



User Guide | PUBLIC
2020-11-05

SAP Cloud Platform Enterprise Messaging

Content

1	What Is Enterprise Messaging?	4
1.1	Scope and Limitations.	6
2	What's New for Enterprise Messaging.	8
2.1	2019 Enterprise Messaging (Archive).	9
2.2	2018 What's New for Enterprise Messaging (Archive).	11
3	Concepts.	13
3.1	Messaging Protocols and Libraries.	16
	REST APIs for Messaging.	18
	REST APIs for Events.	21
3.2	Syntax for Service Descriptor.	23
3.3	Syntax for Naming Queues, Topics, and Topic Patterns.	28
3.4	Concepts (Deprecated Lite Service Plan).	30
	Messaging Protocols and Libraries.	31
4	Initial Setup.	34
4.1	Create an Enterprise Messaging Service Instance.	35
4.2	Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface.	37
4.3	Bind an Application to an Enterprise Messaging Service Instance.	39
4.4	Subscribe to Enterprise Messaging.	41
4.5	Assign Roles to Users.	42
4.6	Initial Setup (Deprecated Lite Service Plan).	45
	Set up a Subaccount.	46
	Create an Enterprise Messaging Service Instance.	47
	Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface.	48
	Bind an Application to an Enterprise Messaging Service Instance.	49
	Subscribe to the Enterprise Messaging Business Application.	52
	Create User Groups, Role Collection and Assign Role Collection.	52
	Integrating the Service with SAP S/4HANA (Deprecated Lite Service Plan).	53
5	Development.	55
5.1	REST APIs for Development.	55
	Use REST APIs to Send and Receive Messages.	55
	Use REST APIs to Manage Queues and Queue Subscriptions.	57
	Use REST APIs to Send and Receive Events.	58

6	Using Enterprise Messaging	60
6.1	View Rules	60
6.2	Manage Queues	61
6.3	Manage Webhooks	62
6.4	View Event Catalog for a Message Client	64
6.5	View Service Descriptor	65
6.6	View Event Catalog for a Subaccount	66
6.7	Monitor Resources	66
6.8	Test Publishing or Consuming Messages	67
6.9	Using Enterprise Messaging (Deprecated Lite Service Plan)	68
	Messaging Administration	68
	Events Administration	76
7	Security	79
7.1	Technical System Landscape	79
7.2	User Roles	80
7.3	Authentication and Authorization	80
7.4	Transport Encryption	82
7.5	Data Protection and Privacy	82
8	Monitoring and Troubleshooting	84
9	Glossary	85

1 What Is Enterprise Messaging?

Connect applications, services, and systems across different landscapes.

SAP Cloud Platform Enterprise Messaging is a fully-managed cloud service that allows applications to communicate through asynchronous events and seamlessly extend your digital core. Create responsive applications that work independently and participate in event-driven business processes inside your company and across your business ecosystem for greater agility and scalability.

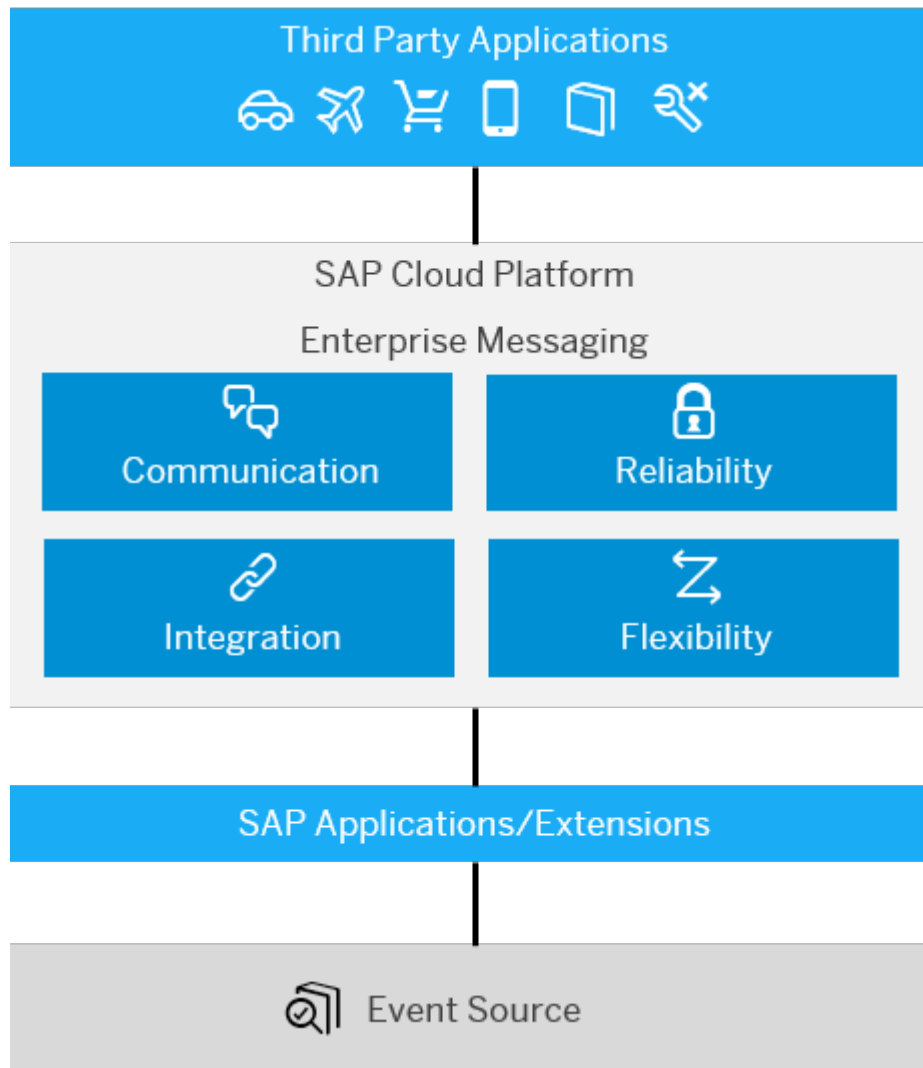
Environment

This service runs in the Cloud Foundry environment.

Features

Build event-driven extensions	Subscribe to business events from core SAP systems like SAP S/4HANA, SAP SuccessFactors, SAP Concur, and SAP Cloud for Customer or third-party sources to enable cloud-native extensions to respond to the latest business developments.
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.
Seamless connectivity	Provides reliable data transmission for extension and integration scenarios through decoupled communication.
Decouple communication	Decouples communication using standard asynchronous messaging patterns and supports open protocols for effective decoupling of devices and applications.

Overview



Tools

Tools	Description
SAP Cloud Platform Cockpit	The central point for managing all activities associated with your subaccount.
Enterprise Messaging 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 \[page 6\]](#).

Information

The lite service plan for SAP Cloud Platform Enterprise Messaging 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. See [Initial Setup](#).

Related Information

[Feature Scope Description](#)

[About Services](#)

1.1 Scope and Limitations

Find information on the service availability and technical limitations for SAP Cloud Platform Enterprise Messaging:

Regional Availability

Get an overview on the availability of Enterprise Messaging 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

- The messaging protocols supported by the service don't allow direct consumption of messages from a topic. Use queue subscriptions to receive messages from a topic.
- 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.
- 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 storage space for all messages in all the queues per subaccount is 10 GB.
- The maximum message throughput per subaccount is 250 KB/s.

