Acceleronix

Hyper MQ User Guide

IoT Platforms&Solutions

Version: 1.0

Date: 2026-07-15

Status: Released

At Acceleronix, our aim is to offer expert consultancy and an extensive partner network to guide you from the start. We provide comprehensive solutions—from PCBA design to global SIM connectivity and our IoT platform—all in one place.

Acceleronix
Headquarters: Haaksbergweg 75, 1101 BR Amsterdam, Netherlands
Email: sales@acceleronix.io

For more information, please visit:
https://www.acceleronix.io/

For technical support, or to report documentation errors, please email us at:
support@acceleronix.io

Legal Notices

We provide this document to support your product design. You are required to design your products based on the specifications and parameters set forth herein. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. You acknowledge and agree that we may add to, amend, or restate this document at any time at our sole discretion without any prior notice to you, and such additions, amendments, or restatements shall be binding upon you.

Use and Disclosure Restrictions

License Agreements

The recipient of any hardware, software, materials, or documentation provided by us shall keep such content confidential, unless expressly authorized by us. The recipient shall not disclose, access, or use any part of the received content for any purpose other than the execution and implementation of the intended project.

Copyright

Our and third-party products hereunder may contain copyrighted materials, including but not limited to protected content, hardware, software, and documentation owned by us or applicable third parties. Unless prior written consent is obtained, you shall not access, use, or disclose any documents or information provided by us, nor shall you copy, reproduce, republish, display, translate, distribute, merge, modify, or create derivative works from any such copyrighted materials. We and the applicable third party retain exclusive rights to all copyrighted materials. No license to any patents, copyrights, trademarks, or service marks shall be granted or transferred. For the avoidance of doubt, no form of purchase shall be construed as granting any license beyond a normal, non-exclusive, royalty-free license to use the product. We reserve the right to pursue legal action against any violation of confidentiality obligations, unauthorized use, or any other unlawful or malicious use of the aforementioned documents and

information.

Trademarks

Unless otherwise expressly provided, nothing in this document shall be construed as conferring any rights to use any trademark, trade name, name, abbreviation, or counterfeit thereof owned by us or any third party in advertising, publicity, or any other contexts.

Third-Party Rights

You understand that this document may refer to hardware, software, and/or documentation owned by one or more third parties (“third-party materials”). Use of such third-party materials is subject to all applicable restrictions and obligations set forth herein.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing, course of performance, or usage of trade.

Privacy Policy

To enable product functionality, certain device data may be uploaded to our or third-party servers, including those operated by carriers, chipset suppliers, or servers designated by you. We strictly comply with applicable laws and regulations and will retain, use, disclose, or otherwise process relevant data solely for the purpose of enabling product functionality, or as permitted by applicable laws. Before interacting with any third party regarding data exchange, please be informed of and understand their privacy and data security policies.

Disclaimer

a) We shall not be liable for any damages resulting from failure to comply with applicable operational or design specifications.
b) We shall bear no liability for any inaccuracies or omissions in this document, nor for any damages arising from the use of the information contained herein.
c) While we make every effort to ensure the integrity, accuracy, and timeliness of the features and functions under development, errors or omissions may nevertheless occur. Unless otherwise provided in a valid written agreement, we make no warranties of any kind, express, implied, or statutory, and disclaim all liability for any loss or damage arising from the use of any features or functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage is foreseeable.
d) We assume no legal responsibility for the accessibility, safety, accuracy, availability, legality, or completeness of any information, content, advertising, commercial offers, products, services, or materials on third-party websites or third-party resources.

