT_LEAST_ONCE, the service tries to deliver the event to the webhook again. With QoS AT_MOST_ONCE, the service deletes the event from the queue after the POST request to the webhook for event delivery.

Related Information

[Use REST APIs to Send and Receive Events \[page 57\]](#)

5.2 Syntax for Naming Queues, Topics, and Topic Patterns

The service follows a specific syntax for queue names, topic names, and topic pattern names to allow a stable configuration for business applications when messaging protocols are changed.

Syntax for Queue Names

When you send a message via AMQP 1.0 over WebSocket, it can be published to a queue. If the target address of a message starts with "queue:" the rest is the queue name. For example, a message with the target address "queue:orders" is sent to the queue "orders". The same schema applies to queue subscriptions.

The syntax for queue names is as follows:

- The queue name must consist of one or more characters such as "A to Z", "a to z" or "0–9", ":", "-", "_", and "/".
- The total length is limited to 100 characters.

Note

All queue names must have a namespace as a prefix, for example, <namespace>/myqueue. See Namespace in [Syntax for Service Descriptor \[page 32\]](#).

Some examples of valid queue names are "A", "Ab", "AB", "ab", "a/b", "a-b", "a.b", "a_b", "a-b/c", and "_".

Some examples of invalid queue names are "a+b", "a\b", "a b", ".", "/", "/a", "a/", and "a//a".

Syntax for Topic Names

If you send a message using MQTT 3.1.1 over WebSocket, it can only be published to a topic, not a queue. Then, the MQTT topic name is the target topic name. When you send a message via AMQP 1.0 over WebSocket, it can be published to a topic. If the target address of a message starts with topic: the rest of the address is the topic name. For example, a message with the target address "topic:changenotif" is published to the topic "changenotif". Each event is assigned to one topic. Topics form a logical tree to organize messages in a folder-like hierarchy in a file system. Therefore, topics appear as strings, consisting of multiple segments, separated by one defined delimiter such as file paths. A topic syntax depends on the protocol that is used. Every time the protocol is changed, the topic syntax must also be changed to facilitate efficient use of mechanisms defined by individual messaging protocols.

The syntax for topic names is as follows:

- Topics must consist of one or more segments.
- Forward slash "/" (ASCII 0x2F) must be used as a segment separator.
- The segment separator must not appear at the beginning or ending of a valid topic.
- The segments must consist of one or more characters such as "A to Z", "a to z" or "0–9".
- Up to 20 segments can be used.
- Empty segments aren't allowed.

- The total topic length is limited to 150 characters.

The following are examples of valid topic names:

- <namespace>/Production/Confirmation/Created
- <namespace>/Process/Order/Released
- <namespace>/Sales/Order/Created

The following are examples of invalid topic names:

- //Production/Order/Released
- <namespace>//Order/Released
- <namespace>/Sales/Sales Order/Created

An event producer can use its own schema for the topic structure. The naming conventions for event topics are as follows (default plan):

- <namespace>/<application-specific topic>
- <up to 63 characters>/<up to 87 characters>

Example:

- default/mycompany.myapp/1/Production/Order/Released (The topic length is 51 characters)
- default/mycompany.myapp/1/Sales/Order/Released (The topic length is 46 characters)
- default/mycompany.myapp/42/GW/Service/Activated (The topic length is 47 characters)

Syntax for Topic Pattern Names

Topic patterns are used when you create queue subscriptions. The following segments are supported for topic pattern names:

- + (ASCII 0x2B) represents a single segment with any content that is valid, but always the whole segment.
- * (ASCII 0x2A) represents a subtree of segments (zero, one or more segments) and can only appear at the end of the filter.

The following are examples of valid topic pattern names:

- default/mycompany.myapp/1/Production/+/Created
- default/mycompany.myapp/1/Sales/*

The following are examples of invalid topic pattern names:

- default/mycompany.myapp/1/Production/Conf+/Created
- default/mycompany.myapp/1/Sales/*/Released

The following are examples of topic matching:

- default/mycompany.myapp/1/Production/* matches default/mycompany.myapp/1/Production/Order/Released
- default/mycompany.myapp/1/Production/* matches WDF/ERP1/BO/Production/Order
- default/mycompany.myapp/1/Production/* doesn't match default/mycompany.myapp/1/Production // the subtree is empty
- default/mycompany.myapp/1/Production/* doesn't match default/mycompany.myapp/1
- default/mycompany.myapp/1/Production/* doesn't match default/mycompany.myapp/1/Sales

Note

The names you create are not intended for personal data.

5.3 Syntax for Service Descriptor

You must maintain a JSON file with parameters describing the attributes of the service instance. The JSON file is called service descriptor. In the service descriptor, you need to enter your message client name, the namespace, and maintain the options to define the access channel and rules for your message client.

General Guidelines

- Maintain rules in your service descriptor. A message client can't publish or consume messages if rules aren't defined in the service descriptor.
- Maintain the options for a message client, for example, allow the usage of management REST APIs, messaging REST APIs, messaging (AMQP 1.0) only if the client requires it.
- Use semantically relevant namespaces when defining your service descriptor.
- Publish to your own namespace only. For example, `rules.topicRules.publishFilter` entries must always start with `${namespace}/...`
- Define your scenario in a consumption driven manner. For example, define your own queues and create a subscription for these queues to required topics.
- It's recommended that you publish to topics and not to queues directly. For example, `rules.queueRules.publishFilter` entry isn't required if you follow this guideline.
- Consume from your own queues only. For example, `rules.queueRules.subscribeFilter` entries must always start with `${namespace}/...`
- Publish to your own queues only. Ensure you remain in your own namespace for end-to-end business scenarios. For example, `rules.queueRules.publishFilter` entries must always start with `${namespace}/...`
- Avoid using queues with foreign namespaces.

i Note

The names you create aren't intended for personal data.

The following sections describe the syntax for the service descriptor properties that you can define in the JSON during service instance creation:

emname

Description: It specifies the name of the message client. `emname` is used in the Event Mesh business application to identify message clients with ease.

Syntax:

- Type: attribute
- Allowed characters: `[a-zA-Z0-9_-]`
- Max Length: 100
- Required: true

Guideline: It must be unique for a subaccount. It's recommended that you use the same value for service instance name and `emname`.

version

Description: It specifies the version of the service descriptor.

Syntax:

- Type: attribute
- Allowed versions: 1.1.0
- Required: true

Guideline: If you're using the new syntax for rules, the field "version": 1.1.0 is required. The version field is optional when you use the deprecated syntax.

namespace

Description: It ensures that every message client within a subaccount is unique. The queues managed by the message client and topics used to publish by the message client have the namespace as prefix. The namespace must be provided as a prefix and isn't done automatically. If a message client is used in a business scenario as a reuse service, the namespace must be globally unique. Instead of a technical client ID, it's recommended to use a namespace that ends with `-` or `svc` in the last segment, for example, `default/sap.myapp/-` for provider instances. Consumer instances aren't allowed to use namespaces that end with `-` or `svc`.

Syntax:

- Type: attribute
- Segments: exactly three (firstSegment/secondSegment/thirdSegment)
- Allowed characters:
 - First segment: [a-zA-Z0-9]
 - Hyphens and dots aren't allowed.
 - Recommended value: default
 - Second segment: [a-zA-Z0-9-.]
 - Starts with a character or number
 - Followed by `.` or `-`
 - Must contain at least one character or number before or after the period or hyphen.
 - Third segment: [a-zA-Z0-9-.]
 - Use the same guidelines given for the second segment.
 - For provider scenarios, use a hyphen.
- Max Length: 63
- Required: true

Guideline: It must:

- be unique within a subaccount
- contain exactly 3 segments, for example, a/b/c
- follow the construction rule `<region>/<applicationNamespace>/<instanceId>`. The following values are recommended:
 - Region: default
 - Application namespace: `<vendor.<application>>`, for example, `mycompany.myapp.mysubapp` without the top-level domain, in `sap.com` remove the `.com`.
 - Instance ID: a fixed number, for example, `default/sap.myapp/1`. For multiple instances or message clients of the same type, use `<region>/<applicationNamespace>/*`

- have the namespace as a prefix for each queue name handled by the message client.

options

Description: It's used to define the privileges and access channels for a message client.

Syntax:

- Type: object
- Required: false
 - Attribute: management
 - Default: false
 - Allowed values: true | false
 - Attribute: messagingrest
 - Default: false
 - Allowed values: true | false
 - Attribute: messaging
 - Default: true
 - Allowed values: true | false

Guideline:

- The management attribute enables or disables the usage of the management REST APIs.
- The messagingrest attribute enables or disables the usage of the messaging REST APIs.
- The messaging attribute enables or disables the usage of the messaging (AMQP 1.0).

rules

Description: It defines the publish or consume privileges for a message client.

In order to allow access to a queue or a topic, the namespace of the corresponding owner message client has to be added. The placeholder `${namespace}` can be used instead of the defined namespace. For example, if the namespace `"namespace": "default/sap.myapp/1"` is defined, the most basic rule is `${namespace}/*`. Ensure you provide the attribute `publishFilter` if you want to send messages and the attribute `subscribeFilter` to consume messages. If you omit these attributes, you can't access messaging. You can omit these attributes if you want to use only the management scenario. Rules support the following wildcards:

- `+` (ASCII 0x2B) represents a single segment with any content that is valid, but always the whole segment.
- `*` (ASCII 0x2A) represents a subtree of segments (zero, one or more segments) and can only appear at the end of the rule.

Syntax:

- Type: object
- Required: false
- Attribute: queueRules
 - Type: object
 - Attribute: publishFilter
 - Type: array
 - Default: None
 - Allowed characters: `[a-zA-Z*+ /]`
 - Sample values: `${namespace}/foo/bar/*`, `${namespace}/*`, `${namespace}/+`

- Allows a message client to send messages to a queue that matches with the defined rule.
- Attribute: subscribeFilter
 - Type: array
 - Default: None
 - Allowed characters: [a-zA-Z*+/]
 - Sample values: \${namespace}/foo/bar/*, \${namespace}/*, \${namespace}/+
 - Allows a message client to receive messages from a queue that matches with the defined rule.
- Attribute: topicRules
 - Type: object
 - Attribute: publishFilter
 - Type: array
 - Default: None
 - Allowed characters: [a-zA-Z*+/]
 - Sample values: \${namespace}/foo/bar/*, \${namespace}/*, \${namespace}/?, \${namespace}/+
 - Allows a message client to send messages to a topic that matches with the defined rule.
 - It's recommended that you publish to your own namespace only, for example, the rule must be \${namespace}/*
 - Attribute: subscribeFilter
 - Type: array
 - Default: None
 - Allowed characters: [a-zA-Z*+/]
 - Sample values: \${namespace}/foo/bar/*, \${namespace}/*, \${namespace}/+, +/+/+/foo/bar/*
 - Allows a message client to subscribe to a topic that matches with the defined rule.

Sample Code

```
"rules": {
  "queueRules": {
    "publishFilter": [
      "${namespace}/q1",
      "${namespace}/q2/+/*",
      "other/namespace/id/q1"
    ],
    "subscribeFilter": [
      "${namespace}/q1",
      "${namespace}/q2/*"
    ]
  },
  "topicRules": {
    "publishFilter": [
      "${namespace}/a",
      "${namespace}/a/+/*"
    ],
    "subscribeFilter": [
      "${namespace}/a/*",
      "+/+/*"
    ]
  }
}
```

Resource Units

Description: Resource Units is a collection of messaging resources such as queues, connections, and so on, required for a message client. You can specify this value in the service descriptor to allocate messaging

resources based a specific business scenario. For example, if you create a service instance with 12 resource units, you're allowed to have 12 queues, 12 webhooks, 12 connections, 36 consumer, 36 producers, and 180 queue subscriptions. There's no limit on message spooling. If you don't define Resource Units, all messaging resources can be used based on the general limits that are applicable for a message client.

The following table describes one resource unit:

Resource	Allocation	Description
Connections	1	Number of connections that can be established per service instance; Total number of connections is the sum of connections created using AMQP 1.0, MQTT, and REST APIs for Messaging respectively.
Endpoints	3	Total number of queues that can be created; sum of queues and topic endpoints.
Consumers	3	The maximum number of consumers (only controlled per connection).
Producers	3	The maximum number of producers (only controlled per connection).
Queue Subscriptions	15	The maximum number of producers (only controlled per connection).
Message Spool	10 MB	The maximum size of all messages that can be stored in the queues.

Syntax:

- Type: attribute
- Default: 10
- Allowed values: 10–50
- Required: false

Guideline: See [Scope and Limitations \[page 14\]](#).

Sample Code

```
{
  ...
  "resources" : {
    "units" : "10"
  }
  ...
}
```

xs-security

Description: The xs-security object is used to configure XSUAA-related properties, for example, the supported credential-types for an instance. The first credential-type in the array is set as default. If a binding request is made without any parameters the default credential-type (binding-secret) is used. The order provided is considered in scenarios when there are multiple credential types. Use the following syntax for new service instances:

Syntax:

- Type: object
- Required: false
- Attribute: oauth2-configuration
 - Type: object
 - Attribute: credential-types
 - Type: array
 - Default: ["binding-secret", "x509"]
 - Allowed values: "binding-secret" | "x509"

(Optional) You can update existing service instances with x509 as the credential-type.

Note

- If you remove a credential-type from your configuration, existing secrets become invalid.
- It's recommended to change credential types in phases on live applications without downtime.
- Only XSUAA-managed certificates are allowed.

To change the credential-type, perform the following steps:

1. Check the existing credential-type for your service instance. To identify the type:
 - binding-secret: If your secrets in the binding look like "6969bada-d7e0-4752-b04c-2c274dacc3c1\$4SABS2kIH9gZY0vCoY53wrLpOIGpqNLljBOy3Ndy2CY="
 - instance-secret: If your secrets in the binding look like "9cA3TFAOFD+A/AHv9eOgx3HrDYg="

If your binding has recently been updated and contains the JSON property UAA, you can also check the credential-type property.
2. Add the new credential-type using update service instance, for example,

```
credential-types: ["x509", "<your current credential-type>"]
```

to allow x509 for new binding and your current credential-type for existing bindings (binding-secret or instance-secret).

3. Rebind to ensure all bindings are using the new credentials.
4. Remove the old credential-type using update service instance, for example, remove binding-secret and only allow credential-types: ["x509"].

Sample Code

JSON snippet of the service descriptor

```
{
  "xs-security": {
    "oauth2-configuration": {
      "credential-types": ["binding-secret", "x509"]
    }
  }
}
```

When you update to a new credential-type without specifying an existing one, you get an error '400 Bad Request'.

If an existing or default credential-type are unknown. It is possible to specify all three credential-types such as credential-types: ["binding-secret", "x509", "instance-secret"]. This is a valid configuration. After an update is completed, the credential-types that are not required can be removed in the next update.