Queues and Queue Subscriptions

- The maximum number of queues per subaccount is 250.
- The maximum number of queue subscriptions per subaccount is 10,000.

Connections

- The maximum number of connections per subaccount is 100.
- The maximum number of connections per message client (service instance) is 10.
- When REST APIs are used to publish messages, three connections are opened per message client (service instance). These connections are closed if there's no active message publish API call for a period of 30 minutes.
- When REST APIs are used to consume messages, three connections are opened per message client (service instance). These connections are closed if there's no active message consume API call for a period of 30 minutes.
- Three connections are opened when one or more webhooks are created per message client (service instance).

Producers

- The maximum number of producers per subaccount is 250.
- The maximum number of producers per connection is 10.
- When REST APIs are used to publish messages, three producers are opened per queue or topic. These producers are closed if there's no active publishing of messages for a period of 30 minutes.

Consumers and Webhooks

- The maximum number of consumers per subaccount is 250.
- The maximum number of consumers per connection is 10.
- The maximum consumer per queue per connection is 1 when you use AMQP over WebSocket.
- When REST APIs are used to consume messages, three consumers are opened per queue or topic. These consumers are closed if there's no active publishing of messages for a period of 30 minutes.
- Three consumers are opened for each webhook subscription.
- The maximum number of webhook subscriptions per message client (service instance) is 10.
- The maximum number of webhook subscriptions per subaccount is 75.

Related Information

[Messaging Protocols and Libraries \[page 16\]](#)

2 What's New for Enterprise Messaging

2020

Techni- cal Com- ponent	Capa- bility	Envi- ron- ment	Title	Description	Type	Availa- ble as of
Enter- prise Mes- saging	Integra- tion Suite	Cloud Foun- dry	UI En- hance- ment - Test Tab	The user interface for the Test tab has been enhanced with a split view.	Chang ed	2020-1 0-22
Enter- prise Mes- saging	Integra- tion Suite	Cloud Foun- dry	REST APIs for Send- ing and Receiv- ing 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	2020-1 0-22
Enter- prise Mes- saging	Integra- tion Suite	Cloud Foun- dry	Set De- fault Con- tent- type	You can now set the default content-type while creating a web-hook 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	2020-0 9-09
Enter- prise Mes- saging	Integra- tion Suite	Cloud Foun- dry	Down- load Bi- nary Mes- sages	When you test consuming a message, if the message is in binary format, it's automatically downloaded as a file to your local ma-chine. See Use a Message Client to Publish or Consume Messages .	Chang ed	2020-0 8-12
Enter- prise Mes- saging	Integra- tion Suite	Cloud Foun- dry	Mes- sage Prop- er- ties	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 .	Chang ed	2020-0 8-12
Enter- prise Mes- saging	Integra- tion Suite	Cloud Foun- dry	Purge Mes- sages	You can now delete all the messages in a queue at one go using the Purge Messages button under Actions. See Manage Queues .	New	2020-0 8-12

Technical Component	Capability	Environment	Title	Description	Type	Available as of
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
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

2.1 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 Cloud Platform Enterprise Messaging 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
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 23] .	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 18] .	Changed	2019-09-03

Technical Component	Capability	Environment	Title	Description	Type	Available as of
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 13] • Create an Enterprise Messaging Service Instance [page 35] • Syntax for Service Descriptor [page 23] 	New	2019-05-20
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 18].</p>	New	2019-05-20

2.2 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 18] .	New	2018-09-13
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

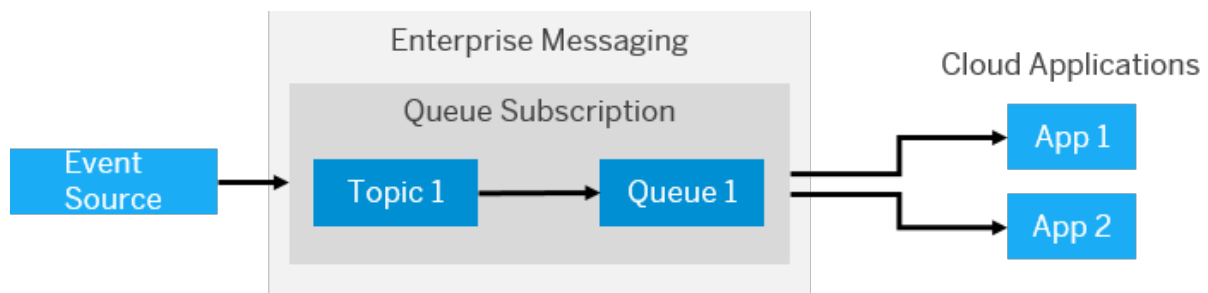
3 Concepts

SAP Cloud Platform Enterprise Messaging 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 event-based and messaging concepts:

Events

An event notifies a receiving application that an object at the event source 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.

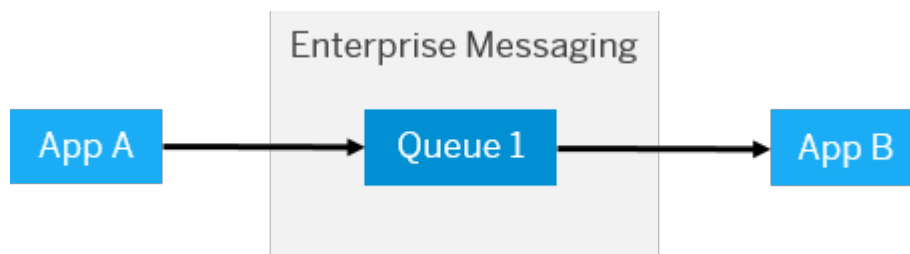


Asynchronous Messaging

Asynchronous messaging facilitates 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.

Queues

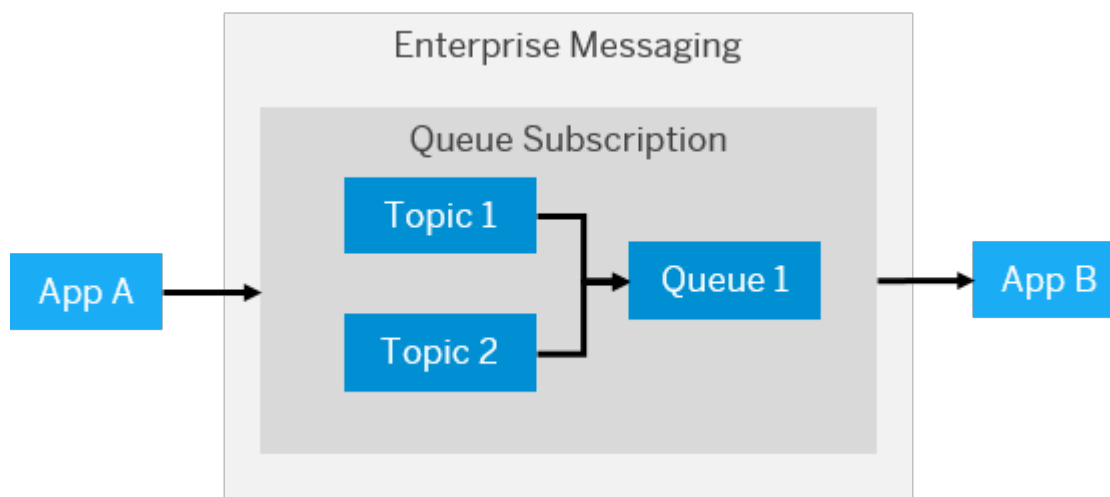
The service enables applications to communicate with each other through message queues. A sending application sends a message to a specific named queue. There's a one on one correspondence between a receiving application and its queue. The message queue retains the messages until the receiving application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