About the Document

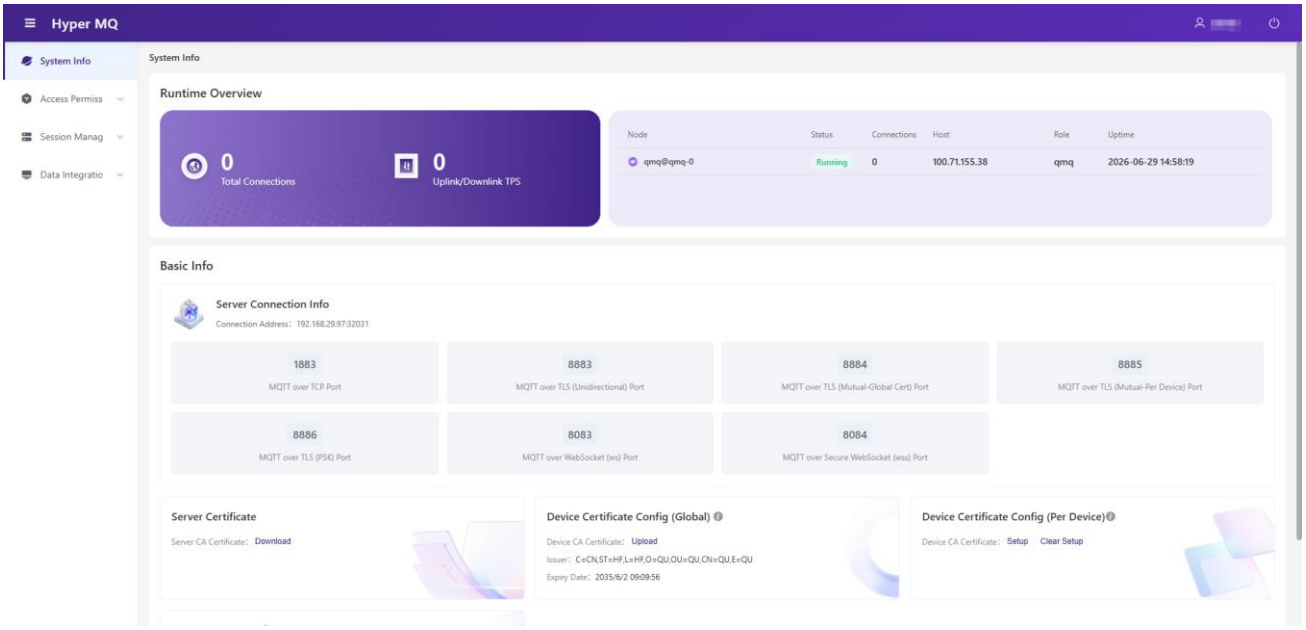
Revision History

Version	Date	Author	Description
-	2026-04-15	Reuben Guo	Creation of the document
1.0	2026-07-15	Reuben Guo	First official release

Contents

Table Index

1 System Info



1.1. Runtime Overview

The "System Info" page serves as the dashboard of the console, displaying the real-time running status of the Broker.

Core metrics at the top of the page:

- **Total Connections**: The total number of devices currently connected to the Broker.
- **Uplink/Downlink TPS**: Current message throughput (transactions per second).

The node list is displayed on the right side, containing the following fields:

Table 1: Node List

Field	Description
Node	The identifier of the node within the cluster.

Status	The current running status of the node (Running/Stopped).
Connections	The total number of devices currently connected to the Broker.
Host	The IP address of the node server.
Role	The role of the node within the cluster.
Uptime	The time when the node was last started.

1.2. Basic Info

The "Basic Info" section displays the Broker's server connection addresses and supported protocol ports.

Table 2: Supported Protocol Ports

Port	Protocol	Description
1883	MQTT over TCP Port	Plaintext transmission, for development and test environments.
8883	MQTT over TLS (Unidirectional) Port	Unidirectional TLS encryption (server certificate only).
8884	MQTT over TLS (Mutual-Global Cert) Port	Mutual TLS authentication (global device certificate).
8885	MQTT over TLS (Mutual-Per Device) Port	Mutual TLS authentication (unique-certificate-per-device authentication).
8886	MQTT over TLS (PSK) Port	PSK encrypted communication.
8083	MQTT over WebSocket (ws) Port	Plaintext WebSocket access.
8084	MQTT over Secure WebSocket (wss) Port	Encrypted WebSocket access.