Examples:

- Current credential-type is instance-secret update to credential-types: ["binding-secret", "x509"], you get an error '400 Bad Request'.
- Current credential-type is instance-secret and binding-secret update to credential-types: ["x509"], you get an error '400 Bad Request'.
- Current credential-type is instance-secret update to credential-types: ["binding-secret", "x509", "instance-secret"] is a valid update.
- Current credential-type is instance-secret and binding-secret update to credential-types: ["binding-secret", "x509", "instance-secret"] is a valid update.

Related Information

[Creating an Event Mesh Service Instance \[page 42\]](#)

[Creating an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface \[page 44\]](#)

5.4 Event Mesh User Interface

The SAP Event Mesh, standard application plan provides access to the user interface that supports Event Mesh.

SAP BTP cockpit

The SAP BTP cockpit is where you set up your SAP Event Mesh plans. You should be familiar with cockpit basic concepts before you get started. For more information, see [Basic Platform Concepts](#).

Subscriptions Page

The [Subscriptions](#) page of the Event Mesh UI is available when you have the Event Mesh Integration Administrator role assigned to your user in your BTP subaccount. From this page, view the SAP-managed subscriptions that are configured for the SAP applications in your landscape.

When subscriptions are listed in the [Subscriptions](#) table, you can enable them. If there's nothing listed in the table, then this page isn't supported by your plan. For more information, see [Enabling SAP Event Subscriptions \[page 72\]](#).

Monitoring Page

The [Monitoring](#) page of the Event Mesh UI is available when you have the Event Mesh Integration Administrator role assigned to your user in your BTP subaccount. From this page, register Event Mesh with SAP Cloud ALM to monitor event errors that occur when CloudEvents are passed between SAP applications. For more information, see [Registering Event Mesh with SAP Cloud ALM \[page 74\]](#).

i Note

You can't monitor default plan events. This page supports SAP-managed events.

Default Plan Menu

The [Default Plan](#) menu is available when you have Enterprise Messaging roles assigned to your user in your BTP subaccount. This menu gives you access to the actions that you perform when using the Event Mesh default plan. For more information, see [Using Event Mesh Default Plan \[page 59\]](#).

Related Information

[Get Started with the Event Mesh User Interface \[page 49\]](#)

[SAP Event Mesh Plans \[page 16\]](#)

6 Initial Setup

After you purchase the default plan for Event Mesh, you must perform the following steps in the SAP BTP cockpit.

[Get Started with the Event Mesh Default Plan \[page 40\]](#)

Learn how to find your service entitlement and create an Event Mesh instance.

[Get Started with the Event Mesh User Interface \[page 49\]](#)

When you subscribe to the Event Mesh standard application Mesh, you gain access to the user interface that supports the service.

6.1 Get Started with the Event Mesh Default Plan

Learn how to find your service entitlement and create an Event Mesh instance.

[Setting Up the SAP Event Mesh Default Plan \[page 40\]](#)

After you purchase the default plan for Event Mesh, find your service entitlement and set up a service instance in your SAP BTP cockpit subaccount.

[Creating an Event Mesh Service Instance \[page 42\]](#)

Use the SAP BTP cockpit to create an instance of the Event Mesh service (default plan).

[Creating an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface \[page 44\]](#)

You can create an Event Mesh (default plan) service instance using the Cloud Foundry Command Line Interface (cf CLI).

[Binding an Application to an Event Mesh Service Instance \[page 46\]](#)

Bind your messaging application to your Event Mesh service instance for the default service plan.

6.1.1 Setting Up the SAP Event Mesh Default Plan

After you purchase the default plan for Event Mesh, find your service entitlement and set up a service instance in your SAP BTP cockpit subaccount.

Prerequisites

- You have a customer account with SAP BTP and an Event Mesh default service plan. See [Getting Started with a Customer Account: Workflow in the Cloud Foundry Environment](#).
- You have a global account that has the entitlement to use Event Mesh. See [Getting a Global Account](#).

- You've created a subaccount. See [Create a Subaccount in the Cloud Foundry Environment](#).
- You've created a space within the subaccount in which Cloud Foundry is enabled. See [Managing Orgs and Spaces Using the Cockpit](#).

Context

The Event Mesh default service plan:

- Allows cross communication between message clients
- Provides a fine granular access control system with message clients that have a unique credential for a service instance
- Allows automatic scaling
- Provides complete enablement for event-driven scenarios
- Supports cross communication scenarios, shared and nonshared instances.

Procedure

1. In the SAP BTP cockpit, navigate to your global account and assign the entitlement for the default service plan for Event Mesh to your subaccount.
 - a. Open your global account and choose ► [Entitlements](#) ► [Subaccount Assignments](#) ►.
 - b. In the dropdown list, select your subaccount and choose [Go](#).
 - c. Choose ► [Configure Entitlements](#) ► [Add Service Plans](#) ►.
 - d. In the [Subaccount Entitlements](#) dialog box, select the Event Mesh service.
 - e. In the [Service Details: Event Mesh](#) window, select the default service plan.
 - f. Choose [Add 1 Service Plans](#) to add this entitlement for the Event Mesh service for your subaccount.
 - g. Choose [Save](#).
2. Create a service instance for Event Mesh using either the cockpit or cf CLI.
 - [Creating an Event Mesh Service Instance \[page 42\]](#).
 - [Creating an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface \[page 44\]](#).

→ Remember

The lite service plan for SAP Event Mesh is being discontinued. It's recommended that you discontinue using service instances based on the lite service plan and create new service instances based on the default service plan. By migrating to the default service plan, you benefit from an optimized messaging infrastructure, which fully supports scalable and versatile API or event-based business scenarios.

3. (Optional) To bind a messaging application to your service instance, see [Binding an Application to an Event Mesh Service Instance \[page 46\]](#).
4. (Optional) To consume SAP S/4HANA Cloud events using the extensions capability on SAP BTP, see [Enable the Consumption of SAP S/4HANA Cloud Events](#).

Results

After setting up Event Mesh, learn how to [Get Started with the Event Mesh User Interface \[page 49\]](#).

Related Information

[Configure Entitlements and Quotas for Subaccounts](#)

[Subscribe to Multitenant Business Applications in the Cloud Foundry Environment Using the Cockpit](#)

6.1.2 Creating an Event Mesh Service Instance

Use the SAP BTP cockpit to create an instance of the Event Mesh service (default plan).

Prerequisites

You have access to a Cloud Foundry space. See [Create Spaces](#).

Context

You need to create a service instance in the cockpit to use the Event Mesh service. If you haven't created a service instance before, see [Creating Service Instances](#).

Procedure

1. Open the cockpit and navigate to your subaccount.
2. Expand **Services** > **Instances and Subscriptions**.
3. Choose **Create**.
4. Select Event Mesh and the **default** service plan.
5. Select your Cloud Foundry space, enter an instance name, and choose **Next**.
6. Specify parameters using a JSON file.

You can provide additional parameters such as the namespace and connection rules for a message client. Follow the [syntax for service descriptor \[page 32\]](#).

Example

```
{
```

```

"emname": "<emname>",
"namespace": "<namespace e.g. a/b/c>",
"version": "1.1.0",
"options": {
  "management": true,
  "messagingrest": true,
  "messaging": true
},
"rules": {
  "queueRules": {
    "publishFilter": [
      "${namespace}/*"
    ],
    "subscribeFilter": [
      "${namespace}/*"
    ]
  },
  "topicRules": {
    "publishFilter": [
      "${namespace}/*"
    ],
    "subscribeFilter": [
      "${namespace}/*"
    ]
  }
},
"xs-security": { //optional//
  "oauth2-configuration": {
    "credential-types": [
      "binding-secret",
      "x509"
    ]
  }
}
}

```

Note

- See [Syntax for Service Descriptor](#) to understand the different parameters that you can provide in the example.
- The default value for resource units is 10.
- We recommend that you use the same value for `service instance name` and `emname`.
- For existing service instances, update your service instance to include new parameters in your JSON file. See [Creating an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface \[page 44\]](#).

7. Review the information you've provided and choose [Create](#).
 - If you want to receive events from SAP S/4HANA using the extensions capability on SAP BTP, see [Enable the Consumption of SAP S/4HANA Cloud Events](#).
 - If you want to delete an Event Mesh service instance, choose [Delete](#) under [Actions](#).

Related Information

[Code Samples on GitHub for Java](#) 📄

[Code Samples on GitHub for Node.js](#) 📄

6.1.3 Creating an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface

You can create an Event Mesh (default plan) service instance using the Cloud Foundry Command Line Interface (cf CLI).

Prerequisites

- You have the administrator role for your global account.
- You have access to a Cloud Foundry space.

Procedure

1. Log on to the Cloud Foundry Environment using the Command Line Interface. See [Log On to the Cloud Foundry Environment Using the Command Line Interface](#).
2. Select your organization and the space.
3. Enter `cf marketplace` to verify the availability of the Event Mesh service in the service marketplace.
4. Access an Event Mesh service instance using one of the following methods:
 - To create a new service instance using the default plan, enter `cf create-service enterprise-messaging default <service-instance-name> -c "service-descriptor.json"`. Follow the [syntax for service descriptor \[page 32\]](#).

❖ Example

```
cf create-service enterprise-messaging default <yourmessageclientname> -c '{
  "emname": "<yourmessageclientname>",
  "namespace": "<yourorgname>/<yourmessageclientname>/<uniqueID>",
  "version": "1.1.0",
  "options": {
    "management": true,
    "messagingrest": true,
    "messaging": true
  },
  "rules": {
    "queueRules": {
      "publishFilter": [
        "${namespace}/*"
      ],
      "subscribeFilter": [
        "${namespace}/*"
      ]
    },
    "topicRules": {
      "publishFilter": [
        "${namespace}/*"
      ],
      "subscribeFilter": [
        "${namespace}/*"
      ]
    }
  }
}
```

```

    ]
  },
  "resources" : {
    "units" : "10"
  },
  "xs-security": { //optional//
    "oauth2-configuration": {
      "credential-types": [
        "binding-secret",
        "x509"
      ]
    }
  }
}
}'

```

Note

We recommend that you use the same value for `service instance name` and `emname`.

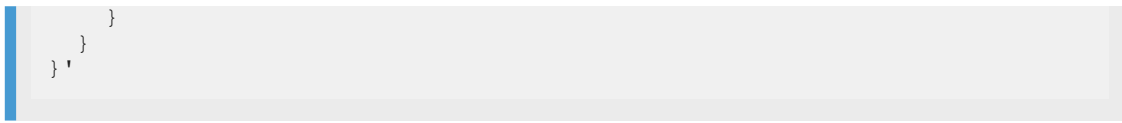
- If the application is deployed in the same space, you can use an existing service instance. Enter `cf services` to find existing instances of the service in your space. To update an existing service instance, enter `cf update-service <service-instance-name> -c 'service-descriptor.json'`

❖ Example

```

cf update-service <yourmessageclientname> -c
'{
  "emname": "<yourmessageclientname>",
  "namespace": "<yourorgname>/<yourmessageclientname>/<uniqueID>",
  "version": "1.1.0",
  "options": {
    "management": true,
    "messagingrest": true,
    "messaging": true
  },
  "rules": {
    "queueRules": {
      "publishFilter": [
        "${namespace}/*"
      ],
      "subscribeFilter": [
        "${namespace}/*"
      ]
    },
    "topicRules": {
      "publishFilter": [
        "${namespace}/*"
      ],
      "subscribeFilter": [
        "${namespace}/*"
      ]
    }
  },
  "resources" : {
    "units" : "25"
  },
  "xs-security": { //optional//
    "oauth2-configuration": {
      "credential-types": [
        "binding-secret",
        "x509"
      ]
    }
  }
}

```



Related Information

[Edit Resource Units for a Subaccount](#)

6.1.4 Binding an Application to an Event Mesh Service Instance

Bind your messaging application to your Event Mesh service instance for the default service plan.

Prerequisites

- You deployed a messaging application in the same space as your service instance. See [Deploy Business Applications in the Cloud Foundry Environment](#).
- You created an Event Mesh service instance. See [Creating an Event Mesh Service Instance \[page 42\]](#).

Context

When you bind an application to the service, the application receives a JSON file with the binding configuration. The binding configuration contains specifications for the different messaging protocols that can be used to connect to the service. OAuth is used to connect to the messaging protocol endpoints. Before establishing a connection, the application must generate a token using the client ID, client secret, token endpoint, and grant type in the `oa2` section of the binding configuration. The `protocol` section contains the URL with the host name to which the applications must connect to use respective client libraries.

The following code shows the structure of a service binding for Event Mesh with credential-type binding secret:

```
{  
  "uaa": {  
    "xsappname": "sb-clientid",  
    "credential-type": "binding-secret",  
    "clientid": "clientid",  
    "clientsecret": "secret",  
    "url": "<your URL>"  
  },  
  "management": [  
    {  
      "oa2": {  
        "clientid": "clientid",  
        "clientsecret": "secret",  
        "tokenendpoint": "<your tokenendpoint>",  
      }  
    }  
  ]  
}
```

```

        "granttype": "client_credentials"
    },
    "uri": "<your URI>"
  }
],
"messaging": [
  {
    "oa2": {
      "clientid": "clientid",
      "clientsecret": "secret",
      "tokenendpoint": "<your tokenendpoint>",
      "granttype": "client_credentials"
    },
    "protocol": [
      "amqp10ws"
    ],
    "broker": {
      "type": "sapmgw"
    },
    "uri": "<your URI>s"
  },
  {
    "oa2": {
      "clientid": "sb-clientid",
      "clientsecret": "secret",
      "tokenendpoint": "<your tokenendpoint>",
      "granttype": "client_credentials"
    },
    "protocol": [
      "mqtt311ws"
    ],
    "broker": {
      "type": "sapmgw"
    },
    "uri": "<your URI>"
  },
  {
    "oa2": {
      "clientid": "sb-clientid",
      "clientsecret": "secret",
      "tokenendpoint": "<your tokenendpoint>",
      "granttype": "client_credentials"
    },
    "protocol": [
      "httprest"
    ],
    "broker": {
      "type": "saprestmgw"
    },
    "uri": "<your URI>"
  }
],
"namespace": "<your namespace>",
"serviceinstanceid": "<your service instance id>"
}

```

The following code shows the structure of a service binding for Event Mesh with credential-type x509 for managed certificates:

```

{
  "xsappname": "clientid",
  "uaa": {
    "clientid": "<your client id>",
    "xsappname": "<your xsappname>",
    "credential-type": "x509",
    "certificate": "-----BEGIN CERTIFICATE-----...-----END CERTIFICATE-----",
    "key": "-----BEGIN RSA PRIVATE KEY-----...-----END RSA PRIVATE KEY-----",

```

```

    "certurl": "<your certificate URL>"
  },
  "management": [
    {
      "oa2": {
        "clientid": "<your client id>",
        "tokenendpoint": "<your certificate URL>",
        "granttype": "client_credentials"
      },
      "uri": "<your URI>"
    }
  ],
  "messaging": [
    {
      "oa2": {
        "clientid": "<your client id>",
        "tokenendpoint": "<your certificate URL>",
        "granttype": "client_credentials"
      },
      "protocol": [
        "amqp10ws"
      ],
      "broker": {
        "type": "sapmgw"
      },
      "uri": "<your URI>"
    },
    {
      "oa2": {
        "clientid": "<your client id>",
        "tokenendpoint": "<your certificate URL>",
        "granttype": "client_credentials"
      },
      "protocol": [
        "mqtt311ws"
      ],
      "broker": {
        "type": "sapmgw"
      },
      "uri": "<your URI>"
    },
    {
      "oa2": {
        "clientid": "<your client id>",
        "tokenendpoint": "<your certificate URL>",
        "granttype": "client_credentials"
      },
      "protocol": [
        "httpprest"
      ],
      "broker": {
        "type": "saprestmgw"
      },
      "uri": "<your URI>"
    }
  ],
  "namespace": "<your namespace>",
  "serviceinstanceid": "<your service instance id>"
}

```

iNote

- The segment management in the service binding information is available only if you have set the option management as true during service instance creation.