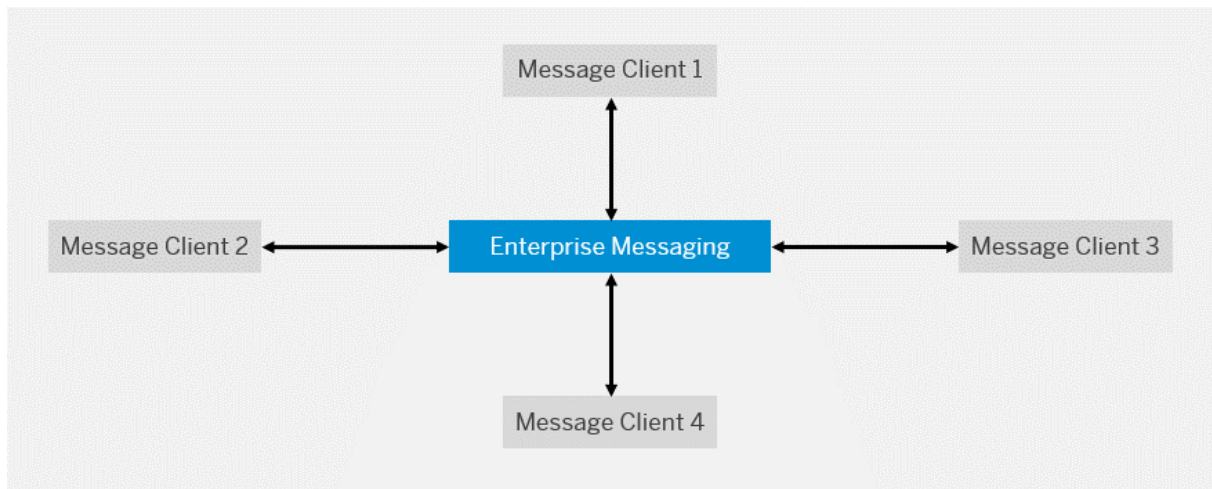


Message Client

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 Cloud Platform 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. The credentials must also define the list of queues or topics to which the message client can send and 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 made. The connection rules specify the queue or topic to which a message client must publish or consume messages.

The default service plan facilitates 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 one message client are exposed to other message clients through the unique credentials in the service descriptor.

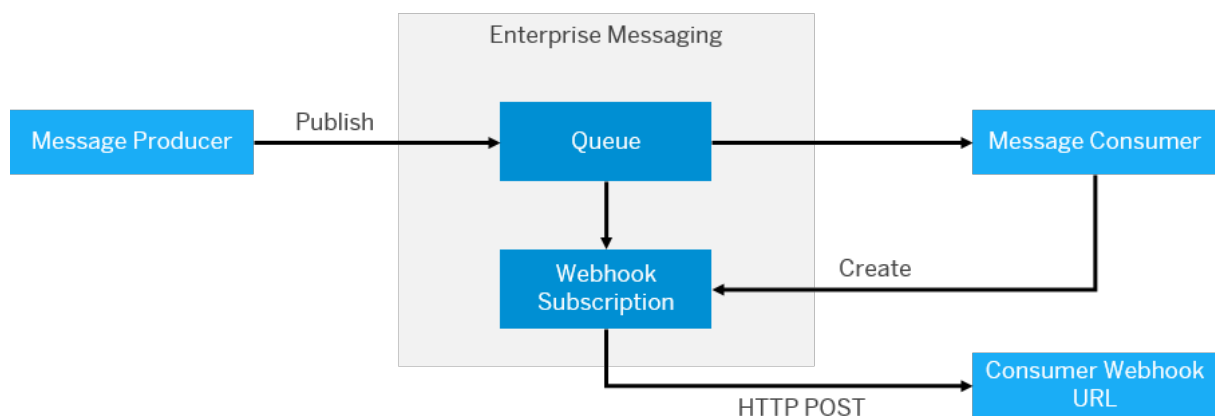


Event Catalog

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 catalog can contain events that are to be published or consumed. Each event in a catalog has a payload schema. An event source can provide an event catalog end point that gives the list of events that are published or consumed by the message client. The event catalog must adhere to the Async API specification, see [AsyncAPI spec](#) .

Webhooks

Webhooks allow you to set up an integration to send real-time data from one 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 POST 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. When an event is triggered, an HTTP POST payload is sent to the webhook URL. The data sent to the webhook URL is known as payload. See [REST APIs for Messaging \[page 18\]](#).



Related Information

[Scope and Limitations \[page 6\]](#)

[Syntax for Service Descriptor \[page 23\]](#)

[Syntax for Naming Queues, Topics, and Topic Patterns \[page 28\]](#)

[Messaging Protocols and Libraries \[page 16\]](#)

3.1 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 enterprise messaging 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 \[page 18\]](#).

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 \[page 21\]](#).

i Note

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

Libraries

The enterprise messaging 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 \[page 18\]](#)

[Development \[page 55\]](#)

3.1.1 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 \[page 57\]](#)

3.1.2 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 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.

Quality of Service with Event Publish (QoS)

The allowed values for specifying the Quality Of Service header are ATLEAST_ONCE and ATMOST_ONCE. When a client calls the service API for publishing events with ATLEAST_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 ATLEAST_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 ATLEAST_ONCE.

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, authentication details of the webhook, and quality of service.

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 ATLEAST_ONCE and ATMOST_ONCE. In the case of Quality of Service ATMOST_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 ATMOST_ONCE, the service tries to deliver the event to the webhook again. With QoS ATLEAST_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 \[page 58\]](#)

3.2 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 you service descriptor. A message client cannot publish or consume messages if rules are not 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 examples, `rules.topicRules.publishFilter` entries must always start with `${namespace}/...`
- Define your scenario in a consumption driven manner. For example, define your own queues and create a subscription for these queues to required topics.
- Publish to topics only not to queues directly. For example, `rules.queueRules.publishFilter` entry is not required if you follow this guideline.
- Consume from your own queues only. For example, `rules.queueRules.subscribeFilter` entries must always start with `${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.

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 Enterprise Messaging 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 is 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 is recommended to use a namespace that ends with `-` or `svc` in the last segment, for example, `default/sap.myapp/-` for provider instances. Consumer instances are not allowed to use namespaces that end with `-` or `svc`.

Syntax:

- Type: attribute
- Segments: exactly three (firstSegment/secondSegment/thirdSegment)
- Allowed characters:
 - 1st segment: [a-zA-Z0-9]
 - Hyphens and dots are not allowed
 - Recommended value: default
 - 2nd segment: [a-zA-Z0-9-.]
 - Starts with a character or number
 - Followed by . or -
 - Should contain at least one character or number before or after the . or -
 - 3rd segment: [a-zA-Z0-9-.]
 - Use the same guidelines given for the second segment
 - In case of provider scenarios use -
- 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.

instanceType

Description: The instanceType field is used for provider instances to distinguish it as a provider instance.