NOTE

Connection addresses and port numbers are displayed according to actual server configurations.

1.3. Server Certificate

Server certificates are configured during service deployment. The section provides a function to download the CA certificate.

Devices require this CA certificate to verify the server's identity when TLS encrypted connections are established.

1.4. Device Certificate Config (Global)

The platform supports mutual TLS authentication where all devices use certificates issued by a single shared device CA certificate.

You need to upload the CA file used to issue device certificates. Both DER and PEM formats are supported.

NOTE

After a certificate is replaced, already connected devices will not be affected. Newly connected devices will be verified using the updated certificate.

1.5. Device Certificate Config (Per Device)

The platform supports mutual TLS authentication with an independent unique certificate for each individual device.

This feature requires configuring an authentication callback endpoint. When a device establishes a TLS connection via the unique-certificate-per-device authentication port, its certificate information will be sent to your business backend for authentication validation.

Table 3: Comparison with Global Certificates

Comparison Item	Global Certificate	Unique-certificate-per-device Authentication
Certificate Quantity	All devices share one single CA certificate.	Separate device certificates are issued by different CAs, while the binding relationship between certificate fingerprints and devices can be verified.

Security Level	Medium	High (leakage of one device certificate does not impact other devices).
Management Complexity	Low	Relatively high (manage individual device certificates for business systems).
Applicable Scenarios	Scenarios with moderate security requirements	Scenarios with extremely high security requirements.
Access Port	8884 (subject to actual system display)	8885 (subject to actual system display).

Table 4: Callback Endpoint Configuration

Item	Required	Description
URL	Yes	Start with http:// or https://.
Request Method	Yes	*GET* or *POST*
Data Format	Yes	*JSON* or *URLEncode*
Timeout	Yes	Request timeout duration.
Headers	No	Custom request headers (e.g., API authentication tokens).
Params	No	Custom request parameters with built-in variables available. ● *${deviceCert}*: Device X.509 certificate AKI ● *${issuer}*: Device X.509 certificate issuer

1.6. Server PSK Config

The platform supports TLS connections via PSK encryption for resource-constrained devices.

You can configure globally valid PSK IDs and PSK Secrets on the platform, or set up a PSK Webhook for the business system to validate PSK information.

- **PSK Global Config**
  - ➢ PSK ID supports up to 128 characters.
  - ➢ PSK Secret length ranges from 128 bits to 2048 bits.
  - ➢ A maximum of 100 PSK entries can be configured.

- **PSK Webhook**

Table 5: PSK Webhook Config

Item	Required	Description
URL	Yes	Start with http:// or https://.
Request Method	Yes	*GET* or *POST*
Data Format	Yes	*JSON* or *URLEncode*
Timeout	Yes	Request timeout duration.
Headers	No	Custom request headers (e.g., API authentication tokens).
Params	No	Custom request parameters with built-in variables available. *${pskId}*: The PSK ID of the device.

2 Access Permission

2.1. Login Auth

The "Login Auth" feature is used to manage device/client login rules to control which devices are allowed to connect to the Broker.

- **Method 1**: Add client usernames and passwords to the whitelist, suitable for device debugging.
- **Method 2**: Add login authentication rules. Device login requests will be forwarded to your business backend for connection approval judgment and you need to implement corresponding validation logic.

2.1.1. Login Authentication Rules

The "Login Authentication Rules" page displays all configured login authentication rules, where you may add, edit, enable, disable and delete rules.

2.1.2. Add/Edit Auth Rule

Table 6: HTTP Authentication Rule Configuration

Item	Required	Description
Rule Name	Yes	Maximum length: 20 characters. Letters, numbers, and underscores are supported.
Priority	Yes	A positive integer. Larger values represent higher priority.
Auth Method	Yes	*HTTP*
Filter Condition	Yes	The rule action will only be executed when the selected parameter matches the regex. If multiple filter conditions are specified, all conditions must be satisfied simultaneously. A filter condition consists of three parts: the key to filter, a regex, and variables to extract.

