Alibaba Cloud
MQTT

Authorization and Authentication

Issue: 20190914

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use**

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified , reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates . The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd / d C :/ windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

1 Authentication overview

This topic describes the principle behind and the types of authentication of MQTT clients.

Authentication principle

When an MQTT client sends and receives messages, the MQTT broker authenticates the MQTT client based on the Username and Password parameters. The definitions of the Username and Password parameters vary depending on different scenarios of permission verification. A proper authentication mode must be selected during MQTT client development based on the actual situation, and the Username and Password parameters must be set correctly based on the corresponding specifications.

Authentication modes

Mode	Description	Scenarios
Signature	Signature verification	Permanent authorization, applicable to secure and trusted clients
Token	On-demand token permission verification	On-demand authorization, applicable to untrusted clients

Username

The Username parameter consists of the authentication mode, AccessKeyId, and InstanceId, which are separated by vertical bars (|).

Example: If MQTT client GID_Test@@@0001 uses InstanceId mqtt-xxxxx and AccessKey YYYYY, the Username parameter for connecting this MQTT client must be set to Signature|YYYYY|mqtt-xxxxx in signature authentication mode and to Token|YYYYY|mqtt-xxxxx in token authentication mode.

Password

Permission verification mode	Password	Scenarios
Signature verification	Client ID signature encoded in Base64. For the setting method, see the document about signature calculation.	Permanent authorization, applicable to secure and trusted clients
On-demand token permission verification	Uploaded token content. For the setting method, see the document about token calculation.	On-demand authorization, applicable to untrusted clients

Types and comparison of authentication modes

Currently, MQ for MQTT supports signature authentication and token authentication, which correspond to fixed permissions and non-fixed on-demand permissions, respectively. The two modes and scenarios are detailed as follows.

Signature authentication (fixed permissions)

The signature authentication mode is the default authentication mode recommended by MQ for MQTT. In this mode, MQTT clients of the same type are assigned the same permissions and calculate signatures by using the same account. The MQTT broker identifies the accounts and permissions of the MQTT clients through signature comparison. The signature authentication mode is easy to understand and use.

- Scenarios

  The used MQTT clients can be logically classified into the same type, belong to the same account, and have the same permissions. The runtime environment of each MQTT client is relatively controllable. You do not have to worry about malicious attacks against devices, such as cracking and stealing.

  From the service perspective, MQTT clients of the same type are managed by the same account, which can be a primary account or a sub-account. Therefore, the MQTT clients only need to calculate signatures based on the corresponding AccessKeySecret. Permissions are managed in the console by the account administrator.

- Signature calculation

  Signatures are calculated by using each MQTT client's independent information to prevent signature stealing. MQ for MQTT requires each MQTT client to use its unique client ID as the to-be-signed content. For the signature calculation method, see .

Token authentication (on-demand permissions)

The token authentication mode supports access through on-demand credentials and is applicable in the scenario where the permissions of each MQTT client need to be classified with a small granularity or MQTT clients need to be assigned only on-demand permissions with a limited validity period. The token service allows you to set the resources that are accessed by a single MQTT client, the permission level that is assigned to the MQTT client, and the permission expiration time.

For more information about the process and precautions of token authentication, see #unique_4.

- Scenarios

  The service party has an independent local account system and needs to split the identities of Alibaba Cloud accounts for the second time, or even needs to assign an independent account and permissions to each MQTT client. In this case, the RAM user system of Alibaba Cloud cannot meet the requirement for refined classification of MQTT clients.

  Though the MQTT clients belong to the same Alibaba Cloud account, they need to play different roles assigned by local accounts (owned by the service department or a single MQTT client). This requirement cannot be met by using Alibaba Cloud accounts in signature authentication mode. The use of fixed permissions cannot meet the requirements of mobile terminals that need to protect against cracking and hijacking. Use of only on-demand and non-fixed permissions makes it more flexible to control permissions on a per-client basis and assign permissions on a per-resource basis.
- Token usage

  The token authentication mode is relatively complex. The service party needs to have the account (device) management capability, manage the permissions and permission validity period of each device, apply for tokens on secure and

controllable application servers and authorization servers, and deliver the tokens to MQTT clients. For usage instructions, see the document about the token service.