Syntax:

- Type: object
- Required: false
- Attribute: reuse

Guideline: Remember the following points when you use a provider instance:

- Subscriptions are only allowed on provider instances
- Namespaces that end with svc or - are allowed only for provider instances
- The instanceType field is required only for provider instances and can be completely omitted in any other service instance's service descriptor.

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

Related Information

[Create an Enterprise Messaging Service Instance \[page 35\]](#)

[Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface \[page 37\]](#)

3.3 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 \[page 23\]](#).

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 are not 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

3.4 Concepts (Deprecated Lite Service Plan)

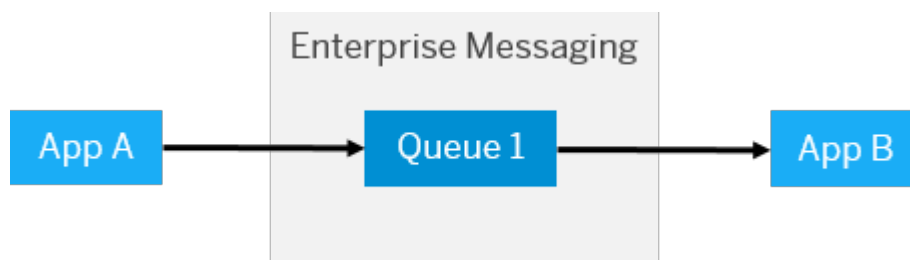
SAP Cloud Platform Enterprise Messaging employs a centralized message-oriented architecture. It's more scalable and reliable compared to the traditional point-to-point communication model.

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:

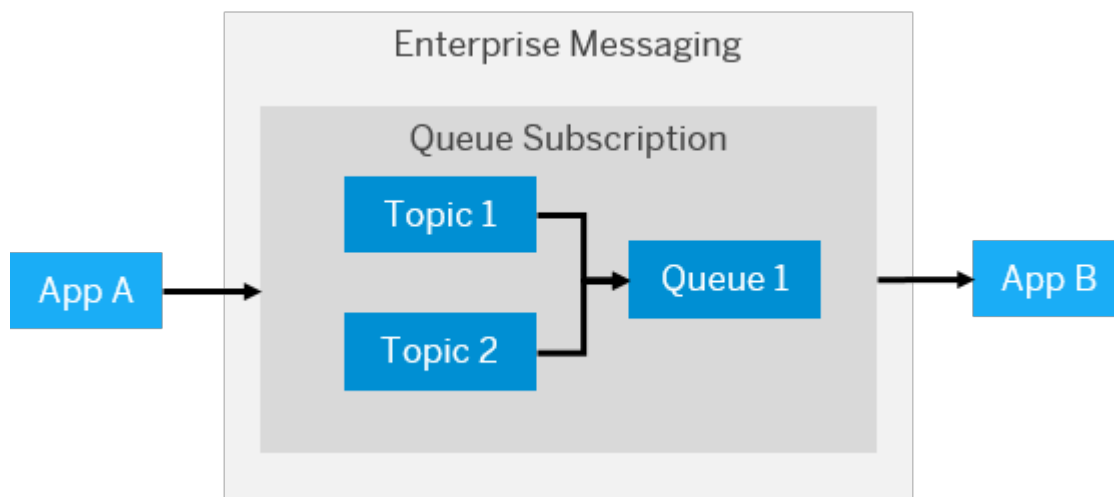
Queues

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 \(Deprecated Lite Service Plan\) \[page 45\]](#)

[Using Enterprise Messaging \(Deprecated Lite Service Plan\) \[page 68\]](#)


3.4.1 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 enterprise messaging 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](#) .

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 \[page 18\]](#).

i Note

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

Libraries

The enterprise messaging 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.

4 Initial Setup

After you purchase the Enterprise Messaging default plan, you must perform the following steps in the SAP Cloud Platform cockpit.

Prerequisites

- You have a customer account with SAP Cloud Platform and an Enterprise Messaging 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 Enterprise Messaging. See [Get a Paid Global Account \[AWS, Azure, or GCP\]](#)
- 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 Enterprise Messaging 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 Cloud Platform cockpit, navigate to your global account and assign the entitlement for the default service plan for Enterprise Messaging 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](#) and then, [Add Service Plans](#).
 - d. In the [Subaccount Entitlements](#) dialog box, select the Enterprise Messaging service.
 - e. In the [Service Details: Enterprise Messaging](#) window, select the default service plan.
 - f. Choose [Add 1 Service Plans](#) to add this entitlement for the Enterprise Messaging service for your subaccount.

- g. Choose [Save](#).
2. Create a service instance for Enterprise Messaging using either the cockpit or cf CLI. See:
 - [Create an Enterprise Messaging Service Instance \[page 35\]](#).
 - [Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface \[page 37\]](#).

→ Remember

The lite service plan for SAP Cloud Platform Enterprise Messaging 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. [Bind an Application to an Enterprise Messaging Service Instance \[page 39\]](#).
4. [Subscribe to Enterprise Messaging \[page 41\]](#).
5. [Assign Roles to Users \[page 42\]](#).
6. To consume SAP S/4HANA Cloud events using the SAP Cloud Platform Extension Factory, see [Enabling the Consumption of SAP S/4HANA Cloud Events](#).

Related Information

[Configure Entitlements and Quotas for Subaccounts](#)

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

[Blog: New service plan and UI for SAP Cloud Platform Enterprise Messaging](#) 

4.1 Create an Enterprise Messaging Service Instance

Use the SAP Cloud Platform cockpit to create a service instance for the Enterprise Messaging service default 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](#) [Enterprise Messaging](#).
3. Choose [Instances](#) [New Instance](#).
4. Select the default service plan and choose [Next](#).
5. 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 23].

Sample Code

```
{
  "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}/*"
      ]
    }
  }
}
```

Note

- We recommend that you use the same value for `service instance name` and `emname`.
- For existing service instances, update your service instance to include the parameters in your JSON file. See, [Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface](#) [page 37].

6. Enter the instance name and choose [Finish](#).
 - If you want to receive events from SAP S/4HANA using SAP Cloud Platform Extension Factory, see [Enabling the Consumption of SAP S/4HANA Cloud Events](#).
 - If you want to delete an Enterprise Messaging service instance, choose [Delete](#) under Actions.

Related Information

[Samples on GitHub for Java](#) 

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

4.2 Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface

Create an Enterprise Messaging service instance using the Cloud Foundry Command Line Interface (cf CLI) for the default service plan.

Prerequisites

- You have the administrator role for your global account.
- 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 enterprise messaging service in the Cloud Foundry marketplace.
4. Access an enterprise messaging 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 \[page 23\]](#).

Sample Code

```
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}/*"
        ]
    }
}
}
'

```

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. Update an existing service instance, enter `cf update-service <service-instance-name> -c 'service-descriptor.json'`

Sample Code

```

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}/*"
      ]
    }
  }
}
'

```

4.3 Bind an Application to an Enterprise Messaging Service Instance

Bind your messaging application to your Enterprise Messaging service instance for the default 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 enterprise messaging 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 enterprise messaging:

```
{
  "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"
    },
    {
      "oa2": {
        "clientid":
"<client_id>",
```

```

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

```

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 Enterprise Messaging.5. Select the Default 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](#)

4.4 Subscribe to Enterprise Messaging

Subscribe to the Enterprise Messaging business application to access its user interface.