		Supported filter keys for login authentication: ● *clientid*: Client ID ● *password*: Connection password ● *username*: Connection username ● *sockport*: Connection port ● *session_id*: Session ID Regex example: *qd(.{6})(.*)* Variable extraction example: *pk,dk*
URL	Yes	Start with http:// or https://.
Request Method	Yes	*GET* or *POST*
Data Format	Yes	*JSON* or *URLEncode*
Timeout	Yes	Request timeout duration.
Headers	No	Custom request headers (e.g., API authentication tokens).
Params	No	Custom request parameters with built-in variables available. Built-in variables: ● *${clientid}*: Client ID ● *${username}*: Username ● *${password}*: Password ● *${password_b64}*: Base64-encoded password ● *${rootFinger}*: Root certificate fingerprint ● *${deviceFinger}*: Device certificate fingerprint ● *${authenticated}*: Authentication status

Table 7: Redis Authentication Rule Configuration

Item	Required	Description
Rule Name	Yes	Maximum length: 20 characters. Letters, numbers, and underscores are supported.
Priority	Yes	A positive integer. Larger values represent higher priority.
Auth Method	Yes	*Redis*
Filter Condition	Yes	The rule action will only be executed when the selected parameter matches the regex. If multiple filter conditions are specified, all conditions must be satisfied simultaneously. A filter condition consists of three parts: the key to filter, a regex, and

		variables to extract. Supported filter keys for login authentication: ● *clientid*: Client ID ● *password*: Connection password ● *username*: Connection username ● *sockport*: Connection port ● *session_id*: Session ID Regex example: *qd(.{6})(.*)* Variable extraction example: *pk,dk*
Auth Policy	Yes	The client is allowed to log in only if all auth policies are satisfied. You need to configure the Redis command, comparison content, and encryption type for the Redis response. Redis command example: Use the HGET command. Specify the Redis key, e.g. *device:$:$* Comparison example: You can use built-in variables or enter custom values such as *true/false*. Supported encryption types for Redis response: ● *plain* (Compare directly), ● *base64* (Compare after base64 encoding), ● *md5* (Compare after MD5 hashing)
Post-Check Action	Yes	● *Stop authentication chain* ● *Proceed to next priority rule on failure* ● *Proceed to next priority rule on success*
redis_host	No	Hostname or IP address of the Redis server.
redis_db	No	Index number of the target Redis database.
redis_password	No	Password for connecting to Redis server.

2.1.3. Static Client Whitelist

If no login authentication rule is matched, the system will check the username and password against the whitelist. If a match is found, the device is allowed to log in. This feature is designed for rapid client setup during debugging. A maximum of 100 entries is supported.

2.2. Topic Auth

The "Topic Auth" feature is used to manage client "publish" and "subscribe" permissions for MQTT Topics, controlling which topics devices may use to send and receive data.

2.2.1. Pub/Sub Rules

The "Pub/Sub Rules" page displays all configured ACL rules, including rule name, priority, ACL auth method, and filter conditions. You may add, edit, enable, disable, and delete rules.

2.2.2. Add/Edit Pub/Sub Rule

Table 8: Local Publish/Subscribe Rule Configuration

Item	Required	Description
Rule Name	Yes	Maximum length: 20 characters. Letters, numbers, and underscores are supported.
Priority	Yes	A positive integer. Larger values represent higher priority.
ACL Auth Method	Yes	*Local Config*
Filter Condition	Yes	The rule action will only be executed when the selected parameter matches the regex. If multiple filter conditions are specified, all conditions must be satisfied simultaneously. A filter condition consists of three parts: the key to filter, a regex, and variables to extract. Supported filter keys for topic ACL: ● *clientid*: Client ID ● *username*: Connection username ● *topic*: MQTT topic Regex example: *qd(.{6})(.*)* Variable extraction example: *pk,dk*
Subscribable Topic	Yes	Configure the topic range that clients can subscribe to. Topics support built-in variables *clientid, username* and custom variables from filter conditions. Topics support wildcards "+", "#", "*". Wildcards "+" and "#" can

		only exist as independent levels, and "#" and "*" can only be the ending level. Example: *tsl/${clientid}/downlink*
Publishable Topic	Yes	Configure the topic range that clients can publish to. Topics support built-in variables *clientid*, *username* and custom variables from filter conditions. Topics support wildcards "+", "#", "*". Wildcards "+" and "#" can only exist as independent levels, and "#" and "*" can only be the ending level. Example: *tsl/${clientid}/uplink*

