

Simple Message Notification

User Guide

Issue 07
Date 2023-06-30



Copyright © Huawei Technologies Co., Ltd. 2024. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:

<https://www.huawei.com/en/psirt/vul-response-process>

For vulnerability information, enterprise customers can visit the following web page:

<https://securitybulletin.huawei.com/enterprise/en/security-advisory>

Contents

1 Topic Management	1
1.1 Creating a Topic	1
1.2 Modifying the Display Name of a Topic	3
1.3 Adding, Modifying, or Deleting a Topic Tag	4
1.4 Configuring Topic Policies	7
1.4.1 Basic Mode	7
1.5 Adding a Subscription to a Topic	9
1.6 Publishing a Message	13
1.6.1 Introduction	13
1.6.2 Publishing a Text Message	13
1.6.3 Publishing a JSON Message	16
1.6.4 Publishing a Template Message	19
1.7 Deleting a Topic	21
2 Subscription Management	23
2.1 Adding a Subscription	23
2.2 Requesting Subscription Confirmation	27
2.3 Canceling a Subscription	29
2.4 Deleting a Subscription	29
3 Message Template Management	31
4 SMN Operation Recording	34
4.1 Introduction	34
4.2 Key SMN Operations Recorded by CTS	34
4.3 CTS Traces	35
5 Monitoring	38
5.1 Metrics	38
5.2 Viewing Metrics	41
5.3 Configuring Alarm Rules	42
6 Logs	43
7 Permissions Management	47
7.1 Creating a User and Granting SMN Permissions	47
7.2 Creating SMN Custom Policies	49

8 Quotas	51
A Appendix	53
A.1 JSON Message Format.....	53
A.2 Template Message Format.....	55
A.3 Messages Using Different Protocols.....	55
A.4 Traffic Control over Subscription Confirmation.....	55
A.5 Country or Region Codes.....	56
A.6 Mappings Between SMN Actions and APIs.....	63
A.7 Restrictions on SMS Messaging.....	64
A.8 HTTP/HTTPS Messages.....	64
A.8.1 Introduction.....	65
A.8.2 HTTP or HTTPS Message Format.....	65
A.8.3 Message Signature Verification.....	69
A.8.4 Sample Code.....	70
B Change History	75

1 Topic Management

1.1 Creating a Topic

Scenarios

A topic is a specified event to publish messages and subscribe to notifications. It serves as a message sending channel, where publishers and subscribers can interact with each other.

Creating a Topic



1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. In the upper right corner, click **Create Topic**.

Figure 1-1 Create Topic

Create Topic

✕

* Topic Name
The name cannot be changed after the topic is created.

Display Name
If a display name is specified, the email sender is presented in the format "Display name<username@example.com>" when email messages are delivered.

* Enterprise Project ↕ [Create Enterprise Project](#)
Enterprise Project Management Service (EPS) allows you to manage cloud resources and user groups by enterprise project.

CTS Log

Tag It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. [View predefined tags](#) ↻
To add a tag, enter a tag key and a tag value below.

Tags you can still add: 20

6. Enter a topic name and display name.

Table 1-1 Parameter descriptions

Parameter	Description
Topic Name	Topic name, which: <ul style="list-style-type: none"> Contains only letters, digits, hyphens (-), and underscores (_), and must start with a letter or digit. Contains 1 to 255 characters. Must be unique and cannot be modified once the topic is created.
Display Name	Message sender name which can contain up to 192 characters NOTE After you specify a display name, the sender in email messages will be presented as <i>Display name<username@example.com></i> . Otherwise, the sender will be username@example.com .
Enterprise Project	Centrally manages cloud resources and members by project.
CTS Log	Whether to enable CTS log.

Parameter	Description
Log Group	Select a log group. A log group is a group of log streams which share the same log retention settings. For details, see Log Groups .
Log Stream	Select a log stream in the specified log group. A log stream is the basic unit for log reads and writes. You can sort logs of different types, such as operation logs and access logs, into different log streams. For details, see Log Streams .
Tag	A tag is a key-value pair. Tags identify cloud resources so that you can easily categorize and search for your resources. <ul style="list-style-type: none">• A key can contain up to 128 characters, and a value can contain up to 256 characters. The value can contain letters, digits, underscores (_), periods (.), colons (:), slashes (/), equal signs (=), plus signs (+), hyphens (-), and at signs (@).• You can add up to 20 tags for each topic.

7. Click **OK**.

The topic you created is displayed in the topic list. The system generates a topic URN, which is the unique resource identifier of the topic and cannot be changed.



8. Click the name of the topic to view its details, including the topic URN, display name tags, and subscriptions.

1.2 Modifying the Display Name of a Topic


Scenarios

You have created a topic and want to modify its display name.

Modifying the Display Name of a Topic

1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.

5. Locate the topic, choose **More > Modify Display Name** in the **Operation** column. In the displayed **Modify Displayed Name** dialog box, enter a new display name.

Alternatively, locate the topic and click its name. In the **Topic Details** area, click  and enter a new display name.

 **NOTE**

After you specify a display name, the sender in email messages will be presented as *Display name<username@example.com>*. Otherwise, the sender will be *username@example.com*.

6. Click **OK**.

1.3 Adding, Modifying, or Deleting a Topic Tag

Scenarios

Tags consist of keys and values. They identify cloud resources so that you can easily categorize and search for your resources.

- A tag key can have multiple values.
- Tag keys for the same resource must be unique.

Adding Tags to a Topic



1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. Click the topic name.
The **Topic Details** page is displayed.
6. Click the **Tags** tab.
7. Click **Add Tag** and specify tag keys and values.

Figure 1-2 Add Tag

Add Tag ×

It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources. [View predefined tags](#)

To add a tag, enter a tag key and a tag value below.

Enter a tag key Enter a tag value

Tags you can still add: 20

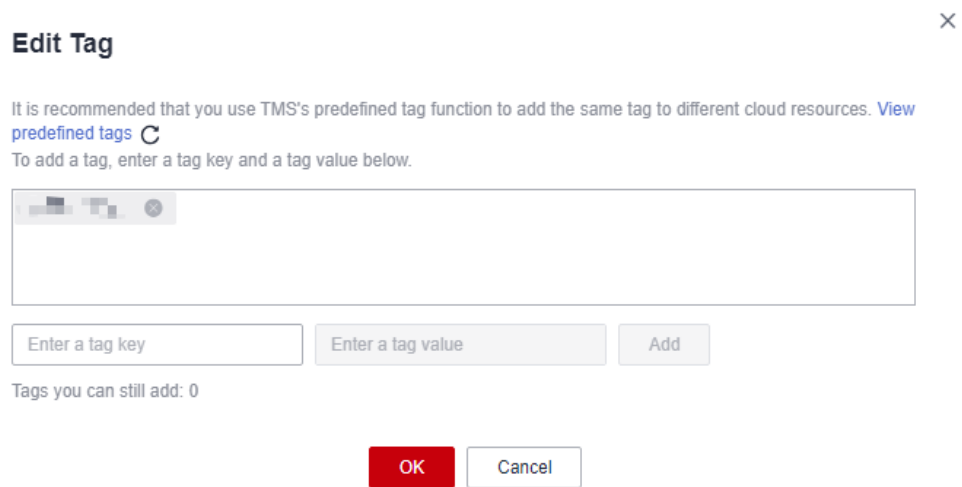
NOTE

- You can add up to 20 tags for each topic.
 - If you have configured tag policies for message notifications, add tags to your topics based on the tag policies. If you add a tag that does not comply with the tag policies, topics may fail to be created. Contact the administrator to learn more about tag policies.
8. Click **OK**.
- The tags you added are displayed in the list.

Modifying a Topic Tag

1. Log in to the management console.
2. In the upper left corner of the page, click and select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. Click the topic name.
The **Topic Details** page is displayed.
6. Click the **Tags** tab.
7. Click **Edit** under **Operation** to modify the tag value.

Figure 1-3 Edit Tag



8. Click **OK**.

Deleting a Topic Tag



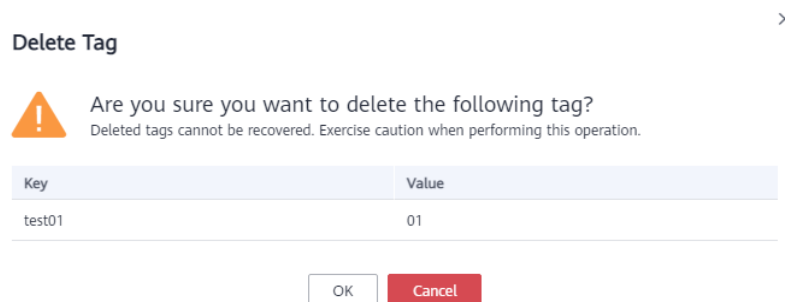
1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. Click the topic name.
The **Topic Details** page is displayed.
6. Click the **Tags** tab.
7. Click **Delete** in the **Operation** column, and click **OK** in the confirmation dialog box.

Figure 1-4 Delete Tag



1.4 Configuring Topic Policies

1.4.1 Basic Mode

Only users under the same account as the topic creator have the permissions to publish messages through the topic. Using topic policies, you can specify which users and cloud services can perform which topic operations, for example, querying topic details and publishing messages. Topic creators always have permissions over a topic even if they grant topic permissions to other users.

Configuring Topic Policies in Basic Mode



1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. Locate a topic, click **More** under **Operation**, and select **Configure Topic Policy**.
Alternatively, click a topic name. In the upper right corner of the displayed page, click **Configure Topic Policy**.
6. In the **Configure Topic Policy** dialog box, configure the topic policy in basic mode.
The basic mode simply specifies which users or cloud services have permissions to publish messages to the topic. For details, see [Table 1-2](#).

Figure 1-5 Basic mode

Configure Topic Policy ×

Topic Name ██████████

Policy **Basic**

The policy allows selected users and cloud services to publish messages to this topic.

Users who can publish messages to this topic

Topic creator

All users

Specified user accounts

Enter one or more account IDs or URNs, each on a separate line.

[Learn how](#) to obtain an account ID.

Services that can publish messages to this topic

OBS DWS APM AAD EFS VOD

MPC LTS CTS

Table 1-2 Description for configuring topic policies in basic mode

Item	Parameter	Description
Users who can publish messages to this topic	Topic creator	All IAM users under the same account as the topic creator have the permissions to publish messages through the topic.
	All users	All users have the permission to publish messages to the topic.



Item	Parameter	Description
	Specified user accounts	<p>Only specified users have the permission to publish messages to the topic. Users are specified in the following format: <code>urn:csp:iam::domainId:root.domainId</code> indicates the account IDs of the users.</p> <p>You only need to enter the account ID and click OK. The system completes all other required information for you. SMN does not limit the number of IDs you enter, but the total size of a topic policy cannot exceed 30 KB.</p> <p>To obtain your account ID, log in to the SMN console. In the upper right corner, hover the mouse over your login account, and select My Credentials from the drop-down list.</p> <p>Enter one account ID or URN for each one line.</p>
Services that can publish messages to this topic	Example: OBS The services that can publish messages to a topic vary in different regions.	<p>The selected cloud services have operation permissions of the topic.</p> <p>NOTE By default, Cloud Eye and Anti-DDoS have the permissions to publish messages to topics created by all users. For details about how to use SMN in other cloud services, see user guides of the related services.</p>

1.5 Adding a Subscription to a Topic

Scenarios

To deliver messages published to a topic to endpoints, you must add the subscription endpoints to the topic.