- The segment `messaging` in the service binding information is available only if you have set the option `messagingrest` as `true` during service instance creation.

Procedure

1. Navigate to the space that contains the deployed application and service instance.
2. Select one of the following methods to proceed:

View	Steps
Application	<ol style="list-style-type: none"> 1. Open your Cloud Foundry space. 2. In the navigation pane, select Applications, then the application you've deployed. 3. In the navigation pane, choose Service Bindings. 4. Choose Bind Service. 5. Select Service from the catalog, then Event Mesh. 6. Select the Default service plan. 7. (Optional) Specify parameters in a JSON format or browse and upload a JSON file. 8. Provide the name of the service instance you've created. 9. Choose Finish.
Service instance	<ol style="list-style-type: none"> 1. Open the service instance that you've created. 2. Choose Bind. 3. Select the application you've deployed. 4. (Optional) Specify parameters in a JSON format or browse and upload a JSON file. 5. Save your changes.

You can also use cf CLI to bind an application to your service instance using the following command:

```
cf bind-service APP-NAME SERVICE_INSTANCE {-c PARAMETERS_AS_JSON}
```

Related Information

[Bind Service Instances to Applications Using the Cockpit](#)

[Bind Service Instances to Applications Using the Cloud Foundry Command Line Interface](#)

6.2 Get Started with the Event Mesh User Interface

When you subscribe to the Event Mesh standard application plan, you gain access to the user interface that supports the service.

[Setting Up an Event Mesh Standard Application Plan \[page 50\]](#)

To access the Event Mesh user interface, you must add an entitlement for the standard application plan to your subaccount in SAP BTP cockpit.

[Assigning Roles to Users \[page 51\]](#)

The service provides standard roles for typical user profiles. You can configure application roles and then assign users to these roles using the SAP BTP cockpit.

[Subscribing to the Event Mesh User Interface \[page 52\]](#)

Subscribe to the Event Mesh business application to access its user interface.

6.2.1 Setting Up an Event Mesh Standard Application Plan

To access the Event Mesh user interface, you must add an entitlement for the standard application plan to your subaccount in SAP BTP cockpit.

Prerequisites

- You have a customer account with SAP BTP. See [Getting Started with a Customer Account: Workflow in the Cloud Foundry Environment](#).
- You have a global account that has the entitlement to use Event Mesh. See [Getting a Global Account](#).
- You created a subaccount. See [Create a Subaccount in the Cloud Foundry Environment](#).

Procedure

In the SAP BTP cockpit, add the standard application plan to the Event Mesh service.

- a. Navigate to your global account open your subaccount.
- b. Choose [Entitlements](#).
- c. Verify that you see a row for the Event Mesh service.
- d. Click [Configure Entitlements](#) > [Add Service Plans](#).
- e. In the [Subaccount Entitlements](#) dialog box, select the Event Mesh service.
- f. In the [Service Details: Event Mesh](#) window, select the standard application plan.
- g. Choose [Add 1 Service Plans](#) to add this entitlement for the Event Mesh service for your subaccount.
- h. Click [Save](#).

Results

You're ready to assign role permissions and subscribe to the Event Mesh user interface. Once you subscribe, a link to the user interface is available in SAP BTP cockpit.

Related Information

[Assigning Roles to Users \[page 51\]](#)

[Using Event Mesh Default Plan \[page 59\]](#)

6.2.2 Assigning Roles to Users

The service provides standard roles for typical user profiles. You can configure application roles and then assign users to these roles using the SAP BTP cockpit.

Prerequisites

- You have the administrator role for your global account.
- You verified that you have an entitlement for Event Mesh standard application plan in your BTP subaccount. For more information, see [Configure Entitlements and Quotas for Subaccounts](#).
- You're using one or both of the following trust configurations:
 - Default trust configuration (SAP ID service), see [Default Identity Federation with SAP ID Service in the Cloud Foundry Environment](#).
 - Custom trust configuration (Identity Authentication service or any SAML 2.0 identity provider), see [Establish Trust and Federation with UAA Using Any SAML Identity Provider](#).

Context

Roles are assigned in the BTP cockpit. You can assign roles to users based on the type of trust configuration that you enabled in SAP BTP. The following options are available:

- Assign a role collection directly to a user.
- Map role collections to user groups defined in the identity provider. You initially maintain the mapping between user groups and role collections once in SAP BTP, and maintain group memberships of users in the identity provider.

If you're using the default trust configuration with SAP ID service, you assign role collections directly to a user. However, if you're using a custom trust configuration, for example, with the Identity Authentication service, you can use both options.

i Note

The following roles are for internal use only.

- OperationsManageRole
- OperationsReadRole
- OperationsReadDetailRole

Procedure

1. Determine the roles or role collections to assign to your users.
For more information on roles, see [User Roles for Event Mesh \[page 77\]](#).
2. In BTP cockpit, navigate to your subaccount and assign roles or role collections to users.

Results

Once you have role permissions, you can access the Event Mesh user interface. You'll find the link in the BTP cockpit subaccount where you subscribed to the Event Mesh standard plan.

Related Information

[Working with Role Collections](#)

6.2.3 Subscribing to the Event Mesh User Interface

Subscribe to the Event Mesh business application to access its user interface.

Prerequisites

- You have an entitlement for Event Mesh standard plan in your BTP subaccount. For more information, see [Configure Entitlements and Quotas for Subaccounts](#).
- You added roles or role collections to the users that need to access the Event Mesh user interface. For more information, see [Assigning Roles to Users \[page 51\]](#).

Context

The Event Mesh user interface is provided as a multi-tenant business application. Subscriptions can be set up only by administrators of the global account.

Procedure

1. Open your global account and then your subaccount.
2. In the left pane, choose *Instances and Subscriptions*.
3. Click *Create*.
4. Scroll through the Service list and select *Event Mesh* and the standard plan.
5. Click *Create*.

The *Go to Application* link becomes available in the *Subscriptions* tab once the subscription is activated.

Note

If you try to access the UI before roles are assigned to your user, then caching issues can prevent access. To resolve this issue, see [Guided Answers](#) .

Results

Users with permissions can click the application link and log in to the Event Mesh user interface.

Related Information

[Assigning Roles to Users \[page 51\]](#)

[User Roles for Event Mesh \[page 77\]](#)

[Using Event Mesh Default Plan \[page 59\]](#)

[SAP Event Mesh Plans \[page 16\]](#)

7 Development

When using the Default plan, you can deploy a messaging application to SAP Event Mesh that you developed using Java or Node.js.

Develop Applications on SAP BTP

To learn how to develop Event Mesh applications on SAP BTP, see [Tutorials](#) .

Build Applications Using Java

To build messaging applications using Java, see [Samples on GitHub for Java](#) .

Build Applications Using Node.js

To build messaging applications using Node.js, see [Samples on GitHub for Node.js](#) .

Use REST APIs

To use the REST APIs provided with the service, see [REST APIs for Development \[page 54\]](#).

7.1 REST APIs for Development

This section contains information on the APIs that can be used for development with Event Mesh.

Related Information

[Use REST APIs to Send and Receive Messages \[page 55\]](#)

[Use REST APIs to Manage Queues and Queue Subscriptions \[page 56\]](#)

[REST APIs for Messaging \[page 23\]](#)

7.1.1 Use REST APIs to Send and Receive Messages

Use REST APIs to send and receive messages with Event Mesh.

Prerequisites

- Ensure you set the `messagingrest` parameter to true while creating your service instance.
- You've generated a service key. See [Creating Service Keys](#).

Context

You can access the messaging APIs [here](#).

Procedure

1. Open a REST client tool.
2. In the "messaging" section for the protocol "httprest" of the service key, note the following:
 - Client ID in the "oa2" section
 - Client Secret in the "oa2" section
 - Token endpoint in the "oa2" section
 - Base URL in the "uri" section
3. Perform the following steps to get the access token:
 - a. Append `?grant_type=client_credentials&response_type=token` to the token endpoint and do a POST request to receive the OAuth token response.
 - b. Provide the client ID and client secret as the username and password for basic authentication.
 - c. Obtain the `access_token` generated as a part of the response.

❖ Example

```
Curl command: curl -X POST -H "Authorization: Basic <base64-encoding-of 'client_id:client_secret'>"  
"<TOKENENDPOINT>?grant_type=client_credentials&response_type=token"
```

4. Add the access token you've obtained in the previous step `Authorization: Bearer <access_token>` as the authorization header to access `messagingrest` endpoints.

i Note

The messaging APIs are protected with an OAuth bearer token.

5. To publish a message to a queue, make a POST call with the authorization header as shown in the example:

❖ Example

Queue name: org/sample/001/q1

After URL encoding: org%2Fsample%2F001%2Fq1

Resultant REST API call: <BaseURL>/messagingrest/v1/queues/org%2Fsample%2F001%2Fq1/messages

Curl command: curl -X POST -H "Authorization: Bearer <access_token>" -H "x-qos: 1" -H "Content-Type: application/json" "<BaseURL>/messagingrest/v1/queues/org%2Fsample%2F001%2Fq1/messages" -d '{"property": "value"}'

7.1.2 Use REST APIs to Manage Queues and Queue Subscriptions

Use REST APIs to manage queues and queue subscriptions with Event Mesh.

Prerequisites

- Ensure you set the `management` parameter to `true` while creating your service instance.
- You've generated a service key. See [Creating Service Keys](#).

Context

You can access the management APIs [here](#).

Procedure

1. Open a REST client tool.
2. In the "management" section of the service key, note the following:
 - Client ID in the "oa2" section
 - Client Secret in the "oa2" section
 - Token endpoint in the "oa2" section
 - Base URL in the "uri" section
3. Perform the following steps to get the access token:
 - a. Append `?grant_type=client_credentials&response_type=token` to the token endpoint and do a POST request to receive the OAuth token response.

- b. Provide the client ID and client secret as the username and password for basic authentication.
- c. Obtain the `access_token` generated as a part of the response.

❖ Example

Curl command: `curl -X POST -H "Authorization: Basic <base64-encoding-of 'client_id:client_secret'>" "<TOKENENDPOINT>?grant_type=client_credentials&response_type=token"`

4. Add the access token you've obtained in the previous step `Authorization: Bearer <access_token>` as the authorization header to access management endpoints.

i Note

The management APIs are protected with an OAuth bearer token.

5. To obtain information for a queue with name `org/sample/001/q1`, URL encode the queue name and make a get call to the resultant URL as shown in the example:

❖ Example

Queue name: `org/sample/001/q1`

After URL encoding: `org%2Fsample%2F001%2Fq1`

Resultant REST API call: `<BaseURL>/hub/rest/api/v1/management/messaging/queues/org%2Fsample%2F001%2Fq1`

Curl command: `curl -X GET -H "Authorization: Bearer <access_token>" "<BaseURL>/hub/rest/api/v1/management/messaging/queues/org%2Fsample%2F001%2Fq1"`

7.1.3 Use REST APIs to Send and Receive Events

Use REST APIs to send and receive events with Event Mesh.

Prerequisites

- Ensure you set the `messagingrest` and `management` parameter to true while creating your service instance.
- You've generated a service key. See [Creating Service Keys](#).

Context

You can access the Events APIs [here](#).

Procedure

1. Open a REST client tool.
2. In the `management` or `messagingrest` section of the service key, note the following:
 - Client ID in the `"oa2"` section
 - Client Secret in the `"oa2"` section
 - Token endpoint in the `"oa2"` section
 - Base URL in the `"uri"` section
3. Perform the following steps to get the access token:
 - a. Append `?grant_type=client_credentials&response_type=token` to the token endpoint and do a POST request to receive the OAuth token response.
 - b. Provide the client ID and client secret as the username and password for basic authentication.
 - c. Obtain the `access_token` generated as a part of the response.

❖ Example

Curl command: `curl -X POST -H "Authorization: Basic <base64-encoding-of 'client_id:client_secret'>" "<TOKENENDPOINT>?grant_type=client_credentials&response_type=token"`

4. For all requests, use this access token `Authorization: Bearer <access_token>` as the authorization header to access events endpoints.
5. Based on whether you choose binary or structured mode, ensure you provide the headers required according to the CloudEvents specification, see [REST APIs for Events \[page 26\]](#). The following example is for publishing a structured cloud event:

❖ Example

Curl command: `curl -X POST -H "Authorization: Bearer <access_token>" -H "Content-Type: application/cloudevents+json" "<BaseURL>/sap/ems/v1/events" -d '{"specversion": "1.0", "source": "<your-event-source>", "type": "<your-event-type>", "id": "05022092-B698-444E-9F8B-A0B4605F163C", "datacontenttype": "application/json", "subject": "<event-subject>", "data": {"property1": "value", "property2": "value"}}'`

8 Using Event Mesh Default Plan

The Event Mesh user interface allows you to manage messaging clients and event catalogs for a BTP subaccount.

To access the user interface, you need to subscribe to the Event Mesh Standard application plan in the SAP BTP cockpit. To learn more, see [Subscribing to the Event Mesh User Interface \[page 52\]](#).

[View Rules \[page 59\]](#)

You can view the connection rules you've provided in the JSON file during service instance creation.

[Manage Queues \[page 60\]](#)

Queues enable point-to-point communication between two applications. An application can subscribe to a queue.

[Manage Webhooks \[page 62\]](#)

You can use a webhook to subscribe to a queue. It allows the service to push messages received on the queue to the webhook.

[View Event Catalog for a Message Client \[page 64\]](#)

You can view the event catalog for a particular message client.

[View Service Descriptor \[page 64\]](#)

You can view the JSON file you provided during service instance creation.