Table 9: HTTP Publish/Subscribe Rule Configuration

Item	Required	Description
Rule Name	Yes	Maximum length: 20 characters. Letters, numbers, and underscores are supported.
Priority	Yes	A positive integer. Larger values represent higher priority.
ACL Auth Method	Yes	*HTTP*
Filter Condition	Yes	The rule action will only be executed when the selected parameter matches the regex. If multiple filter conditions are specified, all conditions must be satisfied simultaneously. A filter condition consists of three parts: the key to filter, a regex, and variables to extract. Supported filter keys for topic ACL: ● *clientid*: Client ID ● *username*: Connection username ● *topic*: MQTT topic Regex example: *qd(.{6})(.*)* Variable extraction example: *pk,dk*
Scope	Yes	● *Pub & Sub* ● *Publish* ● *Subscribe*
Cache Expiry	Yes	Cache expiration timeout duration.

Cache Key	Yes	Caching is used to reduce the number of HTTP calls. In addition to constants, variables extracted from filter conditions can also be entered.
URL	Yes	Start with http:// or https://.
Request Method	Yes	*GET* or *POST*
Data Format	Yes	*JSON* or *URLEncode*
Timeout	Yes	Request timeout duration.
Headers	No	Custom request headers (e.g., API authentication tokens).
Params	No	Custom request parameters with built-in variables available. Built-in variables: ● *clientid*: Client ID ● *username*: Connection username ● *topic*: MQTT topic

Table 10: Redis Publish/Subscribe Rule Configuration

Item	Required	Description
Rule Name	Yes	Maximum length: 20 characters. Letters, numbers, and underscores are supported.
Priority	Yes	A positive integer. Larger values represent higher priority.
ACL Auth Method	Yes	*Redis*
Filter Condition	Yes	The rule action will only be executed when the selected parameter matches the regex. If multiple filter conditions are specified, all conditions must be satisfied simultaneously. A filter condition consists of three parts: the key to filter, a regex, and variables to extract. Supported filter keys for topic ACL: ● *clientid*: Client ID ● *username*: Connection username ● *topic*: MQTT topic Regex example: *qd(.{6})(.*)* Variable extraction example: *pk,dk*
Subscribable	Yes	Use GET command to retrieve the subscribable Topic list for this client

Topic		from Redis. Enter the Redis cmd command. The cmd uses the GET command, and you need to enter the key in Redis, e.g. *mqtt_acl_sub:${pk}:${dk}*
Publishable Topic	Yes	Use GET command to retrieve the publishable Topic list for this client from Redis. Enter the Redis cmd command. The cmd uses the GET command, and you need to enter the key in Redis, e.g. *mqtt_acl_pub:${pk}:${dk}*.
redis_host	No	Hostname or IP address of the Redis server.
redis_db	No	Index number of the target Redis database.
redis_password	No	Password for connecting to Redis server.

Table 11: Wildcard Reference

Wildcard	Name	Description	Example
+	Single-level Match	Match a single topic level; must occupy an independent level	sensor/+/data matches sensor/temp/data
#	Multi-level Match	Match multiple topic levels; must be placed as the final independent level	sensor/# matches sensor/temp/data/room1
*	Prefix Match	Match topics starting with the specified strings; can only be used at the final level.	sensor/te* matches sensor/temp, sensor/temperature

2.3. Session Management

The "Session Management" feature enables you to view and manage all MQTT sessions currently connected to the Broker in real time.