2 Signature authentication

This topic describes how to calculate a signature and create a signature in the console when signature authentication is used by MQ for MQTT.

Use the MQTT SDK to access the MQTT broker

If signature verification is used, the Connect message that the MQTT client sends for connecting to the MQTT broker must set the Username and Password parameters based on the specifications described here. The setting and calculation methods are as follows:

Username

The Username parameter consists of the authentication mode, AccessKeyId, and InstanceId, which are separated by vertical bar (|). The authentication mode is set to Signature in signature authentication mode.

Example: If MQTT client GID_Test@@@0001 uses InstanceId mqtt-xxxxx and AccessKeyId YYYYY, the Username parameter for connecting this MQTT client must be set to Signature|YYYYY|mqtt-xxxxx in signature authentication mode.

Password

The Password parameter indicates the result of client ID signing. The calculation method is as follows:

For example, a client ID is GID_AAA@@@BBB001.

Calculate the signature of the to-be-signed string by using AccesKeySecret and HMAC SHA-1 to obtain a binary array. Encode the binary array in Base64 to obtain the final signature string Password, that is, eqweq+adwe23fssf.

Each language provides a function library to implement the HMAC algorithm. You can also refer to the demo project.

Verify the signature in the console

The MQ for MQTT console provides the signature calculation tool so that you can compare and check whether your signature calculation is correct.

In the left-side navigation pane of the MQ for MQTT console, choose Signature Verification, and enter the AccessKeyId, AccessKeySecret, and Client ID of the

application-used account to obtain the Username and Password parameters that need to be set in the application.

Note:

The tool uses only frontend JavaScript of the web browser for calculation and does not transmit AccessKeySecret to MQ, removing the risk of AccessKeySecret disclosure. In the actual situation, the tool is only used by the console for troubleshooting and data comparison.

Calculate the signature on the MQTT client. Alternatively, calculate on the MQTT broker and then send the result to the MQTT client for security purposes.

3 Token authentication

3.1 Token authentication overview

Message Queue for MQTT provides MQTT clients with temporary access permissions with a limited validity period through token authentication. The MQTT token service issues temporary credentials to users that are managed by the local account system, and limits the users' access permission. This implements refined permissions control on a per-client and per-resource basis.

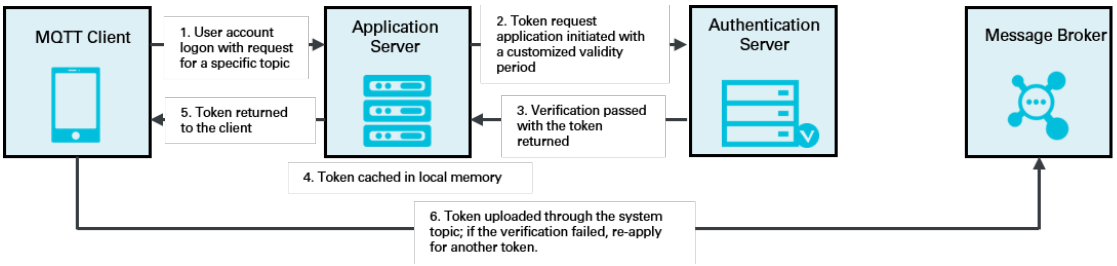
As the administrator of the local account, the user applies for a temporary access credential (Token) for the actual client, and the client uses the temporary credential for actual business access. Users who manage local accounts can apply for temporary credentials (tokens) for clients. Then, the clients use temporary credentials to access services. A token represents the temporary identity of an MQTT client and is assigned permissions, resource lists, and access types by a service administrator. This implements refined permissions control on a per-client and per-resource basis.

Term	Definition	Description
Token	Temporary credential	Message Queue for MQTT issues the temporary credential to assign a single MQTT client the permission to access specific resources.
AppServer	Application server	The server that enables users to manage local accounts and applies for and manages token services on behalf of clients
AuthServer	Authentication Server	Message Queue for MQTT authorization server that processes the token-related requests that are initiated by the application server

Procedure

The token authentication mode is more complex than the signature authentication mode. You must deploy your application server based on the process shown in the following figure, and ensure that the MQTT client is initialized with the capability to interact with the application server to obtain and update tokens.