[View Event Catalog for a Subaccount \[page 65\]](#)

You can view the event catalog for a message client that gets created when you've integrated the service to receive cloud events from SAP S/4HANA Cloud.

[Monitor Resources \[page 66\]](#)

You can view details on the utilization of messaging resources in your subaccount.

[Test Publishing or Consuming Messages \[page 67\]](#)

You can publish or consume messages through a message client. The messages are published or consumed based on the connection rules that you've provided in the service descriptor.

[Edit Resource Units for a Subaccount \[page 68\]](#)

You can edit the number of resource units allocated to your subaccount.

8.1 View Rules

You can view the connection rules you've provided in the JSON file during service instance creation.

Prerequisites

- You've completed the [initial setup \[page 40\]](#).

- You have the ReadRole assigned. See [Assigning Roles to Users \[page 51\]](#).

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose [Instances and Subscriptions](#) from the navigation pane on the left, then the [Event Mesh](#) tile.
3. Choose [Go to Application](#).
4. Open the required message client and then, choose [Rules](#).

You can see the following information:

- List of connection rules you've provided during service creation, for example, `sap/messaging/1/queue1`.
- Type to which the rule belongs, for example, queue or topic.
- Permissions defined for the rule, for example, publish or subscribe.

8.2 Manage Queues

Queues enable point-to-point communication between two applications. An application can subscribe to a queue.

Prerequisites

- You've completed the [initial setup \[page 40\]](#).
- You have the ManageRole assigned. See [Assigning Roles to Users \[page 51\]](#).

Context

Queue names must follow the naming convention specified in [Syntax for Naming Queues, Topics, and Topic Patterns \[page 30\]](#).

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose [Instances and Subscriptions](#) from the navigation pane on the left, then the [Event Mesh](#) tile.
3. Choose [Go to Application](#).

4. Open the required message client and then, choose [Queues](#).

You can see the following information:

- List of queues.
 - Number of messages stored in each queue.
 - Number of unacknowledged messages in each queue.
 - Size (in bytes) of all the messages stored in each queue.
 - Access type for each queue.
 - Under [Actions](#), you can view the following additional information:
 - Whether the queue has been configured to respect message TTLs (Time-To-Live), see [Glossary \[page 85\]](#)
 - Maximum queue size in bytes
 - Maximum message size in bytes
 - Maximum number of unacknowledged messages that have been delivered each time
5. To create a new queue:
 - a. Choose [Create Queue](#).
 - b. Enter a queue name.
 - c. Choose [Create](#).
 6. Under Actions,
 - choose [Delete Queue](#) that corresponds to the queue you want to delete.
 - choose [Queue Subscriptions](#) to create a new queue subscription associated with the selected queue. Enter a topic or topic pattern name and choose [Add](#).
 - choose [View Details](#) to see details related to the queue.
 - choose [Purge Messages](#) to delete all the messages in the queue.

Caution

Deleting a queue also deletes any associated queue subscriptions and messages (if any).

Related Information

[REST APIs for managing queues](#)

8.3 Manage Webhooks

You can use a webhook to subscribe to a queue. It allows the service to push messages received on the queue to the webhook.

Prerequisites

- You've completed the [initial setup \[page 40\]](#).
- You've provided "messagingrest": true while creating a service instance. For more information, see [Creating an Event Mesh Service Instance \[page 42\]](#).
- You have the ManageRole assigned. See [Assigning Roles to Users \[page 51\]](#).
- (Optional) If you're using an on-premise webhook URL, you've connected to the on-premise system using Cloud Connector. See [Initial Configuration](#).

Context

The webhook subscription contains the following:

- subscription name
- queue name
- details of the webhook URL
- subscription status
- handshake status
- quality of service

The subscription status can be active or handshake pending. The handshake gives permission to the service to send messages to the webhook. The handshake status can have the following values:

- Not Initiated: The service hasn't requested a handshake.
- Requested: The handshake has been requested and the webhook responded with a response code 2xx but, the response header WebHook-Allowed-Origin doesn't match with the request header WebHook-Allowed-Origin or "*".
- Completed: The handshake has been completed.
- Failed: The service has requested a handshake and the webhook responded with a code other than 2xx.
- Exempted: You can obtain a handshake exemption by adding the webhook URL under handshake exemptions.

❖ Example

POST Request

Headers

Key Value

Authorization: Basic <base64-encoded-username-pwd>

Content-Type: application/x-www-form-urlencoded

Body

```
grant_type: client_credentials
```

See [REST APIs for Messaging](#).

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose [Instances and Subscriptions](#) from the navigation pane on the left, then the [Event Mesh](#) tile.
3. Choose [Go to Application](#).
4. Open the required message client and then, choose [Webhooks](#).

You can see the following information:

- List of webhook subscriptions available for your service instance
 - List of associated queues
 - Quality of Service used
 - Subscription status
 - Handshake status
5. To create a new webhook subscription:
 - a. Choose [Create Webhook](#).
 - b. Enter a subscription name and provide the webhook URL.
 - c. Select the queue, quality of service and authentication.

Note

The list of queues in your message client is populated.

- d. Change the toggle button setting if you want a handshake exemption for the webhook URL that you've provided.
 - e. Change the toggle button setting if the webhook URL that you've provided is an on-premise webhook. Provide the URL and location ID that you used to configure the Cloud Connector to the on-premise system.
 - f. Provide a default content-type then, select an authentication type.
 - g. Choose [Create](#).
6. Under Actions, choose:
 - [View Details](#) to view details the webhook subscription details and its history. Use the [Refresh](#) button to fetch latest details.
 - [Trigger Handshake](#) to initiate a handshake for corresponding webhook subscription.
 - [Pause](#) to pause the message delivery to the webhook URL.
 - [Resume](#) to resume the message delivery to the webhook URL.
 - [Delete Webhook Subscription](#) to delete the corresponding webhook subscription.

Related Information

[Specifications for messaging REST APIs](#)
[Concepts](#)

8.4 View Event Catalog for a Message Client

You can view the event catalog for a particular message client.

Prerequisites

- You've completed the [initial setup \[page 40\]](#).
- You have the ReadRole assigned. See [Assigning Roles to Users \[page 51\]](#).
- You've integrated the service with SAP S/4HANA to receive cloud events, see [Enable the Consumption of SAP S/4HANA Cloud Events](#).

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose [Instances and Subscriptions](#) from the navigation pane on the left, then the [Event Mesh](#) tile.
3. Choose [Go to Application](#).
4. Open the required message client and then, choose [Events](#).

You can see the list of events and the payload schema associated with each event. Choose [Download](#) to download the Async API specification JSON to your local machine.

8.5 View Service Descriptor

You can view the JSON file you provided during service instance creation.

Prerequisites

- You've completed the [initial setup \[page 40\]](#).
- You have the ReadRole assigned. See [Assigning Roles to Users \[page 51\]](#).

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose [Instances and Subscriptions](#) from the navigation pane on the left, then the [Event Mesh](#) tile.
3. Choose [Go to Application](#).
4. Open the required message client, then choose [Service Descriptor](#) to view the JSON file.
5. Choose [Download](#) to download the service descriptor JSON to your local machine.

Related Information

[Syntax for Service Descriptor \[page 32\]](#)

[Creating an Event Mesh Service Instance \[page 42\]](#)

8.6 View Event Catalog for a Subaccount

You can view the event catalog for a message client that gets created when you've integrated the service to receive cloud events from SAP S/4HANA Cloud.

Prerequisites

- You've completed the [initial setup \[page 40\]](#).
- You have the ReadRole assigned. See [Assigning Roles to Users \[page 51\]](#).
- You've integrated the service with SAP S/4HANA to receive cloud events, see [Enable the Consumption of SAP S/4HANA Cloud Events](#).

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose [Instances and Subscriptions](#) from the navigation pane on the left, then the [Event Mesh](#) tile.
3. Choose [Go to Application](#).
4. Choose [Events Exploration](#) and then, the required message client.

You can see the following information:

- List of events intended for the subaccount from different systems.
- Payload schema for each event.

8.7 Monitor Resources

You can view details on the utilization of messaging resources in your subaccount.

Prerequisites

- You've completed the [initial setup \[page 40\]](#).
- You have the ReadRole assigned. See [Assigning Roles to Users \[page 51\]](#).

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose [Instances and Subscriptions](#) from the navigation pane on the left, then the [Event Mesh](#) tile.
3. Choose [Go to Application](#).
4. Choose [Monitor](#).

You can see the following information based on the limits for your subaccount:

- Number of queues used.
- Number of connections used.
- Number of producers used.
- Number of consumers used.
- Number of spooled resources used.

Related Information

[Scope and Limitations \[page 14\]](#)

8.8 Test Publishing or Consuming Messages

You can publish or consume messages through a message client. The messages are published or consumed based on the connection rules that you've provided in the service descriptor.

Prerequisites

- You've completed the [initial setup \[page 40\]](#).
- You have the TestRole assigned. See [Assigning Roles to Users \[page 51\]](#).
- For publishing messages to a topic, you've created a queue subscription. See [Manage Queues \[page 60\]](#).

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose [Instances and Subscriptions](#) from the navigation pane on the left, then the [Event Mesh](#) tile.
3. Choose [Go to Application](#).
4. You can test messaging within a message client, or between two message clients based on the rules you've provided in the service descriptor. Use one of the following procedures based on your choice:
 - a. Open the required message client, then choose [Test](#) if you want to test messaging within a specific message client.
 - b. Select whether you want to publish messages to a queue, consume messages from a queue, or publish messages to a topic.
 - c. Based on your selection in the previous step, select the required queue or enter the topic name along with the namespace.
 - d. Choose [Publish Message](#) or [Consume Message](#) then, choose [OK](#).

Or

- a. Choose [Test](#) from the left pane if you want to test messaging between different message clients within your subaccount.
- b. Select whether you want to publish messages to a queue, or consume messages from a queue.
- c. Select the queue then, the required message client.
- d. Choose [Publish Message](#) or [Consume Message](#) then, choose [OK](#).
 - A confirmation is shown when you publish a message.
 - The message data and message properties are shown when you consume a message.
 - When a message in binary format is consumed, it's automatically downloaded to your local machine.

8.9 Edit Resource Units for a Subaccount

You can edit the number of resource units allocated to your subaccount.

Prerequisites

- You've completed the [initial setup \[page 40\]](#).
- You've the ManageRole assigned. See [Assigning Roles to Users \[page 51\]](#).

Context

The default resource unit allocation is 200 for newly created and existing subaccounts. The same allocation is applicable if you've a subaccount and you're using the service for the first time in the subaccount.

If necessary, you can adjust the number of resource units for your subaccount up to 400 resource units. When you adjust the number of resource units for your subaccount; ensure you update the resource units for the message client in the service descriptor accordingly. See:

- [Scope and Limitations](#)
- [Syntax for Service Descriptor](#)

If you want to increase the resource unit allocation beyond 400. You can request further allocation up to 1000 by [creating a support ticket](#) with your subaccount details and the number of resource units that you require.

Procedure

1. In the cockpit, navigate to your subaccount.
2. Choose [Instances and Subscriptions](#) from the navigation pane on the left, then the [Event Mesh](#) tile.
3. Choose [Go to Application](#).
4. Expand [Monitor](#) then, choose [Resource Units](#).

You can see the following information:

- Resource units allocated for the subaccount.
 - Resource units allocated for each message client in the subaccount and its limit.
5. Choose [Edit](#) to change the resource units allocation for the subaccount.
 6. Choose [Save](#).

Related Information

[SAP Event Mesh – Resource Units](#) 

9 Working with Event-Driven Integrations

Event-driven integrations allow you to use SAP-managed subscriptions for easy, error-free event handling between the SAP applications in your landscape. These integrations are configured for you by SAP. Once they're in place, you can enable or disable them from the Event Mesh user interface. If there are event errors, you can monitor them from the [Monitoring](#) page of the user interface.

[SAP-Managed Subscriptions \[page 70\]](#)

When registered SAP applications with event catalogs are part of your landscape, those applications can use Event Mesh to distribute events.

[SAP-Managed Subscription Monitoring \[page 71\]](#)

Find information on errors that occur when events that are part of SAP-managed subscriptions flow between SAP applications in your landscape.

[Enabling SAP Event Subscriptions \[page 72\]](#)

When you have SAP applications in your landscape, Event Mesh lets you move events between those applications without using an instance of the Event Mesh default plan. Before events flow between your SAP applications, you need to enable subscriptions in the Event Mesh user interface.

[Registering Event Mesh with SAP Cloud ALM \[page 74\]](#)

When you have SAP applications in your landscape, Event Mesh lets you move events between those SAP applications without using an instance of the Event Mesh default plan. If you want to track errors that occur with the events moving between SAP applications, you can register Event Mesh with SAP Cloud ALM.


9.1 SAP-Managed Subscriptions

When registered SAP applications with event catalogs are part of your landscape, those applications can use Event Mesh to distribute events.

There are several service plans in Event Mesh that support SAP-managed subscriptions. These service plans are only available internally for SAP applications. The configuration and setup of event handling between SAP applications is taken care of for you, and you control when the events flow between the applications.

There are a few important terms for you to understand within the context of SAP-managed subscriptions.

Events

Events that flow through Event Mesh between SAP applications must all conform to the [CloudEvents specification](#) .

SAP-Managed Subscriptions

SAP defines the supported set of subscriptions for SAP applications. A subscription defines the publishing application, the subscribing application, and the type of events that can flow between them.

You control when the SAP-managed subscriptions are enabled in your landscape from the [Subscriptions](#) page of the Event Mesh UI.

i Note

If you don't see subscriptions listed on the [Subscriptions](#) page of the Event Mesh user interface, then SAP-managed subscriptions aren't part of your landscape.

The [Subscriptions](#) page doesn't support other events that are part of the Event Mesh default plan.

Publisher

The publisher is the SAP application that is publishing the event.

Subscriber

The subscriber is the SAP application that is subscribing to the event.

Related Information

[Enabling SAP Event Subscriptions \[page 72\]](#)

[Registering Event Mesh with SAP Cloud ALM \[page 74\]](#)

9.2 SAP-Managed Subscription Monitoring

Find information on errors that occur when events that are part of SAP-managed subscriptions flow between SAP applications in your landscape.

Overview

When SAP applications are part of your application landscape, they can share event data using Event Mesh. You control whether events flow between your SAP applications from the [Subscriptions](#) page in the Event Mesh UI. When events are flowing, the [Monitoring](#) page allows you to connect with SAP Cloud ALM to monitor event errors.