Prerequisites

- You've completed the [initial setup \[page 34\]](#).

- You've created a service instance. See
 - [Create an Enterprise Messaging Service Instance \[page 35\]](#).
 - [Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface \[page 37\]](#).

Context

The user interface for the default plan is provided as a multitenant business application. Subscriptions can be set up only by administrators of the global account.

Procedure

1. Open your global account, then your subaccount.
2. In the left pane, choose *Subscriptions*.
3. Choose *Enterprise Messaging*.
4. In the *Subscription: Enterprise Messaging - Overview* page, choose *Subscribe*.

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

5. Choose the link to launch the application and obtain its URL.

4.5 Assign 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 Cloud Platform cockpit.

Prerequisites

- You have the administrator role for your global account.
- You've subscribed to the Enterprise Messaging business application, see [Subscribe to Enterprise Messaging \[page 41\]](#).
- 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](#).
- (Optional) [Create a role collection](#). Use a role collection if you want to add multiple roles to a user or user group.

- (Optional) [Create a new user group](#). You can also assign roles to individual platform users.

Context

You can assign roles to users based on the type of trust configuration you've enabled in SAP Cloud Platform. The following options are available:

- Directly assign role collections to users.
- 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 Cloud Platform, and maintain group memberships of users in the identity provider.

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

The roles that can be assigned include the following:

- ReadRole (developer) - View queues, rules, service descriptor, webhook subscriptions, message clients, event channel groups and explore events.
- ManageRole (administrator) - Create, edit, and delete queues, rules, service descriptor, webhook subscriptions, message clients, and event channel groups along with the ReadRole.
- TestRole - Test sending and receiving messages through the user interface.

Note

It's recommended not to use the following roles as it is for internal use only:

- OperationsManageRole
- OperationsReadRole
- OperationsReadDetailRole

Procedure

1. Navigate to your subaccount.
2. In the left pane, choose [Subscriptions](#).
3. Choose the [Enterprise Messaging](#) tile then, [Manage Roles](#) to view, create, and modify the application roles.
4. Go back to your subaccount and choose [Security](#) [Role Collections](#) in the left pane to add the standard roles provided with the service to a role collection.
5. Choose [Security](#) [Trust Configuration](#).
6. Choose your active trust configuration.
7. Select one of the following methods to proceed:

Trust Configuration Used	Steps
Default trust configuration (SAP ID service)	<ol style="list-style-type: none"> 1. Choose Role Collection Assignment. 2. Enter the business user's name, for example <code>john.doe@example.com</code>. 3. Enter the user name, for example <code>john.doe@example.com</code>. <div> <p>→ Tip</p> <p>If the user identifier you've entered has never logged on to an application in this subaccount, SAP Cloud Platform can't automatically verify the user name. To avoid mistakes, you must check and confirm that the user name is correct.</p> </div> <ol style="list-style-type: none"> 4. To see the role collections that are currently assigned to this user, choose Show Assignments. 5. To assign a role collection, choose Assign Role Collection. 6. Choose the name of the role collection you want to assign. The following role collections are available by default: <ul style="list-style-type: none"> ◦ Enterprise Messaging Display: It provides the ReadRole. ◦ Enterprise Messaging Developer: It provides the ManageRole and the TestRole. ◦ Enterprise Messaging Administrator: It provides the ManageRole. 7. Save your changes.
Custom trust configuration (Identity Authentication service or any SAML 2.0 identity provider)	<p>You can perform the preceding steps or</p> <ol style="list-style-type: none"> 1. Choose Role Collection Mappings. 2. Choose New Role Collection Mapping. 3. Choose the role collection you want to map and enter the name of the user group in the Value field. <div> <p>❖ Example</p> <p>In the SAP Cloud Platform Identity service, you can find user groups in the Administration console of your Identity Authentication service tenant under Users & Authorizations > User Groups. Open the Administration console of, for example, the Identity Authentication service using <code>https://<Identity_Authentication_tenant>.accounts.ondemand.com/admin</code>.</p> </div> <ol style="list-style-type: none"> 4. The following role collections are available by default: <ul style="list-style-type: none"> ◦ Enterprise Messaging Display: It provides the ReadRole. ◦ Enterprise Messaging Developer: It provides the ManageRole along with the ReadRole and TestRole. ◦ Enterprise Messaging Administrator: It provides the ManageRole. 5. Save your changes.

Related Information

[Assign role collection to users/user groups](#)

[Add roles to role collections](#)

4.6 Initial Setup (Deprecated Lite Service Plan)

Trial Scope

→ Remember

The trial availability for Enterprise Messaging 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.

Enterprise Messaging is available on trial. A trial account lets you try out SAP Cloud Platform 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 Cloud Platform cockpit.

1. [Set up a Subaccount \[page 46\]](#)
2. [Create an Enterprise Messaging Service Instance \[page 47\]](#) or [Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface \[page 48\]](#)
3. [Bind an Application to an Enterprise Messaging Service Instance \[page 39\]](#)
4. [Subscribe to the Enterprise Messaging Business Application \[page 52\]](#)
5. [Create User Groups, Role Collection and Assign Role Collection \[page 52\]](#)
6. [Integrating the Service with SAP S/4HANA \(Deprecated Lite Service Plan\) \[page 53\]](#)

4.6.1 Set up a Subaccount

After you purchase an Enterprise Messaging lite plan, you must perform the following steps in the SAP Cloud Platform cockpit.

Prerequisites

- You have a customer account with SAP Cloud Platform and an Enterprise Messaging lite service plan. See [Getting Started with a Customer Account: Workflow in the Cloud Foundry Environment](#).
- You have a global account. See [Get a Paid Global Account \[AWS, Azure, or GCP\]](#)
- 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 Enterprise Messaging 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 Cloud Platform cockpit, navigate to your global account and assign the entitlement for the lite service plan for Enterprise Messaging to your subaccount, see [Configure Entitlements and Quotas for Subaccounts](#).
2. Create a service instance for Enterprise Messaging using either the cockpit or cf cli, see
 - [Create an Enterprise Messaging Service Instance \[page 47\]](#).
 - [Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface \[page 48\]](#).

→ Remember

The lite service plan for SAP Cloud Platform Enterprise Messaging 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. [Bind an Application to an Enterprise Messaging Service Instance \[page 39\]](#).
4. To access the events administration user interface, you must [Subscribe to the Enterprise Messaging Business Application \[page 52\]](#).

5. [Create User Groups, Role Collection and Assign Role Collection \[page 52\]](#).
6. If you want to receive events from an SAP S/4HANA system, see [Integrating the Service with SAP S/4HANA \(Deprecated Lite Service Plan\) \[page 53\]](#).

4.6.2 Create an Enterprise Messaging Service Instance

Use the SAP Cloud Platform 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](#) ► [Enterprise Messaging 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" or enterprise messaging name 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 Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface \[page 48\]](#).

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.

4.6.3 Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface

Create an Enterprise Messaging 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 enterprise messaging service in the Cloud Foundry marketplace.