Figure 3-1: Authentication process



The procedure is as follows:

1. The MQTT client connects to the application server for authentication upon startup.
2. The MQTT client applies for the required permissions on all the topics from the application server.
3. The application server verifies whether the MQTT client is authorized to operate the topics. If the MQTT client is authorized, the application server applies for the resource-related tokens from the authorization server.
4. Message Queue for MQTT authorization server verifies the token application request and returns the corresponding tokens if the request is valid.
5. The MQTT client performs local persistence of the tokens that are returned by the application server, and maps required permissions to the tokens. Tokens are cached for the following two purposes:
   - When the MQTT client is restarted with the same permissions for access, the application server returns the cached tokens to the MQTT client. This avoids repeated token application.
   - If the authorization server cannot process the client token application request due to an error, the application server returns the previously applied token for local disaster recovery.

6. The MQTT client sets token parameters based on the specifications to connect to the MQTT broker. The MQTT client can send and receive messages after being authenticated by the MQTT broker.
7. The MQTT client sends and receives messages properly. If the MQTT broker determines that the token has expired, it triggers an authentication failure and disconnects the MQTT client. In this case, the MQTT client needs to re-apply for a token.

Client behavior constraints

- The MQTT client must set the token as a connection parameter in Password and upload the token when establishing a connection.
- The MQTT client must know the validity period of the used token and ensure that it is within the validity period. If the token expires, the MQTT broker may disconnect the MQTT client.
- The MQTT client can listen to the token expiration notifications that are pushed by the MQTT broker, but the MQTT broker does not guarantee that the push is always triggered. Those notifications are only for troubleshooting.
- The MQTT client must perform persistence of the tokens that are returned by the application server to avoid applying for the same token during each reconnection . Otherwise, the application server may crash when receiving connection requests from many MQTT clients at the same time.
- The MQTT client can update a token in two ways. One way is to disconnect the MQTT client first and reinitialize the connection using the new token. The other way is to use the MQTT-provided dynamic token update function through system topics. If the token is updated dynamically, the MQTT client must ensure that the local configuration is also updated to avoid that the old token is used during the next connection initialization.

Application server behavior constraints

- The application server must authenticate the MQTT client to prevent the MQTT client from applying for a token by using a forged identity.
- The application server must manage the token-client relationship to prevent the same MQTT client from calling a token repeatedly.
- The application server must implement local disaster recovery to prevent service congestion due to temporary failed access to the authorization server.

Related APIs

You can call related APIs to implement token authentication.

The application server is responsible for token application and revocation, and interacts with the authorization server by using HTTPS APIs. Each API requires authentication by using an AccessKey and a request signature. Currently, the APIs for token application, revocation, and verification are available.

Message Queue for MQTT clients provide three APIs for dynamic token update, listening to token expiration notifications, and listening to token invalidity notifications, respectively.

Token service addresses

Currently, the token service supports public cloud regions in Mainland China and the Singapore region. Finance Cloud is not supported at present. The specific service addresses are as follows:

Service region	Token server address
China (Beijing)	Mqauth.aliyuncs.com
Internet	Mqauth.aliyuncs.com
China (Hangzhou)	Mqauth.aliyuncs.com
China (Shanghai)	Mqauth.aliyuncs.com
China (Shenzhen)	Mqauth.aliyuncs.com
Singapore	Mqauth.ap-southeast-1.aliyuncs.com

3.2 Application server-related methods in token authentication

3.2.1 Specifications on token-related methods for the application server

Your application server must set parameters and obtain return values based on the following specifications when calling methods of the authorization server. Otherwise, the call may fail.

Signature calculation

When the application server calls a method, for example, the token application method, the authorization server verifies the call parameters and filters invalid

requests. The application server must sort and concatenate request parameters by using the specified method based on the following rules to form the to-be-signed string, and then encrypt the string by using the AccessKeySecret to obtain a signature .

> **Note:**
> The signature calculation rules use logic that is different from the signature verification logic of the MQTT client.