2.3.1. Session List

Table 12: Session List

Field	Description
ClientID	Unique client identifier.
Username	Username used for client connection.
IP	IP address of the client.
Keepalive	Heartbeat interval (seconds).
Protocol	MQTT protocol version.
Connection Status	Current connection status.
Connection Time	Timestamp when the connection was established.

NOTE

1. You can search for a session by *ClientID* or *Username*.
2. Deleting a session will force the corresponding client offline.

2.3.2. Session Details

The session detail page contains three sections:

- Client Info: Display ClientID, Username, connection status, IP address, Keepalive, MQTT version, connection time, and disconnection time.
- Subscription Info: List all topics subscribed by the client with corresponding QoS levels.
- Subscription Management: Add or remove subscriptions (QoS 0 and QoS 1 available).

NOTE

Non-persistent sessions (*Clean Session* = *true*) are automatically cleared after client disconnection.

3 Data Integration

3.1. Forward to Kafka

The "Forward to Kafka" feature is used to forward device uplink messages to designated Kafka clusters per defined rules for data integration with upstream business systems.

3.1.1. Forwarding Rules

The "Forwarding Rules" page displays all configured Kafka forwarding rules with rule name, priority and filter conditions. You may add, edit, enable, disable and delete rules.

3.1.2. Add/Edit Forwarding Rule

Table 13: Forwarding Rule Configuration

Item	Required	Description
Rule Name	Yes	Unique identifier for the rule.
Priority	Yes	A positive integer. Larger values represent higher priority.
Filter Condition	Yes	The rule action will only be executed when the selected parameter matches the regex. If multiple filter conditions are specified, all conditions must be satisfied simultaneously. A filter condition consists of three parts: the key to filter, a regex, and variables to extract (optional). Supported filter keys for message forwarding: *clientid*, *topic*. Regex example: *qd(.{6})(.*),\\$event/online/.+,qd/.+* Variable extraction example: *pk*,*dk* When filtering by topic, in addition to device uplink topics, the

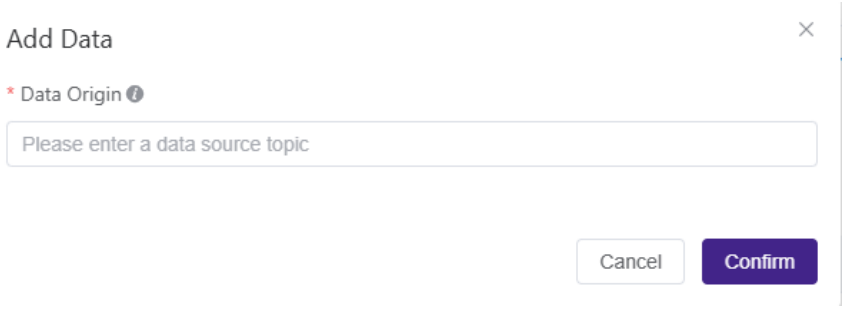
		system provides the following built-in topic events: ● Device online: *$event/online/.+* ● Device offline: *$event/offline/.+* ● Failure event: *$event/fail/.+* ● Subscription event: *$event/subscribe/.+*
Target Kafka Topic	Yes	Destination Kafka topic name.
Target Kafka Key	No	The key for the Kafka message, used for message partitioning.
Extended Info	No	Extended fields will be placed in the *extend* field of the Kafka message.
payload_format	No	Message payload format.
host	Yes	Kafka address.

3.2. Message Forwarding

The "Message Forwarding" feature is used to forward device-reported data to other message middleware or databases for storage. You may edit data processing scripts to convert raw payloads into target formats before delivering data to downstream destinations.