An error occurs when an event fails to reach the subscribing application after it's published and pushed through Event Mesh. The error can occur when the event is forwarded through Event Mesh or after it's dispatched by Event Mesh to subscribers. The error information is sent to SAP Cloud ALM if you register it with Event Mesh. Error information includes:

- event ID
- event source
- event type
- event subject

- error code or error messages, if available

Use SAP Event Mesh to register SAP Cloud ALM. When the registration is complete, you see a [Go To Cloud ALM](#) link on the [Monitoring](#) page in SAP Event Mesh. Clicking the link takes you directly to SAP Cloud ALM. Once you navigate to SAP Cloud ALM, you can troubleshoot event errors.

i Note

The registration with SAP Cloud ALM allows you to see errors that occur with events that are part of SAP event-driven integrations. It doesn't track errors in events that are part of the Event Mesh default plan.

SAP Cloud ALM

As a Customer Integration Admin, you can log in to the SAP Cloud ALM and troubleshoot errors that occurred in Event Mesh during event processing. You can search for a specific error, review all errors that occurred during a defined period of time, and troubleshoot error situations. SAP Cloud ALM maintains Event Mesh error data according to their standard requirements. Consult their customer documentation to learn more about the length of time that error data is preserved.

Related Information

[Registering Event Mesh with SAP Cloud ALM \[page 74\]](#)

[Enabling SAP Event Subscriptions \[page 72\]](#)

[SAP Cloud ALM Documentation](#)

9.3 Enabling SAP Event Subscriptions

When you have SAP applications in your landscape, Event Mesh lets you move events between those applications without using an instance of the Event Mesh default plan. Before events flow between your SAP applications, you need to enable subscriptions in the Event Mesh user interface.

Prerequisites

- You subscribed to the Event Mesh Standard Application plan. For more information, see [Subscribing to the Event Mesh User Interface \[page 52\]](#).
- You have the role **Event Mesh Integration Administrator** assigned to your user in the BTP subaccount that has a subscription to Event Mesh, Standard Application plan.
- You logged in to the user interface and you see subscriptions listed in the table on the [Subscriptions](#) page.

Context

Subscriptions identify the publishing application, the subscribing application, and the type of events that can flow between them. When you enable a subscription, events begin to flow from the publishing SAP application through Event Mesh. The subscribing SAP application can retrieve the events at a convenient time.

i Note

If you logged in to the Event Mesh UI and you don't see any subscriptions listed in the table on the [Subscriptions](#) page, then this feature isn't supported by your plan.

Procedure

1. In the SAP Business Technology Platform Cockpit, navigate to your subaccount.
2. Select **Services** **Instances and Subscriptions**.
3. In the [Subscriptions](#) table, click the link next to Event Mesh.

The SAP Event Mesh application displays.

4. Log in with your credentials.
5. Select [Subscriptions](#).
6. From the table of subscriptions, select the tab for [All Available Subscriptions](#) and click a row that contains a disabled subscription.
7. Select the red status indicator to enable the subscription.

Results

The [Status](#) indicator turns green, which indicates that the subscription is enabled. Events are ready to flow through Event Mesh according to the subscription parameters.

Related Information

[SAP-Managed Subscriptions \[page 70\]](#)


[SAP-Managed Subscription Monitoring \[page 71\]](#)

[Registering Event Mesh with SAP Cloud ALM \[page 74\]](#)

9.4 Registering Event Mesh with SAP Cloud ALM

When you have SAP applications in your landscape, Event Mesh lets you move events between those SAP applications without using an instance of the Event Mesh default plan. If you want to track errors that occur with the events moving between SAP applications, you can register Event Mesh with SAP Cloud ALM.




Prerequisites

- You completed the onboarding activities for SAP Cloud ALM. Refer to SAP Cloud ALM documentation for onboarding support.
- You subscribed to the Event Mesh, Standard Application plan and have the **Event Mesh Integration Administrator** role collection assigned to your BTP subaccount user. For more information see, [User Roles for Event Mesh \[page 77\]](#).
- You have the SAP Cloud ALM service key. For more information, see [Retrieve SAP Cloud ALM Service Key](#) .

i Note

Event errors that occur when using the SAP Event Mesh, Default plan aren't passed to SAP Cloud ALM. The only errors that are tracked are the ones that occur when SAP applications send CloudEvents using the Event Mesh internal service plans. To learn more about Event Mesh plans, see [SAP Event Mesh Plans \[page 16\]](#).

Procedure

1. In the SAP BTP Cockpit, navigate to your subaccount.
2. Select  [Services](#)  [Instances and Subscriptions](#) .
3. In the [Subscriptions](#) table, click the link next to Event Mesh.

The SAP Event Mesh application displays.

4. Log in with your credentials.
5. From the [Subscriptions](#) page, check the table to confirm that you have SAP-managed subscriptions.

If no subscriptions are listed, then you don't have SAP-managed subscriptions. You don't have any events to monitor in SAP Cloud ALM. Registration isn't supported.

If subscriptions are listed, then proceed with the remaining steps.

6. Select the [Monitoring](#) page.
7. Click [Register SAP Cloud ALM](#).

The [SAP Cloud ALM Connection Details](#) window displays.

8. Add the connection details.

i Note

You can get the SAP Cloud ALM connection details from your SAP Cloud ALM administrator. Refer to the SAP Cloud ALM customer documentation for onboarding help.

9. Click [Register](#).

Results

A link is added to the [Monitoring](#) page that takes you directly to SAP Cloud ALM. Use the SAP Cloud ALM link to navigate to the SAP Cloud ALM application.

The text on the registration button changes to [Remove SAP Cloud ALM](#).

If you remove the connection between Event Mesh and SAP Cloud ALM, then event errors aren't sent to SAP Cloud ALM.

Related Information

[SAP Cloud ALM Documentation](#)

[SAP-Managed Subscriptions \[page 70\]](#)

[SAP-Managed Subscription Monitoring \[page 71\]](#)

[Enabling SAP Event Subscriptions \[page 72\]](#)

10 Security

SAP Event Mesh security information describes the policies, technologies, and controls that are applicable specifically to protect data, applications, and their associated infrastructure within SAP BTP.

Security aspects include the following:

- Technical system landscape
- User roles
- Authentication and authorization
- Transport encryption
- Data protection and privacy

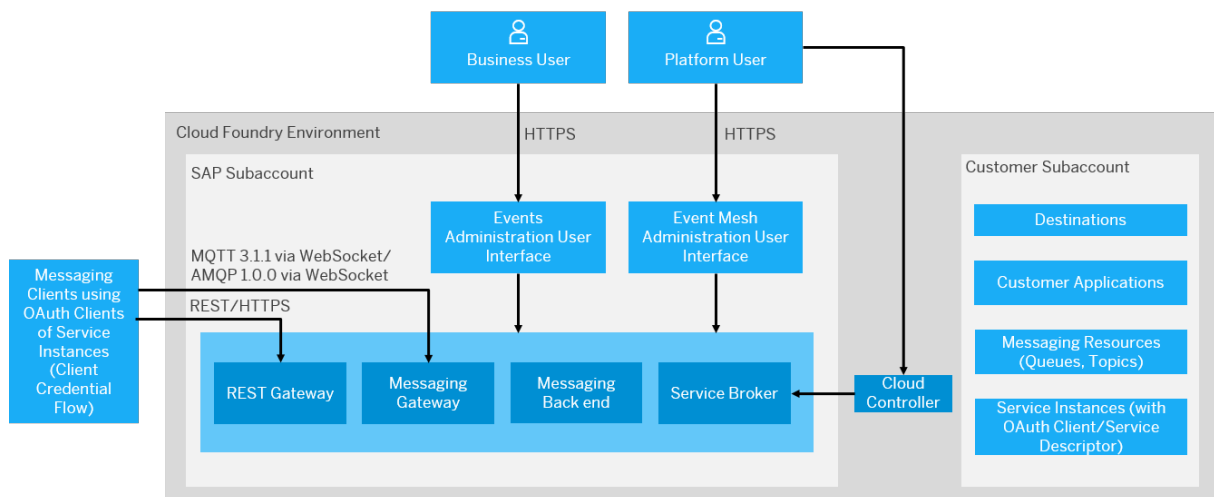
SAP Event Mesh runs on SAP BTP, for this reason, all security requirements for SAP BTP are also applicable to the service.

10.1 Technical System Landscape

Use SAP Event Mesh in the Cloud Foundry environment with a web browser.

Platform as a service scenario

Business users and platform users use a browser to configure the service. Platform users can also use cf cli for configuration. When a user creates a service instance, each service instance relates to an OAuth client that is created automatically with the service instance by the platform. These OAuth client credentials are part of the service instance's service key. Message clients use the OAuth client credential flow to obtain an OAuth token and authenticate at the Messaging Gateway/REST Gateway. Message clients can use MQTT 3.1.1 via WebSocket, AMQP 1.0.0 via WebSocket or REST via HTTP and communication is always encrypted (HTTPS). For more information, see [Security](#).




10.2 User Roles for Event Mesh

User authentication in SAP Event Mesh is provided by the authentication methods supported by SAP BTP, which use the Security Assertion Markup Language (SAML) 2.0 protocol for both authentication and single sign-on.

Types of BTP Users

This table explains how BTP users are authenticated and the types of actions they perform in the Event Mesh user interface.

BTP Users

User Type	Event Mesh Actions	Authentication	Authorization/Scope
Platform user	<ul style="list-style-type: none">Event Mesh administrationEvents administration and Event Mesh administration back-end	OAuth 2.0 (Authorization code grant flow)	Cloud controller runtime check For more information, see Checking User Permissions 
OAuth client (clone)	Messaging gateway handles technical communication with the service during binding.	OAuth (Client credentials flow)	Not applicable
Business user	<ul style="list-style-type: none">Event Mesh administrationEvents administration and Event Mesh administration back-end	OAuth (Authorization code flow)	<ul style="list-style-type: none">ManageRead

Event Mesh Roles

All Event Mesh roles and role collections are automatically available in your subaccount. The roles are tied to the Event Mesh Standard service plan, which is granted to your subaccount with any Event Mesh service plan entitlement.

When you assign an Event Mesh role or role collection to a user in the BTP cockpit, that user gains access to screens and actions in the Event Mesh user interface.

Role collections bundle related roles together to make it easy for you to assign privileges to users. Based on the service plan that you're using, choose the role collections that give your users the right access in the Event Mesh user interface.

Use the following table to map roles and roles collections with the service plan that you're using.

Roles	Purpose	Role Collection	Service Plan Supported
ManageRole	<ul style="list-style-type: none"> Create, edit, and delete queues and rules Create and edit service descriptors Create and manage webhook subscriptions, message clients, and event channel groups 	<ul style="list-style-type: none"> Enterprise Messaging Administrator Enterprise Messaging Subscription Administrator 	Default plan
<div> <div>i Note</div> <div>This role also has all view privileges associated with the ReadRole.</div> </div>			
TestRole	Test the configuration by sending and receiving messages.	Enterprise Messaging Developer	Default plan
ReadRole	<ul style="list-style-type: none"> View queues, rules, service descriptors, webhook subscriptions, message clients, and event channel groups Explore events 	<ul style="list-style-type: none"> Enterprise Messaging Developer Enterprise Messaging Display 	Default plan
SubscriptionManageRole	<ul style="list-style-type: none"> Manage SaaS subscriptions 	Enterprise Messaging Subscription Administrator	Default plan
<div> <div>i Note</div> <div>Use this role carefully. It should only be assigned to a user with administrative privileges.</div> </div>			
EventMeshSubscriptionsManageRole	View, manage, and monitor SAP-managed subscriptions from the Subscriptions menu in the Event Mesh user interface.	Event Mesh Integration Administrator	Internal service plans
EventMeshCloudALMRegistrationRole	Register Event Mesh with SAP Cloud ALM to monitor errors from SAP-managed events.	Event Mesh Integration Administrator	Internal service plans

Related Information

Security Administration: Managing Authentication and Authorization
SAP Event Mesh Plans [page 16]


[Assigning Roles to Users \[page 51\]](#)

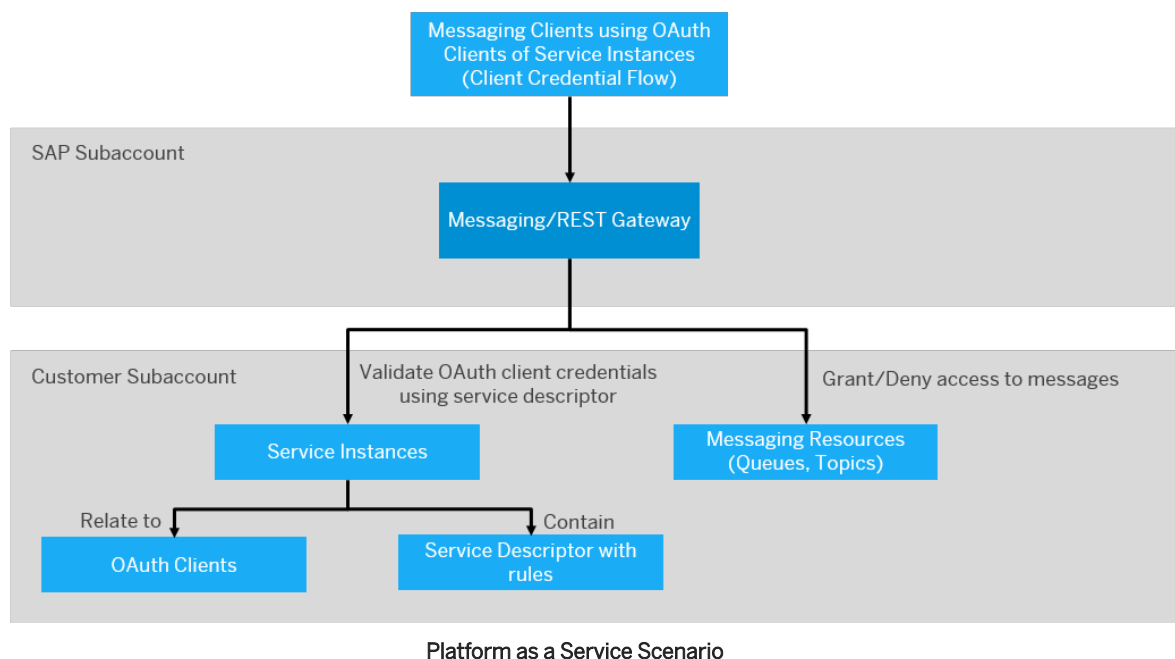
[Create User Groups, Role Collection and Assign Role Collection \[page 97\]](#)

10.3 Authentication and Authorization

Authorization in SAP Event Mesh determines access to applications. Administrators generally authorize users.

Assign authorizations to specify the actions users are allowed to perform.