The details are as follows:

- Parameter pairs are in the format key=value.
- Parameter pairs are separated by ampersands (&).
- Parameter pairs are sorted in the alphabetic order of keys during signature calculation.
- If a parameter pair contains multiple values of the same key, that is, same-name parameters, they are sorted in alphabetic order and connected by commas (,).

Examples

Assume that the original HTTP method of a request is as follows:

```
https :// mqauth . aliyuncs . com / token / apply
```

The parameters are as follows:

```
parama = a
paramc = c2 , c1
paramb = b2 , b1 , b3
```

Sort the parameters by key as follows:

```
parama = a
paramb = b2 , b1 , b3
paramc = c2 , c1
```

Sort the values of the same parameter in the alphabetic order. For example, sort "b2, b1,b3" as follows:

```
parama = a
paramb = b1 , b2 , b3
paramc = c1 , c2
```

The concatenated string to be signed is as follows:

```
parama = a & paramb = b1 , b2 , b3 & paramc = c1 , c2
```

Calculate the signature by using the AccessKeySecret and the HMAC SHA-1 algorithm.

Note:

- HTTP or HTTPS can be selected for HTTP-based token management.
- The HTTP methods are GET and POST. POST is recommended in the production environment, and GET can be used in the test environment.

Response status codes

The authorization server properly processes the received token request and returns a response in the JSON string format to the caller. The caller can retrieve specific values from the string as needed.

The JSON string has the following key-value mapping:

Key	Type	Description
success	Boolean	Indicates whether the call is successful.
message	String	The processing result or error description.
code	Integer	The response code, which indicates the request processing result or error type.
tokenData	String	The token string, which is returned for token application.

The following table lists the error codes.

Code	Description
200	This code is returned when the call is successful.
400	This code is returned when an error occurs during parameter verification.
407	This code is returned when an error occurs during signature calculation.
409	This code is returned when an error occurs during token creation.

Code	Description
410	This code is returned when the token revocation failed.
1	This code is returned when the token is forged and cannot be parsed.
2	This code is returned when the token has expired.
3	This code is returned when the token has been revoked.

3.2.2 Apply for tokens

Method

https://{tokenServerUrl}/token/apply

Scenario

After verifying the permissions of an MQTT client, the application server calls this method to request a token for this client from the MQTT authorization server.

Request parameters

Name	Type	Description
actions	String	The token permission type , which can be R (read), W (write), or R,W (read and write). **Note:** To grant both the read and write permissions, enter R and W and separate them with a comma (,).
resources	String	The resource name that indicates an MQTT topic . Multiple topics are separated by commas (,). Each token can run and operate up to 100 resources. Multiple topics are sorted in the alphabetic order.

Name	Type	Description
accessKey	String	The AccessKeyId of the account used in the current request.
expireTime	long	The token expiration time, in the timestamp format and precise to milliseconds. The minimum expiration interval is 60 seconds.
proxyType	String	The token type, which is set to MQTT here. Set this parameter based on the service that is used.
serviceName	String	Set this parameter to mq. Other values are invalid.
instanceId	String	The ID of the MQTT instance, which must match the client-used instance ID.
signature	String	The signature string. Signature calculation is required by the following fields in the current request: actions, resources, expireTime, serviceName, and instanceId.

3.2.3 Verify tokens

Method

https://{tokenServerUrl}/token/query

Scenario

This method is called by the application server to check whether a single token is valid.

Request parameters

Name	Type	Description
token	String	The token that you want to verify.
accessKey	String	The AccessKeyId of the account used in the current request.
signature	String	The signature string. The Token field in the current request requires signature calculation.

3.2.4 Revoke tokens

Method

https://{tokenServerUrl}/token/revoke

Scenario

This method is called by the application server to terminate previous token authorization in advance.

Request parameters

Name	Type	Description
token	String	The token to be revoked.
accessKey	String	The AccessKeyId of the account used in the current request.
signature	String	The signature string. The Token field in the current request requires signature calculation.

3.3 Client-related methods in token authentication

This topic describes how an MQTT client uploads and updates tokens, and how it listens to notifications about token expiration and invalidity when token authentication is used.

Set parameters for MQTT client connection in token authentication mode

MQTT supports three token types. Each MQTT client can apply for one token per type at most, and may apply for and use one or more types as needed. The following table lists the token types.