To Add a Subscription

1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. Locate the topic that you want to add a subscription to. In the **Operation** column, click **Add Subscription**.

Alternatively, click a topic name. In the upper right corner of the displayed page, click **Add Subscription**.

The **Add Subscription** dialog box is displayed.

Figure 1-6 Add Subscription

Add Subscription ×

Basic Information

Topic Name

* Protocol

⚠ If you add SMS, email, or HTTP/HTTPS subscriptions to a topic, you will be billed as described in [pricing details](#).

* Endpoint ?

Endpoints	Description
<input type="text"/>	<input type="text"/>

+ Add Endpoint
[Batch Add Endpoints](#)

i After your subscription is added, you must confirm it. ?

Filter Policy ☐

Filter policies allow you to limit which subscription endpoints the message will be sent to.

OK Cancel

- Specify the subscription protocol and endpoints.

Table 1-3 Parameters for adding a subscription

Parameter	Description
Topic Name	Specifies the name of the topic to which messages are published.

Parameter	Description
Protocol	<p>Specifies the protocol over which messages are sent. Possible values include SMS, HTTP, HTTPS, FunctionGraph (function), Voice notification, DingTalk chatbot, WeCom chatbot, Lark chatbot, and Email.</p> <p>NOTE</p> <ul style="list-style-type: none"> • Voice notification, DingTalk chatbot, WeCom chatbot, and Lark chatbot are in the open beta test (OBT). To use these protocols, submit a service ticket to apply for the OBT. • After the OBT is enabled, the current IAM user token will be invalid. Log in to the console again to use the token. When calling an API, obtain a new token. • The number of WeCom, DingTalk, and Lark messages that can be sent is limited, and the recipient system may be faulty. As a result, messages may fail to be sent. In this case, SMN does not ensure successful message delivery.
Endpoint	<p>Specifies the subscription endpoint. You can add up to 10 SMS, email, HTTP, or HTTPS endpoints, one in each line.</p> <ul style="list-style-type: none"> • SMS: Enter one or more valid phone numbers. Examples: 18512345678 +8618512345678 • Email: Enter one or more valid email addresses. Examples: username@example.com username2@example.com • HTTP: Enter one or more public network URLs. Example: http://example.com/notification/action • HTTPS: Enter one or more public network URLs. Example: https://example.com/notification/action <div data-bbox="1118 1480 1406 1552" style="border: 1px solid #ccc; border-radius: 5px; padding: 5px; display: inline-block; margin: 10px 0;">Add Endpoint</div> <ul style="list-style-type: none"> • FunctionGraph (function): Click to select a function and specify its version. • Voice notification: Enter one or more valid phone numbers. The phone number must be preceded by a plus sign (+) and country code. If you enter a phone number used in the Chinese mainland, the plus sign (+) and country code are not required. <p>Voice subscriptions are available only in some regions of the Chinese mainland. Voice notification is not supported in Hong Kong, Macao, Taiwan, Xinjiang, and Tibet.</p>

Parameter	Description
Request Header	This parameter is available when you set Protocol to HTTP or HTTPS . It indicates whether to configure the request header immediately. If you select Configure now , you need to specify Key and Value . You can add up to 10 request headers. The value of Key must: <ul style="list-style-type: none">• be case insensitive and unique.• start with x- but cannot start with x-smn.• contain only digits, letters, and hyphens (-), but not end with a hyphen nor contain consecutive hyphens.
Availability Test	This parameter is available when you set Protocol to HTTP or HTTPS . Click Start to check the real-time connectivity between SMN and HTTP/HTTPS endpoints.
Version	This parameter is available when you set Protocol to FunctionGraph (function) . Select the version of a created FunctionGraph (function). For details about how to create a version, see Version Management .
Description	Specifies the remarks of the subscription.

7. (Optional) Configure a subscription filtering policy to limit the scope of message publishing. The default filtering policy applies to message attributes. If you have set a filtering policy and message attributes when publishing a message, the system determines whether to push the message to the subscriber based on the filtering policy.

Subscription filtering policies are configured in JSON. The following is an example.

```
{
  "filter_policies": [
    {
      "name": "policy_name",
      "string_equals": [
        "policy_value"
      ]
    }
  ]
}
```

8. Click **OK**.

The subscription you added is displayed in the subscription list.

 NOTE

- To prevent malicious users from attacking subscription endpoints, SMN limits the number of confirmation messages that can be sent to an endpoint within a specified period. For details, see section "Traffic Control on Subscription Confirmation" in *Simple Message Notification User Guide*.
- SMN does not check whether subscription endpoints exist when you add subscriptions. However, subscribers will not receive notification messages until they confirm their subscriptions.
- After you add a subscription, SMN sends a confirmation message to your subscription endpoint. You can confirm the subscription within 48 hours through the confirmation link via your mobile phone, mailbox, or other endpoints.
- Subscription confirmation messages will be counted as messages sent and will be billed.

1.6 Publishing a Message

1.6.1 Introduction

SMN enables you to publish messages in the following formats:

- Text
- JSON
- Template

After you publish a message to a topic, SMN will deliver the message to all confirmed subscription endpoints in the topic.

For SMS endpoints, if an SMS message is oversized, the system divides it into multiple parts when sending it to subscribers. However, you must note that SMN only sends the first two parts of the SMS message and does not send any additional parts. You are charged based on the actual number of messages sent to the subscribers.

You must ensure that firewall policies of the HTTP or HTTPS endpoints allow SMN to send messages over the Internet. An SMN HTTP or HTTPS message consists of a message header and body. For details, see [HTTP or HTTPS Message Format](#).

1.6.2 Publishing a Text Message

Scenarios

After you publish a text message to a topic, SMN will deliver the message to all confirmed subscription endpoints in the topic.

Prerequisites

Subscribers in the topic must have confirmed the subscription, or they will not be able to receive any messages.

Procedure



1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. In the topic list, locate the topic that you need to publish a message to and click **Publish Message** in the **Operation** column.
Alternatively, locate the topic and click its name. In the upper right corner of the displayed topic details page, click **Publish Message**.
6. Configure the required parameters based on [Table 1-4](#).
 - a. Configure basic information about message publishing.

Table 1-4 Parameter descriptions

Parameter	Description
Subject	(Optional) The message subject must be fewer than 512 bytes.
Message Format	The message format can be Text , JSON , or Template . In this context, select Text . <ul style="list-style-type: none">• Text: common text message• JSON: JSON message• Template: template message. For details, see Message Template Management.
Message	This is the message content, which cannot be left blank nor exceed 256 KB.

- b. (Optional) Configure message attribute parameters. Message attributes specify the scope of message publishing.

Table 1-5 Message attribute parameters

Parameter	Description
Type	Select the type of the message to be published. <ul style="list-style-type: none">• Protocol• String array• String

Parameter	Description
Name	<p>Enter up to 32 characters, including only digits, lowercase letters, and underscores (_). Start with a number or lowercase letter. Do not end with an underscore (_) or enter consecutive underscores (_).</p> <ul style="list-style-type: none">• When you set Type to Protocol, Name is smn_protocol by default.• If you set Type to String array, enter the name of the array that restricts the message to be published.• When you set Type to String, enter the name of the character string that restricts the message to be published.
Value	<ul style="list-style-type: none">• When you set Type to Protocol, select a protocol from the drop-down list. The available options are SMS, Email, HTTP, HTTPS, FunctionGraph (function), and functionGraph (workflow).• When you set Type to String array, enter a string array. Length: 1 to 10• When you set Type to String, you cannot leave Value blank. Enter up to 32 characters, including only digits, letters, and underscores (_).

Figure 1-7 shows an example text message.

Figure 1-7 Text message example

7. Click **OK**.

SMN delivers your message to all subscription endpoints. For details about the messages received by each endpoint, see [Messages Using Different Protocols](#).

1.6.3 Publishing a JSON Message

Scenarios

In a JSON message, you can specify different message content for different protocols, including SMS, email, FunctionGraph (function), HTTP, and HTTPS.

Prerequisites

Subscribers in the topic must have confirmed the subscription, or they will not be able to receive any messages.

Procedure



1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. In the topic list, locate the topic that you need to publish a message to and click **Publish Message** in the **Operation** column.
Alternatively, locate the topic and click its name. In the upper right corner of the displayed topic details page, click **Publish Message**.
6. Configure parameters by referring to [Table 1-6](#).

Table 1-6 Parameters required for publishing a message

Parameter	Description
Subject	(Optional) Specifies the message subject, which must be fewer than 512 bytes.
Message Format	Specifies in which format a message is published. You can select only one message format each time you publish a message. <ul style="list-style-type: none">• Text: common text message• JSON: JSON message• Template: template message. For details, see Message Template Management.
Message	Specifies the message content. The message content cannot be left blank nor exceed 256 KB.

Select **JSON** for **Message Format**. Then, manually type the JSON message in the **Message** box or click **Generate JSON Message** to generate it automatically.

- If you choose to manually type the JSON message, see [JSON Message Format](#).
 - If you choose to automatically generate the JSON message, proceed with steps [7](#) through [10](#).
7. Click **Generate JSON Message**.
 8. Enter your message content, for example, **This is a default message.**, in the **Message** box and select the desired message protocols.

The size of a JSON message varies depending on the protocol combinations. As you type in the message content, the system will calculate the number of

bytes you have entered, the size of the JSON message, and how many bytes are left. The total size of a JSON message includes braces, quotation marks, spaces, line breaks, and message content. For details about how to calculate the size of a JSON message, see [Calculation on the Size of a JSON Message](#) in [JSON Message Format](#).

9. Click **OK**.
10. Modify the message content for each protocol so that different messages are sent to endpoints of different protocols. The system generates JSON-formatted content that includes a default message and content for each protocol. When SMN fails to match any specific message protocol, it sends the default message. For detailed, see [JSON Message Format](#).
11. (Optional) Configure message attribute parameters. Message attributes specify the scope of message publishing.

Table 1-7 Message attribute parameters

Parameter	Description
Type	Select the type of the message to be published. <ul style="list-style-type: none">• Protocol• String array• String
Name	Enter up to 32 characters, including only digits, lowercase letters, and underscores (_). Start with a number or lowercase letter. Do not end with an underscore (_) or enter consecutive underscores (_). <ul style="list-style-type: none">• When you set Type to Protocol, Name is smn_protocol by default.• If you set Type to String array, enter the name of the array that restricts the message to be published.• When you set Type to String, enter the name of the character string that restricts the message to be published.
Value	<ul style="list-style-type: none">• When you set Type to Protocol, select a protocol from the drop-down list. The available options are SMS, Email, HTTP, HTTPS, FunctionGraph (function), and functionGraph (workflow).• When you set Type to String array, enter a string array. Length: 1 to 10• When you set Type to String, you cannot leave Value blank. Enter up to 32 characters, including only digits, letters, and underscores (_).

12. Click **OK**.
SMN delivers your message to all subscription endpoints. For details about the messages received by each endpoint, see [Messages Using Different Protocols](#).

1.6.4 Publishing a Template Message

Scenarios

Message templates contain fixed message content. If you need to send the same or similar messages multiple times, you can create a message template for quick message sending.

You can create different templates for different protocols using the same template name so that each type of subscribers can receive customized messages. Templates contain variables as the placeholders to represent changeable content that you can replace with your own message content. Note that you must create a template whose **Protocol** is **Default**, or the system will prevent you from publishing messages using this template name.