Component	Description
Messaging Gateway	The messaging gateway is called by a consumer, for example SAP S/4HANA with an OAuth token of the OAuth client clone, which is created when a messaging service instance is created with a service broker. The messaging gateway doesn't perform any checks on whether a client is allowed to publish/subscribe to certain topics.
Event Mesh Administration	<p>The Dashboard Management component is called by a Cloud Foundry environment user according to the SSO dashboard flow in Cloud Foundry. For more information, see Checking User Permissions .</p> <p>The user interface initiates the OAuth login flow and forwards the OAuth token, which is issued based on the OAuth authorization code flow, to the Event Mesh administration and events administration back end. The back-end component then checks the user permissions, for example, whether the user has read/manage permissions.</p>
Events Administration	The events administration component is called by a Business User or an SAP Event Mesh Integration Administrator. The Events Administration user interface initiates the OAuth login flow and forwards the OAuth token, which is issued based on the OAuth authorization code flow, to the Event Mesh administration and events administration back end. The back-end component then checks whether the token has the appropriate OAuth scopes.
Management	When a user subscribes to the default plan, the service instance is created with a service descriptor that contains a set of access rules. These access rules grant access to messaging resources such as queues, topics, subscribe or publish. The messaging gateway authenticates messaging clients using the OAuth clients of the service instances. The OAuth client determines the related service instance. Also, the messaging gateway uses the service descriptor to grant or deny access to messaging resources.



10.4 Transport Encryption

All outside communication channels (WebSocket calls to messaging gateway) are encrypted via TLS (Transport Layer Security).

10.5 Data Protection and Privacy

Data protection is associated with numerous legal requirements and privacy concerns. In addition to compliance with general data protection and privacy acts, it's necessary to consider compliance with industry-specific legislation in different countries.

SAP provides specific features and functions to support compliance with regard to relevant legal requirements, including data protection. SAP doesn't give any advice on whether these features and functions are the best method to support company, industry, regional, or country-specific requirements. Furthermore, this information must not be taken as advice or a recommendation regarding additional features that would be required in specific IT environments. Decisions related to data protection must be made on a case-by-case basis, considering the given system landscape and the applicable legal requirements.

SAP Event Mesh doesn't store any private data. As a plain infrastructure component the service offers transport and storage of messages. Message content and storage duration are the customer's responsibility. Messages are stored until they're received by customer applications. The service doesn't access the content of the messages. It provides access control on a service instance level. Access to single messages or messages with certain properties isn't controlled explicitly.

In case, messages are being sent that contain private or personal data, it's recommended to:

- Enable only trusted applications to access the messages in the corresponding Event Mesh service instances.

- Ensure that proper access control is implemented within the applications.
- Ensure that messages aren't stored longer than necessary in queues, for example, by consuming them instantly or by specifying a "time to live" per message (It's supported with AMQP 1.0)
- The service always encrypts events, messages, and configuration such as queue names during transfer over networks and when storing on persistencies.
- The service doesn't use encryption keys that are specific to you or your service instances.

Note

SAP doesn't provide legal advice in any form. SAP software supports data protection compliance by providing security features and specific data protection-relevant functions. In many cases, compliance with applicable data protection and privacy laws aren't covered by a product feature. Definitions and other terms used in this document aren't taken from a particular legal source.

⚠ Caution

The extent to which data protection is supported by technical means depends on secure system operation. Network security, security note implementation, adequate logging of system changes, and appropriate usage of the system are the basic technical requirements for compliance with data privacy legislation and other legislation.

Event Mesh only holds event data for a limited period of time, for example, 7 days, and doesn't permanently store any private data. As a plain infrastructure component, the service offers event distribution. The service doesn't access the event payload. It provides access control on a service-instance level. Access to single events or events with certain properties isn't controlled explicitly. The only exception is the ability to set up filters based on event header attributes for DPP and routing.

If events are sent that contain private or personal data, we recommend the following guidelines.

- Enable only trusted applications to access the events in the corresponding Event Mesh service instances.
- Ensure that proper access control is implemented within the application.

The Customer Integration Administrator is responsible for validating the setup, understanding the business context, and enabling subscription content, which includes filter definitions that are defined by the customer's Data Privacy Officer (DPO). Customers only see subscription content that matches publisher and subscriber applications in their zone.

Events that are part of a subscription that supports SAP-to-SAP integrations could contain personal data. These events must be reviewed by the Customer Integration Administrator who is responsible for validating business requirements. It's the Integration Administrator's responsibility to ensure that all legal obligations with respect to event data are fulfilled before enabling subscriptions.

Related Information

[Security](#)

10.6 Auditing and Logging Information

Find a list of the security events that are logged by SAP Event Mesh.

Security Events Written in Audit Logs

Event Grouping	What Events are Logged	How to Identify Related Log Events	Additional Information
Cloud ALM Authorization	Registration attempts in Event Mesh UI when connecting with SAP Cloud ALM.	em-calm-oauth-credentials-saved: Tracks when credentials are saved.	
		em-calm-oauth-credentials-deleted: Tracks when credentials are deleted.	
Configuration	Creating, updating, or deleting service instances	em-service-instance-created: Tracks a new service instance creation.	
		em-service-instance-deleted: Tracks a service instance deletion.	
	Creating or deleting subscriptions	em-service-binding-created: Tracks a binding when it is created for a service instance.	
		em-service-binding-deleted: Tracks a binding when it is deleted for a service instance.	
		em-zone-subscribed: Tracks a zone subscription to the service.	
		em-zone-unsubscribed: Tracks when a zone unsubscribes the service.	

Event Grouping	What Events are Logged	How to Identify Related Log Events	Additional Information
	Creating or deleting subscriptions using Management APIs	<p>Subscription created: Tracks subscription creation using the Management API.</p> <p>Subscription deleted: Tracks subscription deletion using the Management API.</p> <p>Subscription Content: Tracks subscription content creation using the Management API.</p> <p>MT App produce block: Tracks blocking of multi-tenant application creation using Management API.</p>	Configuration changes include changes to custom data.

Related Information

[Audit Logging in the Cloud Foundry Environment](#)

11 Monitoring and Troubleshooting

If you encounter an issue with this service, we recommend that you do the following:

Announcements and Subscriptions

You can follow the availability of SAP BTP at [SAP Trust Center](#). See,

- the availability by service on the [SAP Integration Suite](#) tile of the [Cloud Status](#) tab page;
- the availability by region on the [Data Center](#) tab page.

To get notifications for updates and downtimes, subscribe at the [Cloud System Notification Subscriptions](#) application. Create a subscription by specifying Cloud Product, Cloud Service, and Notification Type. For more information, see [Cloud System Notification Subscriptions User Guide](#).

Check Guided Answers

In the SAP Support Portal, check the [Guided Answers for Event Mesh](#). You can find solutions for general issues related to the service there.

Contact SAP Support

You can report an incident or error through the [SAP Support Portal](#).

Use the following component for your incident:

Component Name	Component Description
BC-CP-EM-MES	Event Mesh on Cloud Foundry

When submitting the incident, we recommend including the following information:

- Region information (Canary, EU10, US10)
- Subaccount technical name
- The URL of the page where the incident or error occurs
- The steps or clicks used to replicate the error
- Screenshots, videos, or the code entered

12 Glossary

This section contains commonly used terms in the Event Mesh service.

Term	Definition
Message Client	A message client has unique credentials that allow it to perform messaging using the service. Each message client has a name, namespace, and a set of rules that define the permissions for publishing messages and subscribing to it. A message client can also provide an event catalog.
Webhook	A webhook is an event notification sent through HTTP.
Webhook Subscription	A webhook subscription allows your system to receive events from a webhook.
Queue	A queue is a space used to host a message until it's received by the subscriber.
Topic	A topic is a space used to host a message for a specified period of time defined by the sender.
Queue Subscription	A queue subscription allows a topic to subscribe to a queue for its message retention capability.
SAP-Managed Subscription	SAP defines the supported set of subscriptions used with Event Mesh internal service plans. These subscriptions define the publishing SAP application, the subscribing SAP application, and the type of events that can flow between them.
Event	An event notifies a consumer that an object has changed. Events that flow through Event Mesh between SAP applications must all conform to the CloudEvents specification .
Event Catalog	The list of events that are published to a message client or consumed by it.
Service Descriptor	During service instance creation, a message client is defined through a name, namespace, and a set of rules using a JSON file. This JSON file is known as a service descriptor.
Publisher	A publisher is the system or application that is the source of the event.
Subscriber	A subscriber is the system or application that receives the event.
messagingrest	messagingrest denotes APIs used to enable messaging using REST.
management	management denotes APIs used to enable managing queues and queue subscriptions using REST.
emname	emname denotes the name of the message client.
Event Channel Group	The logical grouping of events with a unique name using the service.
QoS	Quality of Service denotes the way the messaging protocol handles retransmission of the message and ensures its delivery regardless of network reliability.
TTL	The Time-To-Live (TTL) value (in milliseconds) can be set for each guaranteed message published by a message producer. Endpoints can be configured to respect message TTLs through the REST APIs for management or the user interface. If message TTLs aren't honored, the message is discarded by the endpoint.

13 Event Mesh Lite Service Plan (Deprecated)

The Event Mesh lite service plan is being discontinued. It's recommended that you discontinue using service instances based on the lite service plan and create new service instances based on the default service plan. By migrating to the default service plan, you benefit from an optimized messaging infrastructure, which fully supports scalable and versatile API or event-based business scenarios.

[Concepts \(Deprecated Lite Service Plan\) \[page 86\]](#)

SAP Event Mesh employs a centralized message-oriented architecture. It's more scalable and reliable compared to the traditional point-to-point communication model.

[Initial Setup \(Deprecated Lite Service Plan\) \[page 89\]](#)

[Using Event Mesh \(Deprecated Lite Service Plan\) \[page 99\]](#)

You can access different user interfaces belonging to the lite plan for administration and end-user tasks.

13.1 Concepts (Deprecated Lite Service Plan)

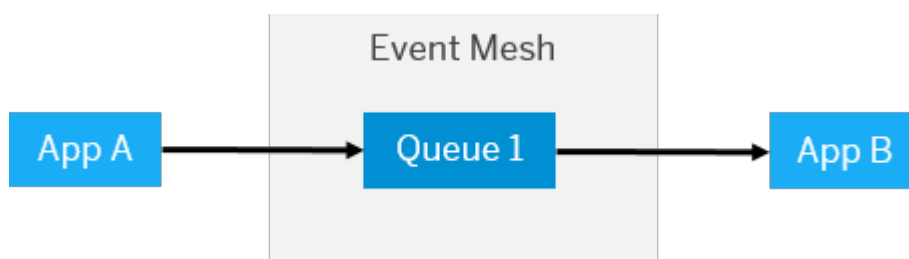
SAP Event Mesh employs a centralized message-oriented architecture. It's more scalable and reliable compared to the traditional point-to-point communication model.

The traditional point-to-point communication model is a decentralized one where applications and services directly communicate with each other. The service decouples communication between the sending and receiving applications and ensures the delivery of messages and events between them.

The lite service plan supports the following messaging and event-based concepts:

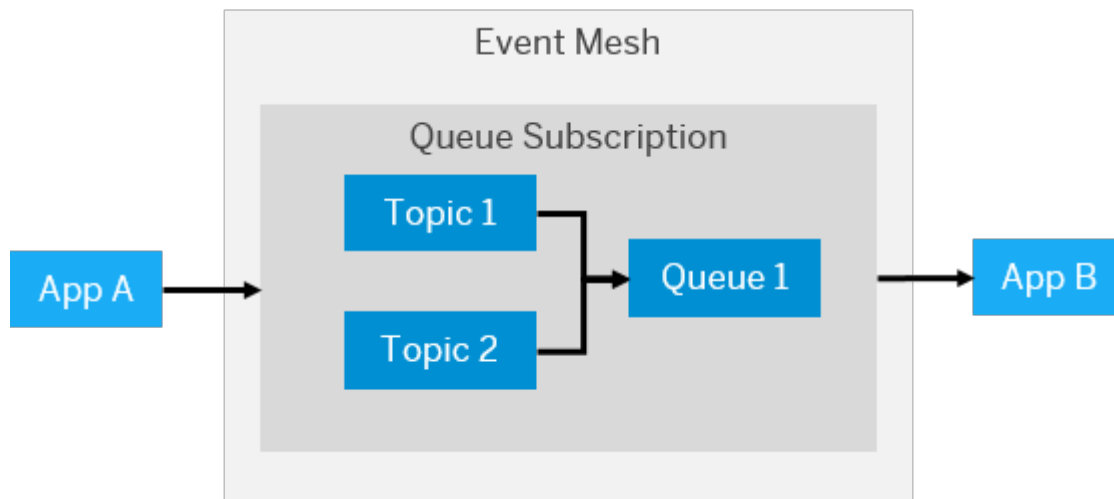
Queues

The service enables applications to communicate with each other through message queues. A sending application sends a message to a specific named queue. There's a one on one correspondence between a receiving application and its queue. The message queue retains the messages until the receiving application consumes it. You can manage these queues using the service.



Queue Subscriptions

The service enables a sending application to publish messages and events to a topic. Topics don't retain messages but, it can be used when each message needs to be consumed by a number of receiving applications. Topics must be managed through queue subscriptions. In queue subscriptions, the service enables a sending application to publish messages to a topic that directly sends the message to the queue to which it's bound. For example, events from an SAP S/4HANA system (event source) can only be sent to a topic. A queue subscription ensures that the message is retained until it's consumed by the receiving application.



Events

An event is a message that is sent to notify a consumer that an object has changed. An event source is the system or application from which the event originates. A receiving application needs to create a connection to the event source to facilitate the flow of events. The service can receive events from an event source, lookup events, and discover events. Different event sources can publish events to the service.

Related Information

[Initial Setup \(Deprecated Lite Service Plan\) \[page 89\]](#)

[Using Event Mesh \(Deprecated Lite Service Plan\) \[page 99\]](#)


13.1.1 Messaging Protocols and Libraries

The service supports open standard messaging protocols and allows you to use client libraries for Java and Node.js.


Protocols

The Event Mesh service supports the following messaging protocols:

Advanced Message Queuing Protocol (AMQP) 1.0 over WebSocket

It's an open standard protocol used for messaging between applications or organizations. We recommend that you use AMQP 1.0 over WebSocket for messaging between applications running on Cloud Foundry. For more information, see [Specification for AMQP 1.0 over WebSocket](#) .

Message Queuing Telemetry Transport (MQTT) 3.1.1 over WebSocket

It's a lightweight messaging protocol designed specifically for constrained devices, low bandwidth, high latency, or unreliable devices. We recommend that you use MQTT 3.1.1 over WebSocket for messaging to a service from applications not running on the Cloud, for example, SAP S/4HANA. For more information, see [Specification for MQTT 3.1.1 over WebSocket](#) .

i Note

Quality of Service (QoS) is a feature of MQTT, where the protocol handles retransmission and guarantees that the message is delivered regardless of network reliability. MQTT 3.1.1 over WebSocket supports only QoS 0, QoS 1, and Messaging Gateway. The supported QoS levels are:

- At most once (0) - It guarantees its best effort with delivery. A message isn't acknowledged by the receiver, stored or redelivered by the sender.
- At least once (1) - It guarantees that a message is delivered at least once to the receiver. The message can also be delivered more than once.