Type identifier	Description
R	A read-only token that has only the read permission for the specified resources
W	A write-only token that has only the write permission for the specified resources
RW	A read-write token that has the read and write permissions for the specified resources

Parameter settings of MQTT client connection in token authentication mode:

Username: consists of the authentication mode, AccessKeyId, and InstanceId, which are separated by vertical bars (|). The authentication mode is Token in token authentication mode.

Example: If MQTT client GID_Test@@@0001 uses InstanceId mqtt-xxxxx and AccessKeyId YYYYY, the Username parameter for connecting this MQTT client must be set to Token|YYYYY|mqtt-xxxxx in token authentication mode.

Password: the content of the token to be used by the MQTT client. The parameter value is a complete string that consists of all the tokens concatenated in sequence by token type and token content that are connected by vertical bars (|). Different types of tokens are concatenated in no particular order.

Example 1: If an MQTT client has only the read-only token 123 in the string format, then Password must be set to R|123.

Example 2: If an MQTT client has two token types (123 as the read-only token and abcd as the write-only token), then Password must be set to R|123|W|abcd.

 Note:

Ensure that the token parameters of MQTT client connection are set based on the preceding rules and that all the tokens are valid. If some tokens are invalid, the MQTT broker may determine that the token settings are invalid.

Update tokens for clients

In normal cases, to update the token of an MQTT client, disconnect the MQTT client first and then connect the MQTT client using the new token. If you want to avoid client disconnection during token update, you can replace the token data of the MQTT broker session through dynamic token update.

During dynamic token update, an MQTT client sends a special message by using a predefined system topic to update the new token content to the MQTT broker. The MQTT client must ensure that the local token configuration is updated along with the dynamic token update. Otherwise, the old token data may still be used during the next connection initialization.

Topic sent for token update: $SYS/uploadToken

Content: JSONString

Content description:

Name	Type	Description
Token	String	The MQTT client must upload the token string if token authentication is used.
type	String	The token type, which can be W, R, or RW, corresponding to three permission types. An MQTT client can have the three types of tokens at most. An incorrect token type may result in a permission verification error.

Response

A common PubAck message is returned. The MQTT client must wait for the response before performing the publish or subscribe operation. Otherwise, the MQTT broker may still use the old token data for authentication, causing an authentication failure and a disconnection.

Listen to notifications about imminent token expiration (subscription not required)

To facilitate service debugging and monitoring, the MQTT broker pushes notifications to the MQTT client by using system topics to notify the client of tokens that will soon expire. The MQTT client can listen to such notifications to detect whether tokens are about to expire.

Received topic: $SYS/tokenExpireNotice

Content: JSONString

Content description:

Name	Type	Description
expireTime	Long	The token expiration time, in the timestamp format and precise to milliseconds. The MQTT client is typically notified of the token expiration time once five minutes in advance. However, the MQTT broker does not guarantee that a notification is always sent.
type	String	The client-uploaded token type, which can be W, R, or RW, corresponding to three permission types.

Response:

After receiving a notification about imminent token expiration, the MQTT client needs to apply for a new token as soon as possible to avoid failed transmission of messages.

Listen to notifications about invalid tokens (subscription not required)

To facilitate service debugging and monitoring, the MQTT broker pushes notifications to the MQTT client by using system topics to notify the client of token authentication errors. The MQTT client can listen to such notifications to detect token mismatch and other permission errors.

Received topic: $SYS/tokenInvalidNotice

Content: JSONString

Content description:

Name	Type	Description
code	Integer	The type of a token verification error.
type	String	The client-uploaded token type, which can be W, R, or RW, corresponding to three permission types.

Response:

Authentication may fail when the MQTT broker encounters a token verification error, and then the MQTT broker proactively disconnects the MQTT client. The MQTT broker pushes an error code to the MQTT client before disconnecting the MQTT client. The MQTT client can identify the cause based on the error code.

Type code	Error type
1	The token is forged and cannot be parsed.
2	The token has expired.
3	The token has been revoked.
4	The resource and token do not match.
5	The permission type and token do not match.
8	The signature is invalid.
-1	The account permission is invalid.