When you are creating messages using a template, select a template name. The system will list all variables in the following protocol sequence: **Default**, **SMS**, **Email**, **FunctionGraph (function)**, **HTTP**, and **HTTPS**. The same variables are listed only once even if they are used in multiple protocols, and the protocols they support are listed after each variable. Specify content for each variable in the message template, and SMN replaces them with the content you entered. If you do not enter any content for a variable, the system will treat it as empty when sending messages.



SMN tries to match different types of subscribers to the template protocols. If there is no template for a specified protocol, SMN will use the default template to send messages to subscribers of that protocol.

This section describes how to publish messages using a template. For more details about message templates, see [Message Template Management](#).

Prerequisites

Subscribers in the topic must have confirmed the subscription, or they will not be able to receive any messages.

Creating a Message Template

1. Log in to the management console.
2. Click  on the upper left to select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Message Templates**.
5. In the upper right corner, click **Create Message Template**. For details, see [Creating a Message Template](#) in [Message Template Management](#).

For example, the template information is as follows:

- **Template Name:** tem_001
- **Protocol:** Default

- **Content: The Arts and Crafts Exposition will be held from {startdate} through {enddate}. We sincerely invite you to join us.**

Publishing a Template Message


1. Log in to the management console.
2. Click  on the upper left to select the desired region and project.
3. Under **Management & Governance**, select **Simple Message Notification**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. In the topic list, locate the topic that you need to publish a message to and click **Publish Message** in the **Operation** column.
Alternatively, locate the topic and click its name. In the upper right corner of the displayed topic details page, click **Publish Message**.
6. Configure the required parameters. (The topic name is provided by default and cannot be changed.)
Select **Template** for **Message Format**. Then, manually type the template content in the **Message** box or click **Generate Template Message** to generate it automatically. The message content cannot be left blank nor exceed 256 KB.
 - If you choose to manually type the template message, see "Template Message Format" in *Simple Message Notification User Guide* for detailed requirements.
 - If you choose to automatically generate the template message, proceed with **7** through **10**.
7. Click **Generate Template Message**.
8. Select a template name, for example, **tem_001**, and enter values for the variables.
The system replaces the variables with the message content you specified. The protocols configured in the template are displayed after each variable. Only the **Default** protocol is specified in **tem_001**. Therefore, all confirmed subscribers in the topic will receive the message content in the default template.
9. Click the **Preview** tab, and click **Message Preview** to preview the message.
In this example, the message generated is **The Arts and Crafts Exposition will be held from February 10 through February 21. We sincerely invite you to join us..**
10. Click **OK**.
The message that is generated contains the template name and variables.
11. (Optional) Configure message attribute parameters. Message attributes specify the scope of message publishing.



Table 1-8 Message attribute parameters

Parameter	Description
Type	Select the type of the message to be published. <ul style="list-style-type: none">• Protocol• String array• String
Name	Enter up to 32 characters, including only digits, lowercase letters, and underscores (_). Start with a number or lowercase letter. Do not end with an underscore (_) or enter consecutive underscores (_). <ul style="list-style-type: none">• When you set Type to Protocol, Name is smn_protocol by default.• If you set Type to String array, enter the name of the array that restricts the message to be published.• When you set Type to String, enter the name of the character string that restricts the message to be published.
Value	<ul style="list-style-type: none">• When you set Type to Protocol, select a protocol from the drop-down list. The available options are SMS, Email, HTTP, HTTPS, FunctionGraph (function), and functionGraph (workflow).• When you set Type to String array, enter a string array. Length: 1 to 10• When you set Type to String, you cannot leave Value blank. Enter up to 32 characters, including only digits, letters, and underscores (_).

12. Click **OK**.

SMN delivers your message to all subscription endpoints. For details about messages for different protocols, see section "Messages of Different Protocols" in *Simple Message Notification User Guide*.

1.7 Deleting a Topic

1. Log in to the management console.
2. Click  on the upper left to select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. Locate a topic, click **More** in the **Operation** column, and select **Delete**.

6. Confirm the information about the topic to be deleted and enter **DELETE**.
7. Click **OK**.

 **NOTE**

Deleting a topic deletes all its subscriptions.

2 Subscription Management

2.1 Adding a Subscription

Scenarios

To deliver messages published to a topic to endpoints, you must add the subscription endpoints to the topic. Endpoints can be email addresses, phone numbers, and HTTP/HTTPS URLs. After you add endpoints to the topic and the subscribers confirm the subscription, they are able to receive messages published to the topic.

You can add multiple subscriptions to each topic. This section describes how you can add a subscription to a topic you created or a topic that you have permissions for.

Adding a Subscription



1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Subscriptions**.
5. In the upper right corner, click **Add Subscription**.
The **Add Subscription** dialog box is displayed.

Figure 2-1 Add Subscription

Add Subscription
✕

Basic Information

Topic Name

* Protocol SMS

⚠ If you add SMS, email, or HTTP/HTTPS subscriptions to a topic, you will be billed as described in [pricing details](#).

* Endpoint ?

Endpoints	Description
<input style="width: 95%;" type="text"/>	<input style="width: 95%;" type="text"/>

+ Add Endpoint

[Batch Add Endpoints](#)

i After your subscription is added, you must confirm it. ?

Filter Policy

Filter policies allow you to limit which subscription endpoints the message will be sent to.


OK
Cancel

6. Specify the required subscription information.
 - a. Beside the **Topic Name** box, click **Select Topic**.
 - b. Specify the subscription protocol and endpoints.

Table 2-1 Parameters for adding a subscription

Parameter	Description
Topic Name	Specifies the name of the topic to which messages are published.

Parameter	Description
Protocol	<p>Specifies the protocol over which messages are sent. Possible values include SMS, HTTP, HTTPS, FunctionGraph (function), Voice notification, DingTalk chatbot, WeCom chatbot, Lark chatbot, and Email.</p> <p>NOTE</p> <ul style="list-style-type: none">• Voice notification, DingTalk chatbot, WeCom chatbot, and Lark chatbot are in the open beta test (OBT). To use these protocols, submit a service ticket to apply for the OBT.• After the OBT is enabled, the current IAM user token will be invalid. Log in to the console again to use the token. When calling an API, obtain a new token.• The number of WeCom, DingTalk, and Lark messages that can be sent is limited, and the recipient system may be faulty. As a result, messages may fail to be sent. In this case, SMN does not ensure successful message delivery.

Parameter	Description
Endpoint	<p>Specifies the subscription endpoint. You can add up to 10 SMS, email, HTTP, or HTTPS endpoints, one in each line.</p> <ul style="list-style-type: none">• SMS: Enter one or more valid phone numbers. Examples: 18512345678 +8618512345678• Email: Enter one or more valid email addresses. Examples: username@example.com username2@example.com• HTTP: Enter one or more public network URLs. Example: http://example.com/notification/action• HTTPS: Enter one or more public network URLs. Example: https://example.com/notification/action• FunctionGraph (function): Click  to select a function and specify its version.• Voice notification: Enter one or more valid phone numbers. The phone number must be preceded by a plus sign (+) and country code. If you enter a phone number used in the Chinese mainland, the plus sign (+) and country code are not required. Voice subscriptions are available only in some regions of the Chinese mainland. Voice notification is not supported in Hong Kong, Macao, Taiwan, Xinjiang, and Tibet.
Request Header	<p>This parameter is available when you set Protocol to HTTP or HTTPS. It indicates whether to configure the request header immediately. If you select Configure now, you need to specify Key and Value. You can add up to 10 request headers.</p> <p>The value of Key must:</p> <ul style="list-style-type: none">• be case insensitive and unique.• start with x- but cannot start with x-smn.• contain only digits, letters, and hyphens (-), but not end with a hyphen nor contain consecutive hyphens.
Availability Test	<p>This parameter is available when you set Protocol to HTTP or HTTPS. Click Start to check the real-time connectivity between SMN and HTTP/HTTPS endpoints.</p>

Parameter	Description
Version	This parameter is available when you set Protocol to FunctionGraph (function) . Select the version of a created FunctionGraph (function). For details about how to create a version, see Version Management .
Description	Specifies the remarks of the subscription.

- c. (Optional) Configure a subscription filtering policy to limit the scope of message publishing. The default filtering policy applies to message attributes. If you have set a filtering policy and message attributes when publishing a message, the system determines whether to push the message to the subscriber based on the filtering policy.

Subscription filtering policies are configured in JSON. The following is an example.

```
{
  "filter_policies": [
    {
      "name": "policy_name",
      "string_equals": [
        "policy_value"
      ]
    }
  ]
}
```

7. Click **OK**.

The subscription you added is displayed in the subscription list.

NOTE



- To prevent malicious users from attacking subscription endpoints, SMN limits the number of confirmation messages that can be sent to an endpoint within a specified period. For details, see section "Traffic Control on Subscription Confirmation" in *Simple Message Notification User Guide*.
- SMN does not check whether subscription endpoints exist when you add subscriptions. However, subscribers will not receive notification messages until they confirm their subscriptions.
- After you add a subscription, SMN sends a confirmation message to your subscription endpoint. You can confirm the subscription within 48 hours through the confirmation link via your mobile phone, mailbox, or other endpoints.
- Subscription confirmation messages will be counted as messages sent and will be billed.

2.2 Requesting Subscription Confirmation

Scenarios

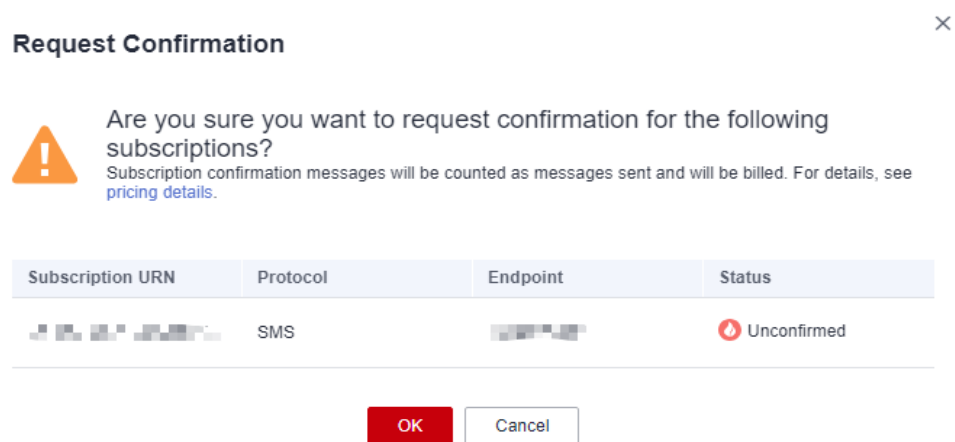
If a subscriber does not receive the confirmation message, request confirmation again. You can send a subscription confirmation message to one or more subscription endpoints at a time. For details, see [Traffic Control over Subscription Confirmation](#).

Requesting Subscription Confirmation

1. Log in to the management console.
2. Click  on the upper left to select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Subscriptions**.
5. In the subscription list, select one or more subscriptions whose status is **Unconfirmed**.
6. Click **Request Confirmation** above the subscription list and SMN will send confirmation requests.

Alternatively, click **Request Confirmation** in the **Operation** column, and SMN will send the confirmation request.