4. Access an enterprise messaging service instance using one of the following methods:

- 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. Update an existing service instance, enter `cf update-service <service-instance-name> -c 'service-descriptor.json'`

Sample Code

```
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":"<enterprise messaging 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
```

4.6.4 Bind an Application to an Enterprise Messaging Service Instance

Bind your messaging application to your Enterprise Messaging 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 enterprise messaging 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 enterprise messaging:

```
{
  "xsappname": "<app-name>",
  "serviceinstanceid": "<instance-id>",
  "messaging": [
    {
      "oa2": {
        "clientid":
        "clientsecret":
        "tokenendpoint":
        "granttype":
      },
      "protocol": [
        "amqp10ws"
      ],
      "broker": {
        "type":
      },
      "uri": "wss://<app-url>/
    },
    {
      "oa2": {
        "clientid":
        "clientsecret":
        "tokenendpoint":
        "granttype":
      },
      "protocol": [
        "mqtt311ws"
      ],
      "broker": {
        "type":
      },
      "uri": "wss://<app-url>/
    },
    {
      "oa2": {
        "clientid":
        "clientsecret":
        "tokenendpoint":
        "granttype":
      },
      "protocol": [
        "httpprest"
      ]
    }
  ]
}
```

```

"saprestmgw"
    ],
    "management": [
        {
            "<client_id>",
            "<client_secret>",
            "https://<app-url>/oauth/token",
            "client_credentials"
        }
    ]
},
"broker": {
    "type":
},
"uri": " https://<app-url>/\"
},
"oa2": {
    "clientid":
    "clientsecret":
    "tokenendpoint":
    "granttype":
},
"uri": " https://<app-url>/\"
}

```

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 Enterprise Messaging. 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.

View	Steps
	<ol style="list-style-type: none"> (Optional) Specify parameters in a JSON format or browse and upload a JSON file. 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](#)

4.6.5 Subscribe to the Enterprise Messaging Business Application

Subscribe to SAP Cloud Platform Enterprise Messaging to access the Events Administration user interface.

The Events Administration user interface for the lite plan is provided as multitenant business application. Select the [Enterprise Messaging](#) 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.

4.6.6 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:

- [Create a new user group.](#)
- [Create a new role collection with the standard roles](#)

3. [Assign role collection to users/user groups](#)

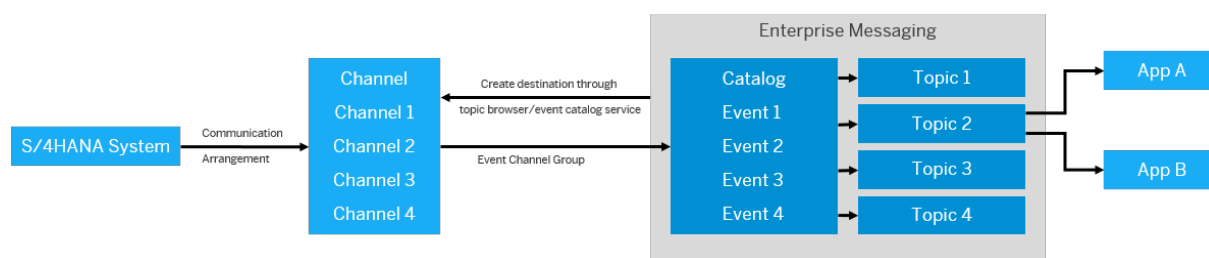
i Note

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.

4.6.7 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](#)

[Blog: SAP Cloud Platform Enterprise Messaging – Making S/4HANA Event Notification Easy](#)

[Configure Destinations to an SAP S/4HANA System \[page 54\]](#)

4.6.7.1 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 Cloud Platform.

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 Cloud Platform. 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 \(Cockpit\)](#)

5 Development

You can deploy messaging applications that you have developed using Java or Node.js to the service.

- Develop Enterprise Messaging applications on SAP Cloud Platform, see [Tutorials](#) .
- To build messaging applications using Java, see [Samples on GitHub for Java](#) .
- To build messaging applications using Node.js, see [Samples on GitHub for Node.js](#) .
- To use the REST APIs provided with the service, see [REST APIs for Development \[page 55\]](#).

5.1 REST APIs for Development

This section contains information on the APIs that can be used for development with Enterprise Messaging.

Related Information

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

[Use REST APIs to Manage Queues and Queue Subscriptions \[page 57\]](#)

[REST APIs for Messaging \[page 18\]](#)

5.1.1 Use REST APIs to Send and Receive Messages

Use REST APIs to send and receive messages with Enterprise Messaging.

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. From the service key, note the following:
 - Client ID
 - Client Secret
 - Token endpoint
 - Base URL

Note

In the "messaging" section for the protocol "httprest" of the service key, the client ID, client secret, and token endpoint are present under "oa2" and the base URL is under "uri".

3. Append `?grant_type=client_credentials&response_type=token` to the token endpoint and do a POST request to receive the OAuth token response.

The topic or queue name needs to be URL encoded. The token URL is `<TOKENENDPOINT>?grant_type=client_credentials&response_type=token`.

4. Provide the client ID and client secret as the username and password for basic authentication.
5. Obtain the `access_token` generated as a part of the response.
6. Add this access token `Authorization: Bearer <access_token>` as the authorization header to access messagingrest endpoints.

Note

The messaging APIs are protected with an OAuth bearer token.

7. 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: `https://enterprise-messaging.com/messagingrest/v1/queues/org%2Fsample%2F001%2Fq1/messages`

5.1.2 Use REST APIs to Manage Queues and Queue Subscriptions

Use REST APIs to manage queues and queue subscriptions with Enterprise Messaging.

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. From the service key, note the following:
 - Client ID
 - Client Secret
 - Token endpoint
 - Base URL

Note

In the "management" section of the service key, the client ID, client secret, and token endpoint are present under "oa2" and the base URL is under "uri".

3. Append `?grant_type=client_credentials&response_type=token` to the token endpoint and do a POST request to receive the OAuth token response.

The token URL is `<TOKENENDPOINT>?grant_type=client_credentials&response_type=token`.
The topic or queue name needs to be URL encoded.

Example

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

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

Resultant REST API call: `https://enterprise-messaging.com/messagingrest/v1/queues/org%2Fsample%2F001%2Fq1/messages`

4. Provide the client ID and client secret as the username and password for basic authentication.
5. Obtain the `access_token` generated as a part of the response.
6. Add this access token `Authorization: Bearer <access_token>` as the authorization header to access messagingrest endpoints.

Note

The management APIs are protected with an OAuth bearer token.

7. To obtain information for a queue, make a GET 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: `https://enterprise-messaging.com/messagingrest/v1/queues/org%2Fsample%2F001%2Fq1/messages`

5.1.3 Use REST APIs to Send and Receive Events

Use REST APIs to send and receive events with Enterprise Messaging.

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. From the service key, note the following:
 - Client ID
 - Client Secret

- Token endpoint
- Base URL

i Note

In the `management` or `messagingrest` section of the service key, the client ID, client secret, and token endpoint are present under `"oa2"` and the base URL is under `"uri"`.

3. Append `?grant_type=client_credentials&response_type=token` to the token endpoint and do a POST request to receive the OAuth token response.

The token URL is `<TOKENENDPOINT>?grant_type=client_credentials&response_type=token`.
The topic or queue name needs to be URL encoded.