REST APIs for Messaging

The service provides REST APIs for messaging. You can use these messaging REST APIs to send and receive messages.

i Note

Quality of Service (QoS) 0 and 1 are supported.

For more information, see [REST APIs for Messaging \[page 23\]](#).

i Note

The order of delivery isn't guaranteed for the messaging protocols supported by the service.

Libraries

The Event Mesh service allows you to use the following libraries:

AMQP 1.0 over WebSocket libraries and MQTT 3.1.1 over WebSocket libraries for Node.js

It enables Node.js developers to connect and use the service. We recommend that you use these libraries to develop your application as doing so significantly reduces development effort. For more information, see [Node.js libraries](#).

Protocol Agnostic Libraries

The service supports protocol agnostic libraries that can be used at the application configuration level for Java and Node.js. It allows you to work with messaging applications without getting into the intricacies of a

messaging protocol. You can create an application configuration that defines an input (source from which the application receives messages) and output (destination to which the application sends messages). The application configuration is typically a JSON file with properties such as input and output configuration, quality of service and so on. Once you create it, you can use it in an environment variable to start the application. Use protocol agnostic libraries in the application code to create stream objects that define the source, destination, quality of service and other technical properties from the application configuration JSON file. The advantages are as follows:

- The messages can be made available through stream objects. The application isn't dependent on a messaging protocol for relaying messages.
- The application configuration resides outside the application code. You can use the same application code and change only the application configuration for different landscapes. For example, when you move the application from development to production landscape, the properties in the application configuration change. In this scenario, only the application configuration has to be changed and the application code remains the same.

Apache QPID JMS Client Library

It enables Java developers to connect and use the service. We recommend that you use these libraries to develop your application as doing so significantly reduces development effort, see [Java Libraries](#) .

i Note

The following limits are applicable when you use the lite service plan:

- MQTT 3.1.1 over WebSocket can be used only for sending messages to topics, not directly to queues. Queue subscriptions aren't supported.
- Cloud Foundry applications can only receive messages from queues, not topics, when they use AMQP 1.0 over WebSocket.
- For all the three supported messaging protocols:
 - The maximum message size is 1 MB. If messages are above 1 MB, the connection is closed. It applies to applications running on Cloud Foundry and other platforms.
 - The maximum size of all the queues for a service instance at a time is 12 GB.
 - The maximum number of all the queues for a service instance is 500.
 - The message rate for a service instance must be limited to 500 KB per second.
 - The maximum number of connections for a service instance is 50.

13.2 Initial Setup (Deprecated Lite Service Plan)

Trial Scope

→ Remember

The trial availability for Event Mesh dev plan with standard (subscription) is being discontinued. It's recommended that you discontinue using trial service instances based on the dev service plan with standard (subscription). As an alternative, you can use the default service plan that provides an optimized messaging infrastructure, which fully supports scalable and versatile API or event-based business scenarios.

Event Mesh is available on trial. A trial account lets you try out SAP BTP for free and is open to everyone. Trial accounts are intended for personal exploration, and not for productive use or team development. They allow restricted use of the platform resources and services. The trial period varies depending on the environment.

To activate your trial account, go to [Welcome to SAP Cloud Platform Trial](#). The dev service plan with standard (subscription) on the trial landscape has a subset of the features available with the lite plan.

See [Trial Accounts](#).

Initial Setup

To start using the service, you need to perform the following tasks in the SAP BTP cockpit.

1. [Set up a Subaccount \[page 90\]](#)
2. [Create an Event Mesh Service Instance \[page 91\]](#) or [Create an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface \[page 93\]](#)
3. [Binding an Application to an Event Mesh Service Instance \[page 46\]](#)
4. [Subscribe to the Event Mesh Business Application \[page 96\]](#)
5. [Create User Groups, Role Collection and Assign Role Collection \[page 97\]](#)
6. [Integrating the Service with SAP S/4HANA \(Deprecated Lite Service Plan\) \[page 97\]](#)

13.2.1 Set up a Subaccount

After you purchase an Event Mesh lite plan, you must perform the following steps in the SAP BTP cockpit.

Prerequisites

- You have a customer account with SAP BTP and an Event Mesh lite service plan. See [Getting Started with a Customer Account: Workflow in the Cloud Foundry Environment](#).
- You have a global account. See [Getting a Global Account](#).
- You've created a subaccount. See [Create a Subaccount in the Cloud Foundry Environment](#).
- You've created a space within the subaccount in which Cloud Foundry is enabled. See [Managing Orgs and Spaces Using the Cockpit](#).

Context

The Event Mesh lite service plan:

- Couples a service instance to a broker.
- Provides exclusive access for one tenant that hosts the service instance.

Procedure

1. In the SAP BTP cockpit, navigate to your global account and assign the entitlement for the lite service plan for Event Mesh to your subaccount, see [Configure Entitlements and Quotas for Subaccounts](#).
2. Create a service instance for Event Mesh using either the cockpit or cf cli, see
 - [Create an Event Mesh Service Instance](#) [page 91].
 - [Create an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface](#) [page 93].

→ Remember

The lite service plan for SAP Event Mesh is being discontinued. It's recommended that you discontinue using service instances based on the lite service plan and create new service instances based on the default service plan. By migrating to the default service plan, you benefit from an optimized messaging infrastructure, which fully supports scalable and versatile API or event-based extension scenarios, see [Onboarding Guide](#).

3. [Binding an Application to an Event Mesh Service Instance](#) [page 46].
4. To access the events administration user interface, you must [Subscribe to the Event Mesh Business Application](#) [page 96].
5. [Create User Groups, Role Collection and Assign Role Collection](#) [page 97].
6. If you want to receive events from an SAP S/4HANA system, see [Integrating the Service with SAP S/4HANA \(Deprecated Lite Service Plan\)](#) [page 97].

13.2.2 Create an Event Mesh Service Instance

Use the SAP BTP cockpit to create a service instance for lite plan.

Prerequisites

Access to a Cloud Foundry space.

Context

To use the service, you need to create a service instance in the cockpit.

Procedure

1. Open the cockpit.

2. Navigate to [Spaces](#) in your Cloud Foundry environment and choose ► [Services](#) ► [Service Marketplace](#) ► [Event Mesh Service](#) .
3. Choose ► [Instances](#) ► [New Instance](#) .
4. Select the lite service plan and choose [Next](#).
5. Specify parameters using a JSON file.

For the lite plan:

The `emname` parameter is mandatory. "emname" is unique for a subaccount. It's recommended that the same name is used for emname and instance name as the emname represents a service instance in Cloud Foundry.

(Optional) To use REST APIs for management the `management` parameter must be set to true. Similarly, to use the REST APIs for messaging the `messagingrest` parameter must be set to true.

≡ Sample Code

```
{
  "emname": "<your-emname>",
  "options": {
    "management": true,
    "messagingrest": true
  }
}
```

i Note

For existing service instances, update your service instance to include the parameters in your JSON file. See, [Create an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface \[page 93\]](#).

6. Enter the instance name and choose [Finish](#).
7. To configure endpoints to an SAP S/4HANA system as an event source:
 - a. Open the service instance, then choose ► [Service Keys](#) ► [Create Service Key](#) .
 - b. Make a note of the client ID, client secret, token endpoint, and base URL, then use these parameters to create a communication arrangement. For more information, see [Create Communication Arrangement](#).

i Note

In the "messaging" section of the service key under the protocol "mqtt311ws", the client ID, client secret, and token endpoint are present under "oa2" and the base URL is under "uri". For the communication arrangement, provide the base URL without the path as the host name and `protocols/mqtt311ws` as the path for outbound services.

8. To delete a messaging service instance, choose [Delete](#) under Actions.

13.2.3 Create an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface

Create an Event Mesh service instance using the Cloud Foundry Command Line Interface (cf CLI) for the lite plan.

Prerequisites

Access to a Cloud Foundry space.

Procedure

1. Log on to the Cloud Foundry Environment using the Command Line Interface. For more information, see [Log On to the Cloud Foundry Environment Using the Command Line Interface](#).
2. Select your org, then the space.
3. Enter `cf marketplace` to verify the availability of the Event Mesh service in the Cloud Foundry marketplace.
4. Access an Event Mesh service instance using one of the following methods:
 - `cf services` to find existing instances of the service in your space. Update an existing service instance, enter If the application is deployed in the same space, you can use an existing service instance. Enter `cf update-service <service-instance-name> -c 'service-descriptor.json'`

Sample Code

If the application is deployed in the same space, you can use an existing service

```
cf update-service messaging-products -c
```

```
{
  "emname": "messaging-products",
  "options": {
    "management": true,
    "messagingrest": true
  }
}
```

- To create a new service instance using the lite plan, enter `cf create-service enterprise-messaging lite -c '{"emname":"<service name>" "options": {"<type>": <value>,"<type>": <value>}}' <instance name>`.

Sample Code

```
cf create-service enterprise-messaging lite -c
{
  "emname": "messaging-products",
  "options": {
    "management": true,
    "messagingrest": true
  }
}
```

```
} "
messaging-products
```

13.2.4 Bind an Application to an Event Mesh Service Instance

Bind your messaging application to your Event Mesh service instance for the lite service plan.

Prerequisites

- Deploy a messaging application in the same space as your service instance, see [Deploy Business Applications in the Cloud Foundry Environment](#).
- You've created an Event Mesh service instance.

Context

When you bind an application to the service, the application receives a JSON file with the binding configuration. The binding configuration contains specifications to the different messaging protocols that can be used to connect to the service. OAuth is used to connect to the messaging protocol endpoints. Before establishing a connection, the application must generate a token using the client ID, client secret, token endpoint, and grant type in the oa2 section of the binding configuration. The protocol section contains the URL with the host name to which the applications must connect to use respective client libraries.

The following code shows the structure of a service binding for Event Mesh:

```
{
  "xsappname": "<app-name>",
  "serviceinstanceid": "<instance-id>",
  "messaging": [
    {
      "oa2": {
        "clientid":
        "<client_id>",
        "clientsecret":
        "<client_secret>",
        "tokenendpoint":
        "https://<app-url>/oauth/token",
        "granttype":
        "client_credentials"
      },
      "protocol": [
        "amqp10ws"
      ],
      "broker": {
        "type":
        "messaginggateway"
      },
      "uri": "wss://<app-url>/
protocols/amqp10ws"
    },
  ]
}
```

```

"client_id",
"client_secret",
"https://<app-url>/oauth/token",
"client_credentials"

"messaginggateway"

protocols/protocols/mqtt311ws"
},
{
"oa2": {
"clientid":
"clientsecret":
"tokenendpoint":
"granttype":
},
"protocol": [
"mqtt311ws"
],
"broker": {
"type":
},
"uri": "wss://<app-url>/

"oa2": {
"clientid":
"clientsecret":
"tokenendpoint":
"granttype":
},
"protocol": [
"httprest"
],
"broker": {
"type":
},
"uri": " https://<app-url>/

"oa2": {
"clientid":
"clientsecret":
"tokenendpoint":
"granttype":
},
"uri": " https://<app-url>/

"management": [
{
"client_id",
"client_secret",
"https://<app-url>/oauth/token",
"client_credentials"

"saprestmgw"

}
],
"management": [
{
"client_id",
"client_secret",
"https://<app-url>/oauth/token",
"client_credentials"

}
]
}

```

Note

- The segment `management` in the service binding information is available only if you have set the option `management` as `true` during service instance creation.
- The segment `messaging` in the service binding information is available only if you have set the option `messagingrest` as `true` during service instance creation.

Procedure

1. Navigate to the space in which your deployed application and service instance exists.
2. Select one of the following methods to proceed:

View	Steps
Application	<ol style="list-style-type: none">1. In the navigation pane, select Applications, then the application you've deployed.2. In the navigation pane, choose Service Bindings.3. Choose Bind Service.4. Select Service from the catalog, then Event Mesh.5. Select the lite service plan.6. (Optional) Specify parameters in a JSON format or browse and upload a JSON file.7. Provide the name of the service instance you've created.8. Choose Finish.
Service instance	<ol style="list-style-type: none">1. Open the service instance that you've created.2. Choose Bind Instance.3. Select the application you've deployed.4. (Optional) Specify parameters in a JSON format or browse and upload a JSON file.5. Save your changes.

You can also use cf CLI to bind an application to your service instance using the following command:

```
cf bind-service APP-NAME SERVICE_INSTANCE {-c PARAMETERS_AS_JSON}
```

Related Information

[Bind Service Instances to Applications Using the Cockpit](#)

[Bind Service Instances to Applications Using the Cloud Foundry Command Line Interface](#)

13.2.5 Subscribe to the Event Mesh Business Application

Subscribe to SAP Event Mesh to access the Events Administration user interface.

The Events Administration user interface for the lite plan is provided as multitenant business application. Select the [Event Mesh](#) tile to subscribe to the application. For information on how to subscribe to a business application using the cockpit, see [Subscribe to Multitenant Business Applications in the Cloud Foundry Environment Using the Cockpit](#).

Note

Subscriptions can be set up only by administrators of the global account.

13.2.6 Create User Groups, Role Collection and Assign Role Collection

The service provides standard roles for typical user profiles. You can create specific user groups and role collection for your business users and assign these standard roles to the user groups.

The roles that can be assigned include the following:

- ReadRole (developer) - View event channel groups and explore events.
- ManageRole (administrator) - Create, edit, and delete event channel groups along with the ReadRole.

Perform the following tasks:

1. [Create a new user group.](#)
2. [Working with Role Collections](#)
3. [Directly Assign Role Collections to Users](#)

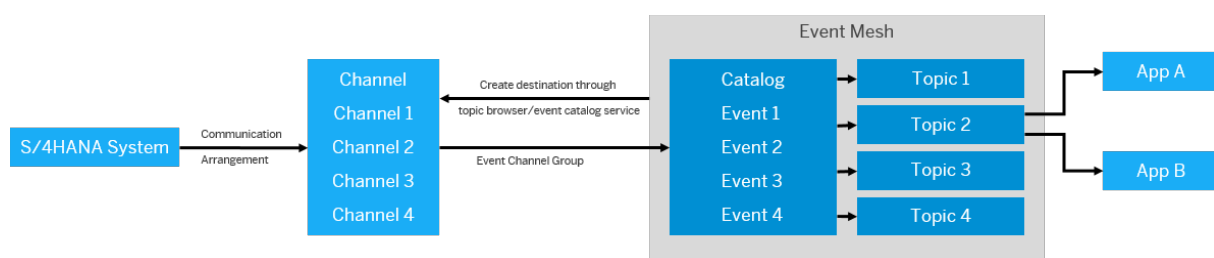
Note

You can set up an Identity Provider to enable IdP-initiated single sign-on (SSO) from all configured corporate identity providers (IdPs), see [Enable IdP-Initiated SSO from All Corporate Identity Providers](#). If you don't configure an identity provider, you have to assign role collections to individual platform users, it can't be done as a user group.