Figure 2-2 Requesting subscription confirmation



7. The subscribers confirm their subscriptions.

NOTE

- To prevent malicious users from attacking subscription endpoints, SMN limits the number of confirmation messages that can be sent to an endpoint within a specified period. For details, see [Traffic Control over Subscription Confirmation](#).
- SMN does not check whether subscription endpoints exist when you add subscriptions. However, subscribers will not receive notification messages until they confirm their subscriptions.
- After you add a subscription, SMN sends a message that contains a link for confirming the subscription to the subscription endpoint. The subscription confirmation link is valid within 48 hours. Confirm the subscription on your mobile phone, mailbox, or other endpoints in time.
- Subscription confirmation messages will be counted as messages sent and will be billed.

2.3 Canceling a Subscription

Scenarios

After you add subscriptions to a topic, the subscribers receive a confirmation message and need to confirm their subscriptions to receive notification messages published to the topic. If the subscribers no longer want to receive notifications from a topic, they can choose to cancel subscriptions.

 **CAUTION**

The subscription management capability of SMN is open to subscribers. You must keep your subscription links secure to avoid being unable to receiving notifications or receiving unexpected notifications.

Canceling a Subscription

A subscriber can choose to cancel the subscription based on the protocol of the subscription endpoint:

- **SMS:** SMN does not provide a link to unsubscribe in SMS notification messages because of the message length limit. To cancel an SMS subscription, the subscriber needs to access the link provided in the subscription confirmation message and cancels the subscription on the web page.
- **Email:** SMN encloses a link to unsubscribe in email notifications. The subscriber can cancel the subscription by clicking the link. After the subscriber has canceled the subscription, SMN re-sends a subscription confirmation email which is valid for 48 hours, so that the subscriber can re-subscribe to the topic if they clicked the link by mistake.
- **HTTP/HTTPS:** SMN provides a link to unsubscribe in the HTTP/HTTPS message body. The subscriber can cancel the subscription by clicking the link. After the subscriber has canceled the subscription, the system returns **200** over HTTP and re-sends a subscription confirmation message which is valid for 48 hours, in case the subscriber has clicked the link by mistake. For details about the HTTP/HTTPS message header and body, see [Introduction](#).



2.4 Deleting a Subscription

Scenarios



If one or multiple subscription endpoints do not need to receive messages published to a topic, you can delete them.

Deleting a Subscription on the Topic Details Page

1. Log in to the management console.

2. Click  on the upper left to select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Topics**.
The **Topics** page is displayed.
5. Click the topic name.
The **Topic Details** page is displayed.
6. In the **Subscriptions** area, view the subscriptions to the topic.
7. Select one or more subscription endpoints and click **Delete**.
8. In the displayed **Delete Subscription** dialog box, click **OK**.

Deleting a Subscription on the Subscription Page

1. Log in to the management console.
2. Click  on the upper left to select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane on the left, choose **Topic Management > Subscriptions**.
The **Subscription** page is displayed.
5. In the subscription list, select one or more subscriptions and click **Delete** above the subscription list.
6. In the displayed **Delete Subscription** dialog box, click **OK**.

3 Message Template Management

Scenarios

Message templates contain fixed and changeable content and can be used to create and send messages more quickly. When you use a template to publish a message, you need to specify values for different variables in the template.

Message templates are identified by name, but you can create different templates with the same name as long as they are configured for different protocols. All template messages must include a **Default** template or they cannot be sent out. The **Default** template is used anytime a template has not been configured for a given protocol, but as long as there is a template for the protocol, then any subscriber who selected that protocol when they subscribed will receive a message using the corresponding template.

This section describes how to publish messages using a template.

Creating a Message Template



1. Log in to the management console.
2. Click  on the upper left to select the desired region and project.
3. Click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
4. In the navigation pane, choose **Topic Management > Message Templates**.
5. In the upper right corner, click **Create Message Template**.
The **Create Message Template** dialog box is displayed.

Figure 3-1 Create Message Template

Create Message Template ×

Template Name

If a template ABC will be used to publish messages, you must create a template named ABC by using the default protocol, and then create other templates named ABC by using other protocols, such as SMS and email. Otherwise, template ABC cannot be used to publish messages.

Protocol

Content

Use variables quoted by brackets, for example, {xxx}, as placeholders to represent changeable content in the template. A variable contains a maximum of 21 characters consisting of letters, digits, hyphens (-), underscores (_), or periods (.) and must start with a letter or digit. Before publishing messages using a template, you can replace variables with specific content.

OK Cancel

6. Specify the template name, protocol, and content.

Table 3-1 Parameters required for creating a message template

Parameter	Description
Template Name	<p>Template name, which:</p> <ul style="list-style-type: none"> • Contains only letters, digits, hyphens (-), and underscores (_), and must start with a letter or digit. • Can contain 1 to 64 bytes. • Cannot be modified once the template is created.
Protocol	<p>Endpoint protocol of the template, which cannot be changed once the template is created</p> <p>The protocol can be Default, SMS, Email, HTTP, HTTPS, or FunctionGraph (function).</p> <p>If you do not specify a protocol, Default is used.</p>

Parameter	Description
Content	<p>Template content</p> <p>Use <code>{xxx}</code> as the placeholder to create a template. When you use this template to send messages, replace <code>{xxx}</code> with specific content. <code>xxx</code> must start with a letter or digit and can contain up to 21 characters, including only letters, digits, hyphens (-), periods (.), and underscores (_).</p> <p>The message template must meet the following requirements:</p> <ul style="list-style-type: none">• The template supports plain text only.• The template content cannot be left blank and cannot exceed 256 KB.• The template can contain up to 256 variables in total, but that includes redundant variables. For unique variables, there can be no more than 90.• When you send messages using a template, the message content you specify for each variable cannot exceed 1 KB.

For example, the template information is as follows:

- **Template Name:** tem_001
- **Protocol:** Default
- **Content:** The Arts and Crafts Exposition will be held from {startdate} through {enddate}. We sincerely invite you to join us.

7. Click **OK**.

The template you created is displayed in the template list.

Modifying a Template

1. On the **Message Templates** page, locate the template to be modified in the template list.
2. Click **Modify** in the **Operation** column to change its content.

Deleting a Template

1. On the **Message Templates** page, locate the template to be deleted in the template list.
2. Click **Delete** in the **Operation** column.

4 SMN Operation Recording

4.1 Introduction

You can use Cloud Trace Service (CTS) to record information about SMN-related operations, including request content, source IP addresses, request senders, and when a request was sent, for future query, audit, and backtracking.

CTS can record operations performed on the management console, performed by calling APIs, and triggered within the CTS system.

4.2 Key SMN Operations Recorded by CTS

After you enable CTS, whenever an SMN API is called, the operation is recorded in a log file, which is then dumped to a specified OBS bucket for storage based on time and data changes.

However, if someone makes an API call to cancel a subscription without login, CTS will not record the operation. For example, if a subscriber clicks the link in an email notification to cancel the subscription, the unsubscribe API is called, but CTS does not record the operation.

Table 4-1 lists the SMN operations that will be recorded by CTS.

Table 4-1 SMN operations recorded by CTS

Operation	Resource	Trace Name
Creating a topic	Topic	createTopic
Deleting a topic	Topic	deleteTopic
Updating a topic	Topic	updateTopic
Updating a topic policy	Topic	updateTopicAttribute
Deleting all topic policies	Topic	deleteTopicAttributes

Operation	Resource	Trace Name
Deleting a specified topic policy	Topic	deleteTopicAttributeByName
Adding a subscription	Subscription	subscribe
Deleting a subscription	Subscription	delsubscribe
Creating a message template	Message template	createMessageTemplate
Creating message templates in batches	Message template	batchCreateMessageTemplate
Modifying a message template	Message template	updateMessageTemplate
Deleting a message template	Message template	deleteMessageTemplate


4.3 CTS Traces

Scenarios

After CTS is enabled, it starts recording operations on cloud resources. You can view the operation records of the last seven days on the management console.

This topic describes how to query or export the last seven days of operation records on the CTS console.

Procedure

1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click **Service List** in the upper left corner. Under **Management & Governance**, select **Cloud Trace Service**.
4. In the left navigation pane, choose **Trace List**.
5. Specify filters to search for your desired traces.
 - **Trace Type, Trace Source, Resource Type, and Search By**
Select the filter from the drop-down list.
If you select **Resource ID** for **Search By**, specify a resource ID.
If you select **Data** for **Trace Type**, you can only filter traces by tracker.
 - **Operator**: Select one or more operators from the drop-down list.
 - **Trace Status**: Available options include **All trace statuses, normal, warning, and incident**. You can select only one of them.
 - **Time range**: You can query traces generated at any time range of the last seven days.


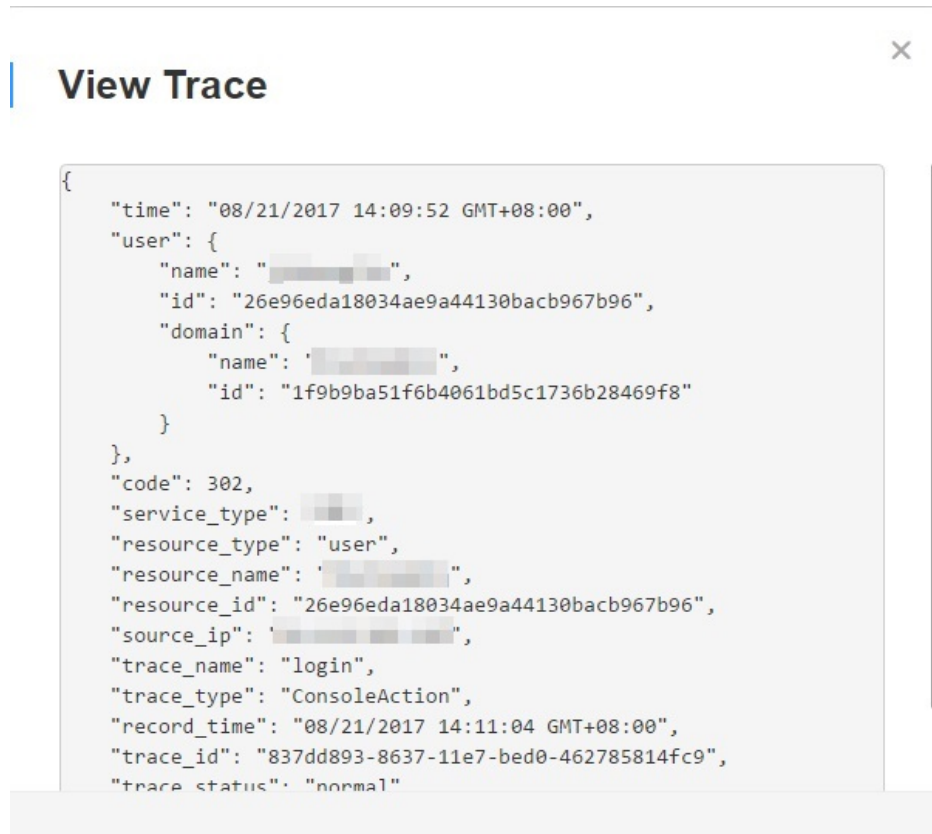
- Click  on the left of the required trace to expand its details.
- Click **View Trace**.

Figure 4-1 View Trace

CTS Log Entries

Each log entry consists of a trace in JSON format. A log entry indicates an SMN API request, including the requested operation, the date and time, operation parameters, and information about the user who sent the request. The user information is obtained from IAM.