❖ Example

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

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

Resultant REST API call: `https://enterprise-messaging.com/messagingrest/v1/queues/org%2Fsample%2F001%2Fq1/messages`

4. Provide the client ID and client secret as the username and password for basic authentication.
5. Obtain the `access_token` generated as a part of the response.
6. For all requests, use this access token `Authorization: Bearer <access_token>` as the authorization header to access events endpoints.
7. Based on whether you choose binary or structured mode, ensure you provide the headers required based on the CloudEvents specification, see [REST APIs for Events \[page 21\]](#).

6 Using Enterprise Messaging

The user interface for the default plan allows you to manage messaging clients and event catalogs for a subaccount.

To access the user interface, you need to subscribe to the *Enterprise Messaging* business application, see [Subscribe to Enterprise Messaging \[page 41\]](#).

Related Information

[View Rules \[page 60\]](#)

[Manage Queues \[page 61\]](#)

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

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

[View Service Descriptor \[page 65\]](#)

[View Event Catalog for a Subaccount \[page 66\]](#)

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

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

6.1 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 \[page 34\]](#).
- You have the ReadRole assigned. See [Assign Roles to Users \[page 42\]](#).

Procedure

1. In the *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Enterprise Messaging* 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.

6.2 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 \[page 34\]](#).
- You have the ManageRole assigned. See [Assign Roles to Users \[page 42\]](#).

Context

Queue names must follow the naming convention specified in [Syntax for Naming Queues, Topics, and Topic Patterns \[page 28\]](#).

Procedure

1. In the *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Enterprise Messaging* 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](#)

6.3 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 34].
- You've provided `"messagingrest": true` while creating a service instance. For more information, see [Create an Enterprise Messaging Service Instance](#) [page 35].
- You have the ManageRole assigned. See [Assign Roles to Users](#) [page 42].
- (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 *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Enterprise Messaging* 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.

i 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](#)

6.4 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 \[page 34\]](#).
- You have the ReadRole assigned. See [Assign Roles to Users \[page 42\]](#).
- You've integrated the service with SAP S/4HANA to receive cloud events, see [Enabling the Consumption of SAP S/4HANA Cloud Events](#).

Procedure

1. In the *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Enterprise Messaging* 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.

6.5 View Service Descriptor

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

Prerequisites

- You've completed the [initial setup \[page 34\]](#).
- You have the ReadRole assigned. See [Assign Roles to Users \[page 42\]](#).

Procedure

1. In the *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Enterprise Messaging* 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 \[page 23\]](#)

[Create an Enterprise Messaging Service Instance \[page 35\]](#)

6.6 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 \[page 34\]](#).
- You have the ReadRole assigned. See [Assign Roles to Users \[page 42\]](#).
- You've integrated the service with SAP S/4HANA to receive cloud events, see [Enabling the Consumption of SAP S/4HANA Cloud Events](#).

Procedure

1. In the *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Enterprise Messaging* 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.

6.7 Monitor Resources

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

Prerequisites

- You've completed the [initial setup \[page 34\]](#).
- You have the ReadRole assigned. See [Assign Roles to Users \[page 42\]](#).

Procedure

1. In the *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Enterprise Messaging* 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 \[page 6\]](#)

6.8 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 \[page 34\]](#).
- You have the TestRole assigned. See [Assign Roles to Users \[page 42\]](#).
- For publishing messages to a topic, you've created a queue subscription. See [Manage Queues \[page 61\]](#).

Procedure

1. In the *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Enterprise Messaging* 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.

6.9 Using Enterprise Messaging (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 \[page 68\]](#)

[Events Administration \[page 76\]](#)

6.9.1 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 \[page 69\]](#)

[Manage Queue Subscriptions \[page 70\]](#)

[Generate Application Configurations \[page 73\]](#)

[Check Application Configuration \[page 76\]](#)

6.9.1.1 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 \[page 28\]](#).

Procedure

1. Navigate to your Cloud Foundry space and select your Enterprise Messaging service instance.
2. In the left pane, select [Service Instances](#).
3. Choose [Open Dashboard](#) under the list of [Actions](#) that correspond to your Enterprise Messaging 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).

6.9.1.2 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 \[page 28\]](#).

Procedure

1. Navigate to your Cloud Foundry space and select your Enterprise Messaging service instance.
2. In the left pane, select [Service Instances](#).
3. Choose [Open Dashboard](#) under the list of [Actions](#) that correspond to your Enterprise Messaging 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.

6.9.1.3 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 Enterprise Messaging Service Instance \[page 47\]](#).

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 Enterprise Messaging service instance.
2. In the left pane, select [Service Instances](#).
3. Choose [Open Dashboard](#) under the list of [Actions](#) that correspond to your Enterprise Messaging 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

i 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 \[page 18\]](#)

6.9.1.4 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 Enterprise Messaging service instance.
2. In the left pane, select [Service Instances](#).
3. Choose [Open Dashboard](#) under the list of [Actions](#) that correspond to your Enterprise Messaging service instance.
4. In the messaging administration page, choose [Service Descriptor](#) to view the JSON file.

Related Information

[Create an Enterprise Messaging Service Instance \[page 47\]](#)

[Create an Enterprise Messaging Service Instance Using the Cloud Foundry Command Line Interface \[page 48\]](#)

6.9.1.5 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.
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.

Property	Description	Requirement	Default Value	Valid Values
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:nameof-queue") or "topic:" ("topic:name/of/topic").</p> <p>For more information, see Syntax for Naming Queues, Topics, and Topic Patterns [page 28].</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.