13.2.7 Integrating the Service with SAP S/4HANA (Deprecated Lite Service Plan)

After you create a service instance, you can share the instance credentials to the event administrator. The event source such as an SAP S/4HANA system uses the credentials to create a communication arrangement. The communication arrangement allows the inflow of events to the service.

In the SAP S/4HANA system, related events can be logically grouped by an event administrator. This logical grouping of events with a unique name at the event source is known as event channel group. This group is then associated with a topic using the service that allows receiving applications to subscribe to a topic to access events from the SAP S/4HANA system. Next, you can create a destination to see which channels are sending the events and also, the number of channels available for sending events to the service. One subaccount can create a destination to one SAP S/4HANA system and through the topic browser service, the events intended for the subaccount are sent. All the SAP S/4HANA systems for that subaccount as listed as event sources in the user interface. When a communication arrangement is created, channels get enabled and allow the events to be sent to the service. The event administrator can control which events get pushed to the service, for example, filter the events that are sent. The following diagram illustrates this process:



Related Information

[Enterprise Event Enablement](#)

[Configure Destinations to an SAP S/4HANA System \[page 98\]](#)

13.2.7.1 Configure Destinations to an SAP S/4HANA System

Configure destinations to communicate with an event source such as SAP S/4HANA.

Prerequisites

- You've created a communication arrangement with SAP S/4HANA, see [Communication Management](#).
- You've logged into the cockpit and opened the Destinations editor, see [Configure Destinations from the Cockpit](#).
- (Optional) If the event source isn't on the cloud, you have set up a connectivity service using [cloud connector](#) to configure on-premise systems that are exposed to SAP BTP.

Procedure

1. Choose [New Destination](#).
2. From the [Type](#) dropdown menu, choose HTTP.
3. Specify the destination URL, that you've obtained from the communication arrangement.

i Note

The URL must be in the format `https://<hostname>/sap/opu/odata/IWXBE/BROWSER_SRV`. BROWSER_SRV is used to explore events.

4. From the [Authentication](#) dropdown box, select Basic.
5. Provide the username and password, that you've obtained from the communication arrangement.
6. In the [Additional Properties](#) panel, choose [New Property](#) and provide "EnterpriseMessaging=true".
7. Choose [Save](#).

Cloud applications can consume events without creating a destination in SAP BTP. But, you must create a destination that points to an OData service for discovering events from an SAP S/4HANA system. An event channel group allows you to view the events at the event source while exploring events.

Related Information

[Create HTTP Destinations](#)

[Edit and Delete Destinations \(Cockpit\)](#)

13.3 Using Event Mesh (Deprecated Lite Service Plan)

You can access different user interfaces belonging to the lite plan for administration and end-user tasks.

- **Messaging Administration:** You can use this interface to manage queues, queue subscriptions, and webhook subscriptions.
- **Events Administration:** You can use this interface to group related events from an event source such that developers can look it up with ease.

Related Information

[Messaging Administration \[page 99\]](#)

[Events Administration \[page 106\]](#)

13.3.1 Messaging Administration

The messaging administration user interface allows you to manage queues, queue subscriptions and create application configurations for different messaging scenarios.

You can access this user interface through the dashboard for messaging administration. Choose [Open Dashboard](#).

Related Information

[Manage Queues \[page 100\]](#)

[Manage Queue Subscriptions \[page 100\]](#)

[Generate Application Configurations \[page 103\]](#)

[Check Application Configuration \[page 106\]](#)

13.3.1.1 Manage Queues

Queues enable point-to-point communication between two applications. An application can subscribe to a queue.

Context

Queue names must follow the naming convention specified in [Syntax for Naming Queues, Topics, and Topic Patterns \[page 30\]](#).

Procedure

1. Navigate to your Cloud Foundry space and select your Event Mesh service instance.
2. In the left pane, select [Service Instances](#).
3. Choose [Open Dashboard](#) under the list of [Actions](#) that correspond to your Event Mesh service instance.
4. In the messaging administration page, choose [Queues](#).

You can see the following information:

- List of queues.
 - Number of messages stored in each queue.
 - Size (in kilobytes) of all the messages stored in each queue.
5. To create a new queue:
 - a. Choose [Create](#).
 - b. Enter a queue name.
 - c. Choose [Create](#).
 6. Under Actions, choose [Delete Queue](#) that corresponds to the queue you want to delete.

Caution

Deleting a queue also deletes any associated queue subscriptions and messages (if any).

13.3.1.2 Manage Queue Subscriptions

Create a queue subscription when you want to retain messages that are sent to a topic.

Context

Topic and topic pattern names must follow the naming convention specified in [Syntax for Naming Queues, Topics, and Topic Patterns \[page 30\]](#).

Procedure

1. Navigate to your Cloud Foundry space and select your Event Mesh service instance.
2. In the left pane, select [Service Instances](#).
3. Choose [Open Dashboard](#) under the list of [Actions](#) that correspond to your Event Mesh service instance.
4. In the messaging administration page, choose [Queue Subscriptions](#).

You can see the following information:

- List of queues.
 - List of queue subscriptions.
5. To create a new queue subscription:
 - a. Choose [Create](#).
 - b. Select a queue.

i Note

The list of queues in your service instance is populated.

- c. Enter a topic name or topic pattern.
 - d. Choose [Create](#).
6. Under Actions, choose [Delete Queue](#) that corresponds to the queue you want to delete.

13.3.1.3 Manage Webhook Subscriptions

You can use a webhook to subscribe to a queue. It allows the service to push messages received on the queue to the webhook.

Prerequisites

You've provided `"messagingrest": true` while creating a service instance. See [Create an Event Mesh Service Instance \[page 91\]](#).

Context

The webhook subscription contains a subscription name, queue name, details of the webhook URL, subscription status, handshake status, and quality of service. The subscription status can be active or handshake pending. The handshake gives permission to the service to send messages to the webhook. The handshake status can have the following values:

- Not Initiated: The service hasn't requested a handshake.

- Requested: The handshake has been requested and the webhook responded with a response code 2xx but, the response header WebHook-Allowed-Origin doesn't match with the request header WebHook-Allowed-Origin or "*".
- Completed: The handshake has been completed.
- Failed: The service has requested a handshake and the webhook responded with a code other than 2xx.
- Exempted: You can obtain a handshake exemption by adding the webhook URL under handshake exemptions.

Procedure

1. Navigate to your Cloud Foundry space and select your Event Mesh service instance.
2. In the left pane, select [Service Instances](#).
3. Choose [Open Dashboard](#) under the list of [Actions](#) that correspond to your Event Mesh service instance.
4. In the messaging administration page, choose [Webhook Subscriptions](#).

You can see the following information:

- List of webhook subscriptions available for your service instance
 - List of associated queues
 - Quality of Service used
 - Subscription status
 - Handshake status
5. To create a new webhook subscription:
 - a. Choose [Create](#).
 - b. Enter a subscription name and provide the webhook URL.
 - c. Select the queue, quality of service and authentication

Note

The list of queues in your service instance is populated.

- d. Choose [Create](#).
6. To manage handshake exemptions:
 - a. Choose [Manage Handshake Exemptions](#).
 - b. Provide the webhook URL you want to exempt from handshake and enter comments if necessary.
 - c. Choose [Add](#).
 - d. You can see the list of exempted webhook URLs and associated comments. Choose [Delete](#) to delete a handshake exemption.
 7. Under Actions, choose:
 - [Webhook Subscription Details](#) to view details of the corresponding webhook subscription.
 - [Trigger Handshake](#) to initiate a handshake for corresponding webhook subscription.
 - [Delete Webhook Subscription](#) to delete the corresponding webhook subscription.
 - [Pause](#) to pause the message delivery to the webhook URL.
 - [Resume](#) to resume the message delivery to the webhook URL.

Related Information

[REST APIs for Messaging \[page 23\]](#)

13.3.1.4 View Service Descriptor

You can view the JSON file you provided during service instance creation.

Procedure

1. Navigate to your Cloud Foundry space and select your Event Mesh service instance.
2. In the left pane, select [Service Instances](#).
3. Choose [Open Dashboard](#) under the list of [Actions](#) that correspond to your Event Mesh service instance.
4. In the messaging administration page, choose [Service Descriptor](#) to view the JSON file.

Related Information

[Create an Event Mesh Service Instance \[page 91\]](#)

[Create an Event Mesh Service Instance Using the Cloud Foundry Command Line Interface \[page 93\]](#)

13.3.1.5 Generate Application Configurations

An application can send and receive messages in a queue or topic using application configurations. They allow applications to be independent of messaging protocol endpoints.

Context

You can generate application configurations by creating a JSON file using the following properties or using the following procedure.

Property	Description	Requirement	Default Value	Valid Values
Input	Source from which the application receives messages.	Mandatory	Not applicable	JSON objects that define inputs.

Property	Description	Requirement	Default Value	Valid Values
Output	Destination to which the application sends messages.	Mandatory	Not applicable	JSON objects that define outputs.
Service	A valid service name inside the Cloud Foundry space.	Mandatory	Not applicable	Service name.
Address	An endpoint, for example, queue or topic.	Mandatory	Not applicable	<p>Any string that matches the name of a queue or a topic created in the service. It must contain the prefix, "queue:" (For example, "queue:name-ofqueue") or "topic:" ("topic:name/of/topic").</p> <p>For more information, see Syntax for Naming Queues, Topics, and Topic Patterns [page 30].</p>
Maximum messages in flight	This property is used when reliable is set to true. Messages in flight are messages that are stored in the service until the consumer acknowledges receipt. A limit can be set to the maximum number of messages that can be in flight with this property. For example, if the value is set as four, after four messages are sent to the consumer, if acknowledgment isn't received, the service waits until it receives an acknowledgment before sending next set of messages.	Optional	2000	<p>It's applicable only when reliable is true,</p> <ul style="list-style-type: none"> • A positive value greater than or equal to 512. • Zero to indicate that there's no restriction in the maximum number of messages.
Exclusive	To denote if an exclusive client connection handles the queue used. This property isn't available for topics.	Optional	<ul style="list-style-type: none"> • False for durable queues. • True for nondurable queues that belong to a session. 	<p>Only JSON Boolean values.</p> <div> <p>i Note</p> <p>It's applicable only when the address is a queue.</p> </div>

Property	Description	Requirement	Default Value	Valid Values
Reliable	For a client connection, this property is used as an acknowledgment for messages handled by libraries and the service. When reliable is set to true, the messages are guaranteed delivery at least once to the output/input (queues or topics).	Optional	True	Only JSON Boolean values.

The following example shows an application configuration:

Sample Code

```
{
  "inputs": {
    "input1": {
      "service": "event mesh service",
      "address": "queue:queue1",
      "reliable": true,
      "maxMsgInFlight": 2000
    }
  },
  "outputs": {
    "output1": {
      "service": "event mesh service",
      "address": "queue:queue1",
      "reliable": true,
      "maxMsgInFlight": 2000
    }
  }
}
```

Procedure

1. Choose [Open Dashboard](#) under Actions corresponding to your Event Mesh service instance.
2. In the left pane, select [Service Instances](#).
3. Navigate to your Cloud Foundry space and select your Event Mesh service instance.
4. In the messaging administration page, choose [Application Configurations](#).
5. Enter a configuration name.
6. Select the configuration type, message source or destination, and queue name.
The topics name or pattern is populated based on the queue subscription.
7. Enter the maximum number of messages that can be in flight.
8. Specify if the messages are reliable.
9. Choose [Generate Configuration](#).

13.3.1.6 Check Application Configuration

Check your application configuration to see if there are any missing properties.

Prerequisites

You have an application configuration.

Procedure

1. Navigate to your Cloud Foundry space and select your Event Mesh service instance.
2. In the left pane, select [Service Instances](#).
3. Choose [Open Dashboard](#) under Actions corresponding to your Event Mesh service instance.
4. In the messaging administration page, choose [Check Configuration](#).
5. Paste the application configuration that you have created or generated using [Generate Application Configurations \[page 103\]](#).
6. Choose [Check Configuration](#).

13.3.2 Events Administration

The events administration user interface allows you to view events that are available in your service instance from a particular event source.

You can access this user interface by subscribing to the [Event Mesh](#) business application. See [Subscribe to the Event Mesh Business Application \[page 96\]](#).

Related Information

[Manage Event Channel Groups \[page 107\]](#)

[Explore Events \[page 108\]](#)

[Integrating the Service with SAP S/4HANA \(Deprecated Lite Service Plan\) \[page 97\]](#)

13.3.2.1 Manage Event Channel Groups

As an administrator, you can create event channel groups that contain related events from different event sources.

Prerequisites

- You've subscribed to Event Mesh. See [Subscribe to the Event Mesh Business Application \[page 96\]](#).
- You've configured a destination. See [Configure Destinations to an SAP S/4HANA System \[page 98\]](#).
- You have the ManageRole assigned. See [Create User Groups, Role Collection and Assign Role Collection \[page 97\]](#).

Procedure

1. In the *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Event Mesh* tile.
3. Choose *Go to Application*. For this button to appear you must be subscribed to the application.
4. Choose *Manage Event Channel Groups*.

You see the following information:

- List of event channel groups.
 - Description of the event channel group.
5. To create a new event channel group:
 - a. Choose *Create Event Channel Group*.
 - b. Enter a name and description for the event channel group.
 - c. Select the event system and channel.
 - d. Choose *Create*.
 6. Under *Actions*, you can perform the following administrative tasks:
 - View information about the event channel group.
 - Edit the event system or channel for an event catalog. Here you can:
 - View the service instance associated with a target event system.
 - Delete an existing target event system.
 - Delete an event channel group.

13.3.2.2 Explore Events

As a developer, you can access an event channel group containing related events from multiple event sources using topics that are associated with the event channel group.

Prerequisites

- You've subscribed to Event Mesh. See [Subscribe to the Event Mesh Business Application \[page 96\]](#).
- You've configured a destination. See [Configure Destinations to an SAP S/4HANA System \[page 98\]](#).
- You have the ReadRole assigned. See [Create User Groups, Role Collection and Assign Role Collection \[page 97\]](#).

Procedure

1. In the *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Event Mesh* tile.
3. Choose *Go to Application*. For this button to appear you must be subscribed to the application.
4. Choose *Explore Events*.
5. Select an event channel group.

You can see the following information:

- List of event topics.
- Associated service instance for each topic.

Related Information

[Manage Event Channel Groups \[page 107\]](#)



[Initial Setup \(Deprecated Lite Service Plan\) \[page 89\]](#)

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.

About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
 - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
 - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.

The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

© 2022 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see <https://www.sap.com/about/legal/trademark.html> for additional trademark information and notices.