The following example shows CTS log entries for the **CreateTopic**, **DeleteTopic**, and **UpdateTopic** actions:

```
{
  "time": "2017-02-15 14:21:50 GMT+08:00",
  "user": "xxx",
  "request": "xxx",
  "response": "xxx",
  "code": 200,
  "service_type": "SMN",
  "resource_type": "topic",
  "resource_id": "topicUrn instance",
  "source_ip": "127.0.0.1",
  "trace_name": "createTopic",
  "trace_rating": "normal",
  "trace_type": "ApiCall",
  "api_version": "2.0",
  "project_id": "tenantId instance",
  "record_time": "2017-02-15 14:21:50 GMT+08:00",
  "trace_id": "xxx"
}
```

```
}  
  
{  
  "time": "2017-02-15 14:12:15 GMT+08:00",  
  "user": "xxx",  
  "response": "xxx",  
  "code": 200,  
  "service_type": "SMN",  
  "resource_type": "topic",  
  "resource_id": "topicUrn instance",  
  "source_ip": "127.0.0.1",  
  "trace_name": "deleteTopic",  
  "trace_rating": "normal",  
  "trace_type": "ApiCall",  
  "api_version": "2.0",  
  "project_id": "tenantId instance",  
  "record_time": "2017-02-15 14:12:15 GMT+08:00",  
  "trace_id": "xxx"  
}  
  
{  
  "time": "2017-02-13 15:38:30 GMT+08:00",  
  "user": "xxx",  
  "request": "xxx",  
  "response": "xxx",  
  "code": 200,  
  "service_type": "SMN",  
  "resource_type": "topic",  
  "resource_id": "topicUrn instance",  
  "source_ip": "127.0.0.1",  
  "trace_name": "updateTopic",  
  "trace_rating": "normal",  
  "trace_type": "ApiCall",  
  "api_version": "2.0",  
  "project_id": "tenantId instance",  
  "record_time": "2017-02-13 15:38:30 GMT+08:00",  
  "trace_id": "xxx"  
}
```

5 Monitoring

5.1 Metrics

Overview

This section describes metrics reported from SMN to Cloud Eye. You can use Cloud Eye to query the metrics and alarms generated for SMN.

Namespace

SYS.SMN

Metrics

[Table 5-1](#) lists the metrics of SMN.

Table 5-1 SMN metrics

ID	Name	Description	Value	Monitored Object	Monitoring Period (Raw Data)
email_total_count	Email Subscription Pushed	Total number of email messages published to your topic	≥ 0	A single topic	5 minutes

ID	Name	Description	Value	Monitored Object	Monitoring Period (Raw Data)
email_success_rate	Rate of Email Messages Delivered	Rate of messages successfully delivered to email subscription endpoints Unit: percent	0-100%	A single topic	5 minutes
sms_total_count	SMS Messages Published	Total number of SMS messages that are attempted to be published to your topic	≥ 0	A single topic	5 minutes
sms_success_rate	Rate of SMS Message Delivered	Percentage of messages successfully delivered to SMS subscription endpoints Unit: percent	0-100%	A single topic	5 minutes
http_total_count	HTTP(S) Messages Published	Total number of HTTP or HTTPS messages published to your topics	≥ 0	A single topic	5 minutes

ID	Name	Description	Value	Monitored Object	Monitoring Period (Raw Data)
http_2xx_count	2xx Return Codes	Number of 2xx return codes when messages are successfully delivered to HTTP or HTTPS subscription endpoints	≥0	A single topic	5 minutes
http_4xx_count	4xx Error Codes	Number of 4xx error codes returned when SMN fails to deliver messages from your topics to HTTP or HTTPS subscription endpoints	≥0	A single topic	5 minutes
http_5xx_count	5xx Error Codes	Number of 5xx error codes returned when subscribers fail to receive HTTP or HTTPS messages from SMN	≥0	A single topic	5 minutes

ID	Name	Description	Value	Monitored Object	Monitoring Period (Raw Data)
http_success_rate	Rate of HTTP(S) Message Delivered	Percentage of messages successfully delivered to HTTP and HTTPS subscription endpoints Unit: percent	0-100%	A single topic	5 minutes

Dimension

Key	Value
topic_id	Topic ID

5.2 Viewing Metrics

Scenario

Cloud Eye helps you better obtain the subscription push results of your topics. This section describes how to view SMN metrics.


Prerequisites

There are topics.


NOTE

If a subscription protocol is not configured to a topic, the value of the corresponding metric to be reported to Cloud Eye is 0.

Viewing Metrics on Cloud Eye

1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click **Service List** in the upper left corner, and select **Cloud Eye**.
4. In the navigation pane on the left, choose **Cloud Service Monitoring > Simple Message Notification**.

5. Click **View Metric** in the **Operation** column to view SMN metrics.
You can view metric data of the last 1, 3, 12, or 24 hours, or last 7 days.

Hover your mouse over a graph and click  to view data for a longer time range.

 **NOTE**


It can take a period of time to obtain and transfer the monitoring data. Wait for a while and then check the data.

5.3 Configuring Alarm Rules

Scenario

By configuring alarm rules for a topic, you can customize the monitored objects and notification policies to obtain the subscription push results of the topic in a timely manner.

Procedure

1. Log in to the management console.
2. In the upper left corner of the page, click  and select the desired region and project.
3. Click **Service List** in the upper left corner, and select **Cloud Eye**.
4. In the navigation pane on the left, choose **Alarm Management > Alarm Rules**.
5. On the **Alarm Rules** page, modify an existing alarm rule or click **Create Alarm Rule** and create an alarm rule for a topic.
6. After configuring the parameters, click **Create**.

 **NOTE**



For more information about configuring alarm rules, see [Cloud Eye User Guide](#).

6 Logs

Scenarios

You can use logs when you want to know the statuses of messages published to a topic. Protocols including SMS, FunctionGraph (function), email, HTTP and HTTPS are supported. Before configuring logs, you need to interconnect SMN with Log Tank Service (LTS) and have created a log group and log stream to be associated. For details about how to configure and operate LTS, see [Log Tank Service User Guide](#).

Configuring Cloud Logs

1. Create a log group.
 - a. Log in to the management console.
 - b. Click  on the upper left to select the desired region and project.
 - c. In the upper left corner of the page, click . Select **Log Tank Service** under **Management & Governance**.
The LTS console is displayed.
 - d. In the navigation pane on the left, choose **Log Management**.
The **Log Management** page is displayed.
 - e. Click **Create Log Group**. In the displayed dialog box, enter a log group name.
 - f. Click **OK**.
2. Create a log stream.
 - a. Locate the created log group and click its name.
 - b. Click **Create Log Stream**. In the displayed dialog box, enter a name for the log stream.
 - c. Click **OK**.

Configuring Message Transmission Logs

On the SMN console, configure logs.



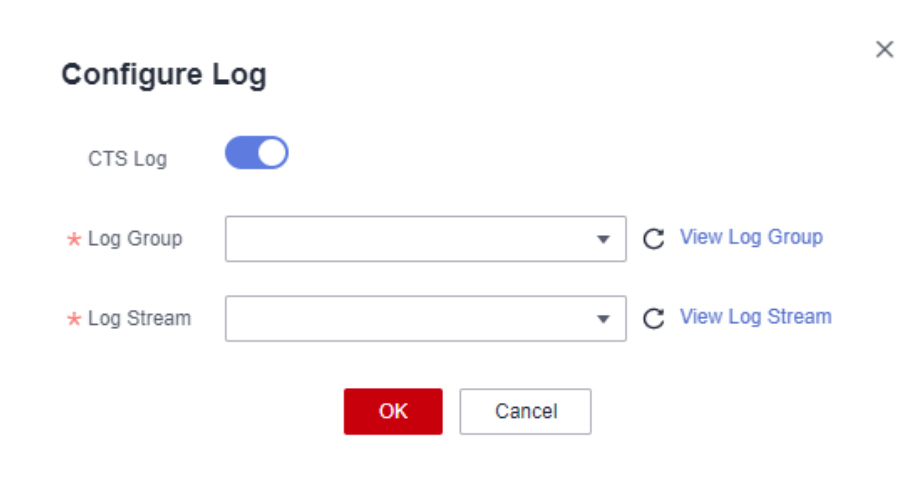
1. Set this parameter on the **Create Topic** page.
 - a. Log in to the management console.
 - b. In the upper left corner of the page, click  and select the desired region and project.
 - c. In the upper left corner of the page, click . Select **Simple Message Notification** under **Management & Governance**.
The SMN console is displayed.
 - d. In the navigation pane on the left, choose **Topic Management > Topics**.
The **Topics** page is displayed.
 - e. On the **Topics** page, click the name of the topic to be configured.
 - f. On the displayed page, click the **Message Transfer Logs** tab, and click **Configure Access Logs**.

Figure 1 Message transmission log



- g. Click **Start Logging** and select the log group and log stream that you have created on the LTS console.

Figure 2 Configuring logs



- h. Click **OK**.

Viewing Access Logs

You can view details about logs you configured.

Log format

```
{"message_id":"$message_id","project_id":"$project_id","topic_urn":"$topic_urn","subscriber_urn":"$subscriber_urn","protocol_name":"$protocol_name","endpoint":"$endpoint","status":"$status","http_code":"$http_code","create_time":"$create_time","send_time":"$send_time"}
```

The log format cannot be modified. Table 1 describes the log fields.

Table 6-1 Parameters in a FunctionGraph (function) message

Parameter	Type	Description
message_id	String	Message ID
project_id	String	Project ID
topic_urn	String	Resource identifier of a topic, which is unique
subscriber_urn	String	Resource identifier of a subscription, which is unique
protocol_name	String	Specifies the subscription protocol. (Different protocols indicate different types of endpoints to receive messages.) The following protocols are supported: Email: The endpoints are email addresses. SMS: The endpoints are phone numbers. FunctionGraph: FunctionGraph (function) transmission protocol. The endpoint is a function. FunctionStage: FunctionStage (workflow) transmission protocol. The endpoint is a function workflow. HTTP and HTTPS: The endpoints are URLs.
endpoint	String	Message receiving endpoint
status	String	Message status. The options are as follows: DELIVERED: The message has been delivered. FAIL_DELIVERED: The message fails to be sent. REJECTS: The message has been rejected. The flow control mechanism is triggered.
http_code	Integer	HTTP return code. Only HTTP/HTTPS messages are supported.

Parameter	Type	Description
create_time	String	Time when a message was created. The UTC time is in YYYY-MM-DDTHH:MM:SSZ format.
send_time	String	Specifies the time when the message was sent. The UTC time is in YYYY-MM-DDTHH:MM:SSZ format.

Example Log

```
{
  "message_id": "1ae49922602a42fc83acb9689a2eb5f4",
  "project_id": "5a9f32e4f1ec4bbe9695ff9da51c2925",
  "topic_urn": "urn:smn:regionid:5a9f32e4f1ec4bbe9695ff9da51c2925:demo",
  "subscriber_urn": "urn:smn:regionid:5a9f32e4f1ec4bbe9695ff9da51c2925:demo:b55c3c6fa7cd471b9f24818d530a8740",
  "protocol_name": "https",
  "endpoint": "https://127.0.0.1:443/https",
  "status": "DELIVERED",
  "http_code": 200,
  "create_time": "2022-11-01T00:00:00Z",
  "send_time": "2022-11-01T00:00:10Z"
}
```

The following table describes the fields in the log.