Property	Description	Requirement	Default Value	Valid Values
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 non-durable queues that belong to a session. 	Only JSON Boolean values. <div> Note It's applicable only when the address is a queue. </div>
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": "enterprise messaging service",
      "address": "queue:queue1",
      "reliable": true,
      "maxMsgInFlight": 2000
    }
  },
  "outputs": {
    "output1": {
      "service": "enterprise messaging service",
      "address": "queue:queue1",
      "reliable": true,
      "maxMsgInFlight": 2000
    }
  }
}
```

Procedure

1. Choose [Open Dashboard](#) under Actions corresponding to your Enterprise Messaging service instance.
2. In the left pane, select [Service Instances](#).
3. Navigate to your Cloud Foundry space and select your Enterprise Messaging 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](#).

6.9.1.6 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 Enterprise Messaging service instance.
2. In the left pane, select [Service Instances](#).
3. Choose [Open Dashboard](#) under Actions corresponding to your Enterprise Messaging 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 73].
6. Choose [Check Configuration](#).

6.9.2 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 [Enterprise Messaging](#) business application. See [Subscribe to the Enterprise Messaging Business Application](#) [page 52].

Related Information

[Manage Event Channel Groups \[page 77\]](#)

[Explore Events \[page 78\]](#)

[Integrating the Service with SAP S/4HANA \(Deprecated Lite Service Plan\) \[page 53\]](#)

6.9.2.1 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 Enterprise Messaging. See [Subscribe to the Enterprise Messaging Business Application \[page 52\]](#).
- You've configured a destination. See [Configure Destinations to an SAP S/4HANA System \[page 54\]](#).
- You have the ManageRole assigned. See [Create User Groups, Role Collection and Assign Role Collection \[page 52\]](#).

Procedure

1. In the *SAP Cloud Platform Cockpit*, navigate to your subaccount.
2. Choose *Subscriptions* from the navigation pane on the left, then the *Enterprise Messaging* 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.

6.9.2.2 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 Enterprise Messaging. See [Subscribe to the Enterprise Messaging Business Application \[page 52\]](#).
- You've configured a destination. See [Configure Destinations to an SAP S/4HANA System \[page 54\]](#).
- You have the ReadRole assigned. See [Create User Groups, Role Collection and Assign Role Collection \[page 52\]](#).

Procedure

1. In the [SAP Cloud Platform Cockpit](#), navigate to your subaccount.
2. Choose [Subscriptions](#) from the navigation pane on the left, then the [Enterprise Messaging](#) 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 \[page 77\]](#)

[Initial Setup \(Deprecated Lite Service Plan\) \[page 45\]](#)

7 Security

SAP Cloud Platform Enterprise Messaging security information describes the policies, technologies, and controls that are applicable specifically to the service to protect data, applications, and their associated infrastructure within SAP Cloud Platform.

Security aspects include the following:

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

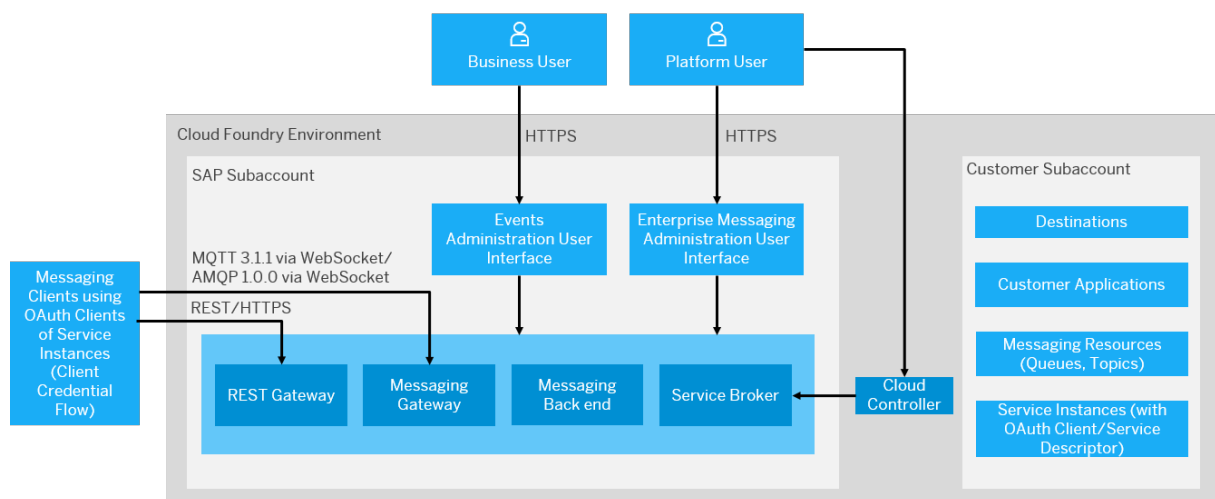
SAP Cloud Platform Enterprise Messaging runs on SAP Cloud Platform, for this reason, all security requirements for SAP Cloud Platform are also applicable to the service.

7.1 Technical System Landscape

Use SAP Cloud Platform Enterprise Messaging 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](#).



7.2 User Roles

User authentication in SAP Cloud Platform Enterprise Messaging uses the authentication methods supported by SAP Cloud Platform, which use the Security Assertion Markup Language (SAML) 2.0 protocol for both authentication and single sign-on.

User Type	Usage	Authentication	Authorization/Scope
Platform user	<ul style="list-style-type: none">Enterprise messaging administrationEvents administration and enterprise messaging 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 is used for technical communication with the service during binding.	OAuth (Client credentials flow)	Not applicable
Business user	<ul style="list-style-type: none">Enterprise messaging administrationEvents administration and enterprise messaging administration back end	OAuth (Authorization code flow)	<ul style="list-style-type: none">ManageRead

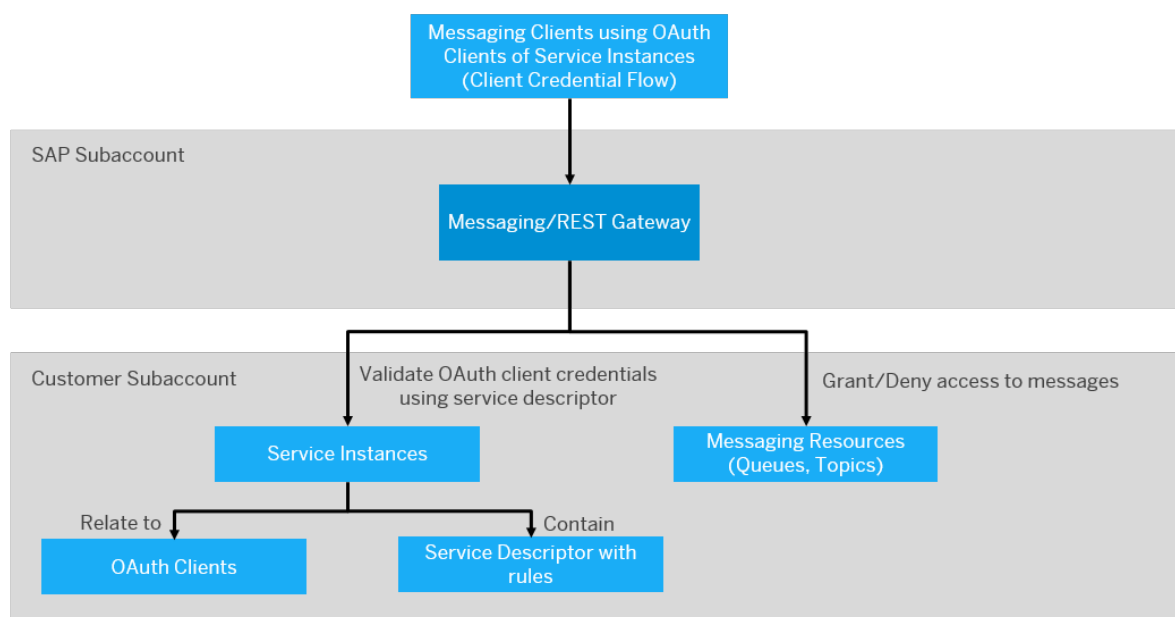
7.3 Authentication and Authorization

Authorization in SAP Cloud Platform Enterprise Messaging 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.

Component	Description
Enterprise Messaging 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 enterprise messaging 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	<p>The events administration component is called by a business user. 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 enterprise messaging administration and events administration back end. The back-end component then checks whether the token has the appropriate OAuth scopes.</p>
Management	<p>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.</p>



Platform as a Service Scenario

7.4 Transport Encryption

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

7.5 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 Cloud Platform Enterprise Messaging 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 enterprise messaging 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)
- Encrypt messages at the application level.

i 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.

Related Information

[Security](#)

8 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 Cloud Platform at [SAP Trust Center](#). See,

- the availability by service on the [SAP Cloud Platform](#) 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 Enterprise Messaging](#). 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-CF-MES	Enterprise Messaging 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

9 Glossary

This section contains the commonly used terms in the enterprise messaging 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.
Event	An event is a message that is sent to notify a consumer that an object has changed.
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.
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.

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.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

© 2020 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.