3.2.1. Data Source

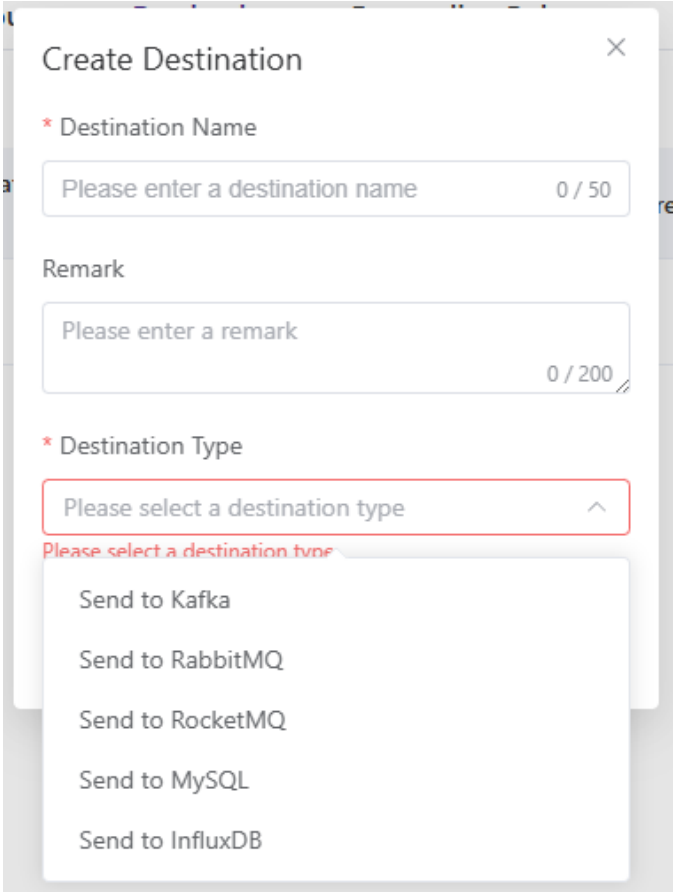
1. Navigate to "**Message Forwarding**" → "**Data Source**" and create a data source.
2. Fill in the source name and remark its usage purpose.
3. Click "**View**" in the list to open the data source management page.
4. Click "**Add**" to input detailed data origin.



5. Input the source topic name. Allowed characters: letters, digits and special characters _ / # +. / is used to separate topic levels. # and + must occupy independent levels, and # can only be placed as the final level.
6. One data source may be bound to multiple forwarding rules. Unbind all associated rules before deleting a data source.

3.2.2. Destination

1. Navigate to "**Message Forwarding**" → "**Destination**" and create a destination.
2. Fill in the destination name and remark its usage purpose. Supported destination types: Kafka, RabbitMQ, RocketMQ, MySQL, InfluxDB, and OpenGemini.



3. The system sends data to the selected destination as a client. Therefore, the required client configuration parameters vary depending on the destination type. Fill them in according to actual requirements.
4. Run "**Connectivity Test**" after completing destination configurations. Invalid destination addresses will prevent rule activation.
5. One destination may be bound to multiple forwarding rules. Unbind all associated rules before deleting a destination.

3.2.3. Forwarding Rule

1. Navigate to "**Message Forwarding**" → "**Forwarding Rule**" and create a rule.
2. Fill in the rule name and remark its usage purpose.
3. Click "**View**" in the list to open the rule management page.
4. Bind one data source to the rule to define which device data flows into the processing script (one source per rule).

5. Bind target destinations available for data delivery within the processing script.
6. Use the built-in functions provided by the system to write a JavaScript script. You can process raw input data before forwarding it to different destinations. For detailed scripting guidelines, refer to the *Message Forwarding Script Development Guide*.
7. After writing the script, you must complete debugging to ensure it executes successfully and takes less than 100ms. After the script passes debugging, you can publish the script. The script will only be loaded and executed after publishing.
8. Set the rule status to "**Enabled**" after all configurations are finished.

3.3. Token Management

The "Token Management" feature is used to manage authentication tokens required for calling REST APIs.

A maximum of 3 valid tokens may exist simultaneously. Please store tokens securely to avoid leakage.

4 My Account

4.1. Personal Info

You can view your current account username on the "Personal Info" page.

4.2. Change Password

You can update your login password on the "Change Password" page.