Table 6-2 Example values of fields in the log

Parameter	Example Value
message_id	1ae49922602a42fc83acb9689a2eb5f4
project_id	5a9f32e4f1ec4bbe9695ff9da51c2925
topic_urn	urn:smn:regionid:5a9f32e4f1ec4bbe9695ff9da51c2925:demo
subscriber_urn	urn:smn:regionid:5a9f32e4f1ec4bbe9695ff9da51c2925:demo:b55c3c6fa7cd471b9f24818d530a8740
protocol_name	https
endpoint	https://127.0.0.1:443/https
status	DELIVERED
http_code	200
create_time	2022-11-01T00:00:00Z
send_time	2022-11-01T00:00:10Z

7 Permissions Management

7.1 Creating a User and Granting SMN Permissions

Use [IAM](#) to implement fine-grained permissions control over your SMN resources. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing SMN resources.
- Grant only the permissions required for users to perform a specific task.
- Entrust a Huawei Cloud account or cloud service to perform efficient O&M on your SMN resources.

If your Huawei Cloud account does not require individual IAM users, skip this chapter.

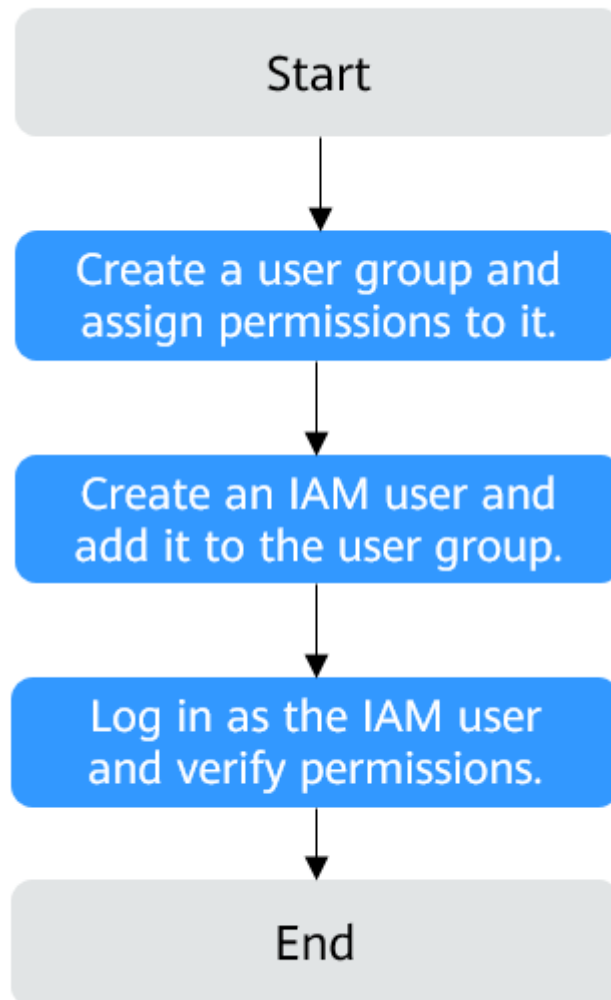
This section describes the procedure for granting permissions (see [Figure 7-1](#)).

Prerequisites

Learn about the system permissions (see [Permissions Management](#)) supported by SMN and choose policies or roles according to your requirements. For system permissions of other cloud services, see [System Permissions](#).

Process Flow

Figure 7-1 Process for granting the **SMN Administrator** permissions



1. **Create a user group and assign permissions.**
Create a user group on the IAM console and assign the **SMN Administrator** permissions to the group.
2. **Create a user and add it to the user group.**
Create a user on the IAM console and add the user to the group created in 1.
3. **Log in as the created user** and verify the **SMN Administrator** permissions.
Log in to the SMN console by using the created user, and verify that the user only has the **SMN Administrator** permissions.
 - Choose **Service List > Simple Message Notification**. On the SMN console, choose **Topic Management > Topics**, and click **Create Topic** in

- the upper right corner. If the topic is successfully created, the **SMN Administrator** permissions have already taken effect.
- Choose any other service in **Service List**. If a message appears indicating that you have insufficient permissions to access the service, the **SMN Administrator** permissions have already taken effect.

7.2 Creating SMN Custom Policies

You can create custom policies to supplement the system-defined policies of SMN. For the actions supported by custom policies, see [Permissions Policies and Supported Actions](#) in *Simple Message Notification API Reference*.

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Create a JSON policy or edit an existing one.

The following are examples of custom policies created for SMN. For details, see [Creating a Custom Policy](#).

Example SMN Custom Policies

- Example 1: allowing topic creation

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "smn:topic:create"
      ]
    }
  ]
}
```

- Example 2: denying topic deletion

A policy with only **Deny** permissions must work with other policies. If the policies assigned to a user contain both **Allow** and **Deny** actions, the **Deny** actions take precedence over the **Allow** actions.

You can assign a system policy of **SMN FullAccess** and a custom policy of denying topic deletion to the user group which the user belongs to at the same time. Thus the user can perform all operations on SMN except deleting topics. The following is an example of a deny policy:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "smn:topic:delete"
      ]
    }
  ]
}
```

- Example 3: defining multiple actions in a policy

A custom policy can contain multiple actions that belong to any global or project-level services. The following is an example policy containing actions of multiple services:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "smn:topic:create",
        "smn:tag:create",
        "smn:application:create"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elb:certificates:create",
        "elb:whitelists:create",
        "elb:pools:create",
        "elb:members:create",
        "elb:healthmonitors:create",
        "elb:l7policies:create",
        "elb:listeners:create",
        "elb:loadbalancers:create"
      ]
    }
  ]
}
```

8 Quotas

What Is Quota?

Quotas can limit the number or amount of resources available to users, such as the maximum number of ECSs or EVS disks that can be created.

If the existing resource quota cannot meet your service requirements, you can apply for a higher quota.

How Do I View My Quotas?


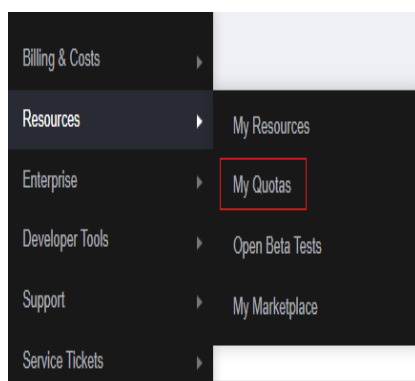
1. Log in to the management console.
2. Click  in the upper left corner and select the desired region and project.
3. In the upper right corner of the page, choose **Resources > My Quotas**.
The **Service Quota** page is displayed.

Figure 8-1 My Quotas



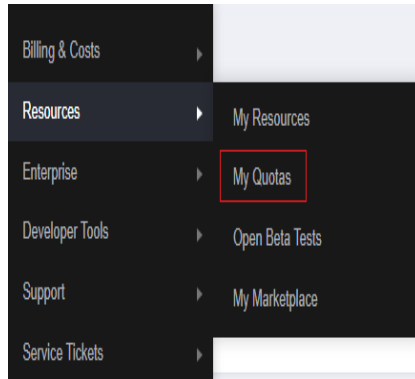
4. View the used and total quota of each type of resources on the displayed page.
If a quota cannot meet service requirements, apply for a higher quota.

How Do I Apply for a Higher Quota?

1. Log in to the management console.

2. In the upper right corner of the page, choose **Resources > My Quotas**.
The **Service Quota** page is displayed.

Figure 8-2 My Quotas



3. Click **Increase Quota** in the upper right corner of the page.

Figure 8-3 Increasing quota

Service Quota ⊙ Increase Quota

Service	Resource Type	Used Quota	Total Quota
Auto Scaling	AS group	0	
	AS configuration	0	
Image Management Service	Image	0	
Cloud Container Engine	Cluster	0	
FunctionGraph	Function	0	
	Code storage(MB)	0	
Elastic Volume Service	Disk	3	
	Disk capacity(OB)	120	
	Snapshots	4	
Storage Disaster Recovery Service	Protection group	0	
	Replication pair	0	
Cloud Server Backup Service	Backup Capacity(OB)	0	
	Backup	0	
Scalable File Service	File system	0	
	File system capacity(OB)	0	
CDN	Domain name	0	
	File URL refreshing	0	
	Directory URL refreshing	0	
	URL prefetching	0	

4. On the **Create Service Ticket** page, configure parameters as required.
In the **Problem Description** area, fill in the content and reason for adjustment.
5. After all necessary parameters are configured, select **I have read and agree to the Ticket Service Protocol and Privacy Statement** and click **Submit**.

A Appendix

A.1 JSON Message Format

Description

The JSON format allows you to specify different message content for different subscription protocols, including **Default**, **SMS**, **HTTP**, **HTTPS**, and **Email**. The message content you specify will be sent to subscription endpoints using applicable protocols.

```
{
  "default": "Dear Sir or Madam, this is a default message.",
  "email": "Dear Sir or Madam, this is an email message.",
  "http": "{ 'message': 'Dear Sir or Madam, this is an HTTP message.' }",
  "https": "{ 'message': 'Dear Sir or Madam, this is an HTTPS message.' }",
  "sms": "This is an SMS message."
}
```

It is recommended that you specify general message content for all subscription types in the **Default** protocol and enter customized content for specific protocols.

In the following example, you enter a shorter message for the SMS protocol because of the length limit on SMS messages. SMS subscribers in the topic receive the message "This is an SMS message.", while other types of subscribers (email, HTTP, and HTTPS) receive the one "Dear Sir or Madam, this is a default message."

```
{
  "sms": "This is an SMS message.",
  "default": "Dear Sir or Madam, this is a default message."
}
```

Constraints

- The content must be in JSON format.
- You must configure the **Default** protocol in the JSON message.
- The size of a JSON message cannot exceed 256 KB.

Calculation on the Size of a JSON Message

The size of a JSON message, including braces, quotation marks, spaces, line breaks, protocols, and message content, cannot exceed 256 KB. The size of a JSON message generated for each protocol may vary.

For example, message content "This is a default message." contains 26 bytes.

The system automatically adds the **Default** protocol when generating a JSON message.

```
{  
  "default": "This is a default message.",  
  "protocol1": "This is a default message.",  
  "protocol2": "This is a default message.",  
  ...  
}
```

The total number of protocols is N , including the **Default** protocol and those you selected.

The size of the message is calculated as follows:

- Three spaces in each of the N protocols: $3 \times N = 3N$ bytes
- Four quotation marks in each of the N protocols: $4 \times N = 4N$ bytes
- One colon in each of the N protocols: $1 \times N = N$ bytes
- Message content "This is a default message." in each of the N protocols: $26 \times N = 26N$ bytes
- Commas in $(N - 1)$ protocols: $1 \times (N - 1) = (N - 1)$ bytes
- Line breaks in $(N + 1)$ protocols: $1 \times (N + 1) = (N + 1)$ bytes
- Two braces: 2 bytes
- Protocol name **Default**: 7 bytes

Bytes of protocols you selected:

- **HTTP**: 4 bytes
- **HTTPS**: 5 bytes
- **Email**: 5 bytes
- **SMS**: 3 bytes

Total size = $36N + 9$ + Bytes of protocols you selected

For example, you selected the HTTP, HTTPS, and email protocols, and the message is as follows:

```
{  
  "default": "This is a default message.",  
  "email": "This is a default message.",  
  "http": "This is a default message.",  
  "https": "This is a default message."  
}
```

The system adds a **Default** protocol, and the value of N is 4. The size of this JSON message is:

- Fixed length: $36 \times 4 + 9 = 153$ bytes
- **http**: 4 bytes

- **https:** 5 bytes
- **email:** 5 bytes

The total size is 167 bytes (153 + 4 + 5 + 5 = 167).

A.2 Template Message Format

Message templates are used to publish messages with fixed content and use variables as placeholders to represent content that you can change.

The size of template message cannot exceed 256 KB. The following is an example of how to format a template when you manually type the template message content:

```
{
  "message_template_name":"confirm_message",
  "tags":{"
    "topic_urn":"urn:smn:regionId:xxxx:SMN_01"
  }
}
```

Table A-1 Parameters description and setting

Parameter	Description
message_template_name	Specifies the template name, which must be specified. You can query the template name in the template list. You must create a template of the default protocol so that SMN can send messages using the default template once it fails to match a specified protocol.
tags	Variables in the template, which are presented as JSON mappings. You can create templates for different protocols using the same template name and configure different variables in each template.

A.3 Messages Using Different Protocols

Message contents delivered to endpoints using different protocols differ.

- Email or HTTP/HTTPS endpoints will receive the message subject, content, and a link to unsubscribe.
- SMS endpoints receive only the message content.

A.4 Traffic Control over Subscription Confirmation

To prevent malicious users from harassing subscribers, SMN limits the number of subscription confirmation messages a user can send to an individual subscriber within a specified period of time. Traffic control policies apply to confirmation requests issued both from the SMN console and by API calling. Traffic control policies for different subscription protocols are as follows:

- Email: A user can send up to 20 confirmation messages within one hour or 40 within two days. When the threshold is met, SMN will not send any more confirmation messages to that email address in the next seven days. After the subscriber confirms the subscription, SMN clears the count in the traffic control policy.
- SMS: A user can send up to 10 confirmation messages within one hour or 20 within two days. When the threshold is met, SMN will not send any more confirmation messages to that phone number in the next seven days. After the subscriber confirms the subscription, SMN clears the count in the traffic control policy.
- HTTP or HTTPS: A user can send up to 200 confirmation messages within 10 minutes.

A.5 Country or Region Codes

English	Code
Afghanistan	93
Albania	355
Algeria	213
American Samoa	1684
Andorra	376
Angola	244
Anguilla	1264
Antigua and Barbuda	1268
Argentina	54
Armenia	374
Aruba	297
Australia	61
Austria	43
Azerbaijan	994
Bahamas	1242
Bahrain	973
Bangladesh	880
Barbados	1246
Belarus	375
Belgium	32

English	Code
Belize	501
Benin	229
Bermuda	1441
Bhutan	975
Bolivia	591
Bosnia and Herzegovina	387
Botswana	267
Brazil	55
British Indian Ocean Territory	246
Brunei	673
Bulgaria	359
Burkina Faso	226
Burundi	257
Cambodia	855
Cameroon	237
Canada	1
Cape Verde	238
Cayman Islands	1345
Central African Republic	236
Chad	235
Chile	56
Colombia	57
Comoros	269
Republic of the Congo	242
Democratic Republic of the Congo	243
Cook Islands	682
Costa Rica	506
Croatia	385
Curacao	599

English	Code
Cyprus	357
Czech	420
Denmark	45
Djibouti	253
Dominica	1767
Dominican Republic	1809
Ecuador	593
Egypt	20
El Salvador	503
Equatorial Guinea	240
Eritrea	291
Estonia	372
Eswatini	268
Ethiopia	251
Falkland Islands	500
Faroe Islands	298
Fiji	679
Finland	358
France	33
French Guiana	594
French Polynesia	689
Gabon	241
Gambia	220
Georgia	995
Germany	49
Ghana	233
Gibraltar	350
Greece	30
Greenland	299
Grenada	1473

English	Code
Guadeloupe	590
Guatemala	502
Guernsey	44
Guinea	224
Guinea-Bissau	245
Guyana	592
Haiti	509
Honduras	504
Hong Kong (China)	852
Hungary	36
Iceland	354
India	91
Indonesia	62
Iraq	964
Ireland	353
Isle of Man	44
Israel	972
Italy	39
Ivory Coast	225
Jamaica	1876
Japan	81
Jersey	44
Jordan	962
Kazakhstan	7
Kenya	254
Kuwait	965
Kyrgyzstan	996
Laos	856
Latvia	371
Lebanon	961

English	Code
Lesotho	266
Liberia	231
Libya	218
Liechtenstein	423
Lithuania	370
Luxembourg	352
Macao (China)	853
North Macedonia	389
Madagascar	261
Malawi	265
Malaysia	60
Maldives	960
Mali	223
Malta	356
Marshall Islands	692
Martinique	596
Mauritania	222
Mauritius	230
Mexico	52
Moldova	373
Monaco	377
Mongolia	976
Montenegro	382
Montserrat	1664
Morocco	212
Mozambique	258
Myanmar	95
Namibia	264
Nauru	674
Nepal	977

English	Code
Netherlands	31
New Caledonia	687
New Zealand	64
Nicaragua	505
Niger	227
Nigeria	234
Niuē	683
Norfolk Island	672
Norway	47
Oman	968
Pakistan	92
Palau	680
Palestine	970
Panama	507
Papua New Guinea	675
Paraguay	595
Peru	51
Philippines	63
Poland	48
Portugal	351
Qatar	974
Réunion	262
Romania	40
Rwanda	250
Saint Kitts and Nevis	1869
Saint Lucia	1758
Saint Pierre and Miquelon	508
Saint Vincent and the Grenadines	1784
Samoa	685

English	Code
San Marino	378
Sao Tome and Principe	239
Saudi Arabia	966
Senegal	221
Serbia	381
Seychelles	248
Sierra Leone	232
Singapore	65
Sint Maarten (Dutch Part)	1721
Slovakia	421
Slovenia	386
Solomon Islands	677
Somalia	252
South Africa	27
South Korea	82
Spain	34
Sri Lanka	94
Suriname	597
Sweden	46
Switzerland	41
Taiwan (China)	886
Tajikistan	992
Tanzania	255
Thailand	66
Timor-Leste	670
Togo	228
Tonga	676
Trinidad and Tobago	1868
Tunisia	216
Türkiye	90

English	Code
Turkmenistan	993
Turks and Caicos Islands	1649
Tuvalu	688
Uganda	256
Ukraine	380
United Arab Emirates	971
United Kingdom	44
United States	1
Uruguay	598
Uzbekistan	998
Vanuatu	678
Venezuela	58
Vietnam	84
Virgin Islands, British	1284
Wallis and Futuna	681
Yemen	967
Zambia	260
Zimbabwe	263

A.6 Mappings Between SMN Actions and APIs

Table A-2 Mappings between SMN actions and APIs

Action	API	Function
SMN:UpdateTopic	UpdateTopic	Modify the topic. Only the display_name value can be changed.
SMN>DeleteTopic	DeleteTopic	Delete a topic and its subscribers. If a topic is deleted, any pending messages may fail to send to the topic subscribers.
SMN:QueryTopicDetail	QueryTopicDetail	Query details about a topic.

Action	API	Function
SMN:ListTopicAttributes	ListTopicAttributes	Query topic attributes.
SMN:UpdateTopicAttribute	UpdateTopicAttribute	Modify an attribute of a topic.
SMN>DeleteTopicAttributes	DeleteTopicAttributes	Delete all attributes of a topic.
SMN>DeleteTopicAttributeByName	DeleteTopicAttributeByName	Delete an attribute of a specified topic.
SMN:ListSubscriptionsByTopic	ListSubscriptionsByTopic	Query the subscription list of a specified topic by page. The list is sorted by time when the subscriptions are added in ascending order. You can specify values of offset and limit . If no subscription has been added, an empty list is returned.
SMN:Subscribe	Subscribe	Add a subscription to a specified topic and send a confirmation message to the subscriber. After confirming the subscription, the subscriber can receive notification messages published to the topic.
SMN:Unsubscribe	Unsubscribe	Delete a subscription. This operation requires identity authentication. Only the subscriber or the topic owner can delete a subscription.
SMN:Publish	Publish	Publish messages to a topic. After a message ID is returned, the message has been saved and is to be delivered to subscribers of the topic. The message form varies depending on the protocol of each subscription.

A.7 Restrictions on SMS Messaging

- There are restrictions on sending SMS messages. For details, see section [Sending Rules and Restrictions](#) in *Message & SMS Service Overview*.

A.8 HTTP/HTTPS Messages

A.8.1 Introduction

HTTP/HTTPS messages can be classified as management messages and service messages. The former includes subscription messages and subscription cancellation messages, while the latter includes notification messages. An HTTPS/HTTPS message is composed of a message header and body, which are illustrated in detail in this topic.

A.8.2 HTTP or HTTPS Message Format

Scenarios

⚠ CAUTION

When receiving HTTP or HTTPS messages sent by SMN, refer to the industry standards for the common name (CN) of the terminal certificate. Some special characters may cause HTTPS message sending failures.

This section describes the format of messages sent to HTTP or HTTPS endpoints. You can identify messages based on message types in the headers. HTTP/HTTPS message types include: subscription confirmation messages, notification messages, and subscription cancellation messages. POST is used for HTTP/HTTPS messages.

The header of an SMN HTTP/HTTPS message contains the following parameters: **X-SMN-MESSAGE-TYPE**, **X-SMN-MESSAGE-ID**, **X-SMN-TOPIC-URN**, and **X-SMN-SUBSCRIPTION-URN**.

Table A-3 HTTP header fields

Parameter	Description
X-SMN-MESSAGE-TYPE	Indicates the message type, which can be: <ul style="list-style-type: none">• SubscriptionConfirmation• SubscriptionConfirmation• SubscriptionConfirmation
X-SMN-MESSAGE-ID	Indicates the unique message ID.
X-SMN-TOPIC-URN	Indicates the URN of the topic the message belongs to.
X-SMN-SUBSCRIPTION-URN	Identifies the subscription endpoint. This parameter is required only when messages are pushed over HTTP/HTTPS and when you cancel your HTTP/HTTPS subscriptions.

 NOTE

The requirements for HTTP header fields are described as follows:

- According to section 3.2 in RFC 7230, header fields should be case insensitive.
- According to section 8.1.2 in RFC 7540, header fields in HTTP/2 should be lowercase.
- Custom HTTP header fields must comply with the above two requirements for SMN.
- You are advised to use case-insensitive letters to obtain HTTP header fields.

HTTP/HTTPS Subscription Confirmation Message Format

After you add an HTTP/HTTPS endpoint, SMN sends a subscription confirmation message to the subscriber. The message body is composed of JSON character strings. The subscriber must obtain the subscription URL (**subscribe_url**) to confirm the subscription. [Table A-4](#) describes the JSON field in detail.

Table A-4 HTTP/HTTPS subscription confirmation message body

Parameter	Description
type	Indicates the message type, which is SubscriptionConfirmation .
signature	Indicates the signature information. The signature includes the message , message_id , subscribe_url , timestamp , topic_urn , and type fields. For details about signature verification, see Message Signature Verification .
topic_urn	Indicates the URN of the topic the message belongs to.
message_id	Indicates the unique message ID.
signature_version	Indicates the signature version, which is V1 .
message	Indicates the message content.
subscribe_url	Indicates the URL to be accessed for subscription confirmation.
signing_cert_url	Indicates the certificate URL for a message signature. It can be directly accessed without authentication.
timestamp	Indicates the time stamp when the message was initially sent.

The following is an example HTTP/HTTPS subscription confirmation message:

```
{
  "signature": "ViE96uGbBkl
+S8eWqgebi5KdmRht2U8+Rs88yuyMHq1k4h3JkcdZ6HCqTqdpJ8nrLcdqETyyEIOQyTszJdU05z
+LhfE8jerCCdSbL4zeInVkydHh0kcCRWmORye0/EuQ/gLC1UIXwvUsqbUCPnBRhNFXOeXMOPPCzK
+d04xjy4QHd1H/bHxgsY3AlTe0gCFT068Zru7OK6w9aQaY44mXnN3OWGmBmoHyFab5TRXLSQNz/9u/
Vj646cQMMal0PPQ30QzGYD0MtgDZi12m8jMTHAnMkTEcbLaEgaqmaoEnATSpEcspFKNXv2skwk7rsVakMOI
SpMH3+qC6RzhETA2A==",
  "topic_urn": "urn:smn:region01:0553db98c800d5192f9bc01232b89622:vpc_status_report_topic",
  "message_id": "d86c201092574e71a3ca85826652c58b",
```

```

"signature_version": "v1",
"type": "SubscriptionConfirmation",
"message": "You are invited to subscribe to topic:
urn:smn:region01:0553db98c800d5192f9bc01232b89622:vpc_status_report_topic. To confirm this
subscription, please visit the subscribe_url included in this message. The subscribe_url is valid only within 48
hours.",
"subscribe_url": "https://console.xxx.com/smn/subscription/unsubscribe?
subscription_urn=urn:smn:region01:0553db98c800d5192f9bc01232b89622:vpc_status_report_topic:653e212a
43884f7188ca656c537e31ce",
"signing_cert_url": "https://smn.cn-north-9.myhuaweicloud.com/smn/SMN_cn-
north-9_94f60ccdfbee4588aa4d555935a56ba3.pem",
"timestamp": "2019-08-12T22:40:56Z"
}

```

HTTP/HTTPS Notification Message Format

After an HTTP/HTTPS subscriber confirms the subscription, the subscriber can receive notification messages published to the topic. The notification message body is composed of JSON character strings, which are described in [Table A-5](#).

Table A-5 HTTP/HTTPS notification message body

Parameter	Description
type	Indicates the message type, which is Notification .
signature	Indicates the signature information. The signature includes the message , message_id , subject , timestamp , topic_urn , and type fields. If the subject field is empty, the signature is not verified. For details about signature verification, see Message Signature Verification .
subject	Indicates the message subject.
topic_urn	Indicates the URN of the topic the message belongs to.
message_id	Indicates the unique message ID.
signature_version	Indicates the signature version, which is V1 .
message	Indicates the message content.
unsubscribe_url	Indicates the URL for canceling a subscription.
signing_cert_url	Indicates the certificate URL for generating the message signature.
timestamp	Indicates the time stamp when the message was initially sent.

The following is an example HTTP/HTTPS notification message:

```

{
  "signature": "ViE96uGbbKl
+S8eWqgebi5KdmRht2U8+Rs88yuyMHq1k4h3jUkcDZ6HCqTqdpJ8nrLcdqETyyEiOQyTszJdU05z
+LhfE8jerCCdSbL4zeInVkydHh0kcCRWmORye0/EuQ/gLC1UIXwvUsqbUCPnBRhNFXOeXMOPPCzK
+d04xjy4QHd1H/bHxgsY3AlTe0gCFT068Zru7OK6w9aQaY44mXnN3OWGmBmoHyFab5TRXLSQNz/9u/
Vj646cQMMaiOPPPQ30QzGYD0MtzgDZi12m8jMTHAnMkTEcbLaEgaqmaoEnATSpEcspFKNXv2skwk7rsVakMOI
SpMH3+qC6RzhETA2A==",

```



```

"topic_urn": "urn:smn:region01:0553db98c800d5192f9bc01232b89622:vpc_status_report_topic",
"message_id": "d86c201092574e71a3ca85826652c58b",
"signature_version": "v1",
"type": "Notification",
"message": "{\"enterpriseProjectId\": \"0\", \"eventTime\": \"2019-08-12 22:40:55.040632\",
\"chargingMode\": \"postPaid\", \"cloudserviceType\": \"xxx.service.type.bandwidth\", \"eventType\": 1,
\"regionId\": \"region01\", \"tenantId\": \"057eefe55400d2742f8cc0017870ceef\", \"resourceType\":
\"xxx.resource.type.bandwidth\", \"resourceSpecCode\": \"19_bgp\", \"resourceSize\": 10, \"resourceId\":
\"e091f1b1-08ef-4e2b-a27e-f85e4c19026a\", \"resourceSizeMeasureId\": 15, \"resourceName\":
\"elbauto_2019_08_13_06_40_46\"}",
"unsubscribe_url": "https://console.xxx.com/smn/subscription/unsubscribe?
subscription_urn=urn:smn:region01:0553db98c800d5192f9bc01232b89622:vpc_status_report_topic:653e212a
43884f7188ca656c537e31ce",
"signing_cert_url": "https://smn.cn-north-9.myhuaweicloud.com/smn/SMN_cn-
north-9_94f60ccdfbee4588aa4d555935a56ba3.pem",
"timestamp": "2019-08-12T22:40:56Z"
}

```

HTTP/HTTPS Subscription Cancellation Message Format

After an HTTP/HTTPS subscription is canceled, the subscriber receives a subscription cancellation message sent by SMN. The message body is composed of JSON character strings, which are described in [Table A-6](#).

Table A-6 Parameters of HTTP/HTTPS subscription cancellation message format

Parameter	Description
type	Indicates the message type. Its value is UnsubscribeConfirmation .
signature	Indicates the signature information. The signature includes the message , message_id , subscribe_url , timestamp , topic_urn , and type fields. For details about signature verification, see Message Signature Verification .
topic_urn	Indicates the URN of the topic the message belongs to.
message_id	Indicates the unique message ID.
signature_version	Indicates the signature version, which is V1 .
message	Indicates the message content.
subscribe_url	Indicates the URL for a re-subscription.
signing_cert_url	Indicates the certificate URL for generating the message signature.
timestamp	Indicates the time stamp when the message was initially sent.

The following is an example HTTP/HTTPS message for canceling a subscription:

```

{
  "signature": "ViE96uGbBkl
+S8eWqgebI5KdmRht2U8+Rs88yuyMHq1k4h3jUkcDZ6HCqTqdpJ8nrLcdqETyyEiOQyTszJdU05z
+LhfE8jerCCdSbL4zeInVkydHh0kcCRWmORye0/EuQ/gLC1UIXwwUsqbUCPnBRhNFXOeXMOppCzK
+d04xjy4QHd1H/bHxgsY3AlTe0gCFT068Zru7OK6w9aQaY44mXnN3OWGmBmoHyFab5TRXLSQNz/9u/

```

```
Vj646cQMMaI0PPQ30QzGYD0MtzgDZi12m8jMTHAnMkTEcbLaEgaqmaoEnATSpEcspFKNXv2skwk7rsVakMOI  
SpMH3+qC6RzhETA2A==",  
  "topic_urn": "urn:smn:region01:0553db98c800d5192f9bc01232b89622:vpc_status_report_topic",  
  "message_id": "d86c201092574e71a3ca85826652c58b",  
  "signature_version": "v1",  
  "type": "UnsubscribeConfirmation",  
  "message": "You are unsubscribed from topic:  
urn:smn:region01:0553db98c800d5192f9bc01232b89622:vpc_status_report_topic. To subscribe to this topic  
again, please visit the subscribe_url included in this message. The subscribe_url is valid only within 48  
hours.",  
  "subscribe_url": "https://console.xxx.com/smn/subscription/unsubscribe?  
subscription_urn=urn:smn:region01:0553db98c800d5192f9bc01232b89622:vpc_status_report_topic:653e212a  
43884f7188ca656c537e31ce",  
  "signing_cert_url": "https://smn.cn-north-9.myhuaweicloud.com/smn/SMN_cn-  
north-9_94f60ccdfbee4588aa4d555935a56ba3.pem",  
  "timestamp": "2019-08-12T22:40:56Z"  
}
```

A.8.3 Message Signature Verification

Scenarios

To ensure message security, SMN provides signature authentication for HTTP/HTTPS subscription confirmation messages, subscription cancellation messages, and notification messages. After you receive HTTP/HTTPS messages, check them based on the signatures.

Procedure

After receiving an HTTP/HTTPS message, check it with the following procedure:

1. Verify the key-value pairs (which vary depending on the message type) contained in the message signature. For details, see [Signature Strings for Different Message Types](#).
2. Download the X509 certificate from the certificate URL (**signing_cert_url**) contained in the message.

NOTE

The request to download the certificate is always sent over HTTPS. When you download a certificate, verify the identity of the certificate server.

3. Extract the public key from the X509 certificate for verifying the message reliability and integrity.
4. Determine which method will be used to verify the signature based on the message type (the **type** field in the message).
5. Create signature strings. Obtain the signature parameters from the message and sort them in alphabetical order. Each parameter occupies a line, with its value following in the next line.

Signature Strings for Different Message Types

1. Notification messages
 - A notification message signature must contain the following parameters (if **subject** is left blank, omit **subject** in the signature):

```
message  
message_id  
subject  
timestamp
```

```
topic_urn  
type
```

- Example signature information for a notification message

```
message  
My test message  
message_id  
88c726942175432bac921eafd0036163  
subject  
demo  
timestamp  
2016-08-15T07:29:16Z  
topic_urn  
urn:smn:regionId:74dc9e44d0cc4573adfce91cdfdd3ba9:xxxx  
type  
Notification
```

NOTE

Each parameter occupies a line and its value follows in the next line.

2. Subscription confirmation and subscription cancellation messages

- A subscription confirmation or subscription cancellation message signature must contain the following parameters:

```
message  
message_id  
subscribe_url  
timestamp  
topic_urn  
type
```

- Example signature information for a subscription confirmation message

```
message  
You are invited to subscribe to topic:  
urn:smn:regionId:d91989905b8449b896f3a4f0ad57222d:demo. To confirm this subscription,  
please visit the following SubscribeURL in this message.  
message_id  
def5c309cbff44d5a870787ed937edf8  
subscribe_url  
https://IP_address/smn/subscription/confirm?Region ID&Token&Topic URN:demo  
timestamp  
2016-08-15T07:29:16Z  
topic_urn  
urn:smn:regionId:d91989905b8449b896f3a4f0ad57222d:demo  
type  
SubscriptionConfirmation
```

NOTE

Each parameter occupies a line and its value follows in the next line.

A.8.4 Sample Code

Java

Verify **signing_cert_url**, **signature** that obtained in [HTTP or HTTPS Message Format](#), and **message** (contained in the message signature) to check the message validity, as shown in the following:

```
private static void isMessageValid(String signing_cert_url,  
    String signature, Map<String, String> message) {  
    InputStream in = null;  
    try {  
        URL url = new URL(signing_cert_url);  
        in = url.openStream();  
        CertificateFactory cf = CertificateFactory.getInstance("X.509");
```

```
X509Certificate cert = (X509Certificate) cf.generateCertificate(in);
Signature sig = Signature.getInstance(cert.getSigAlgName());
sig.initVerify(cert.getPublicKey());
sig.update(buildSignMessage(message).getBytes("UTF-8"));
byte[] sigByte = Base64.getDecoder().decode(signature);
if (sig.verify(sigByte)) {
    System.out.println("Verify success");
} else {
    System.out.println("Verify failed");
}
} catch (Exception e) {
    throw new SecurityException("Verify method failed.", e);
} finally {
    if (in != null) {
        try {
            in.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
}
```

NOTE

If your Java version is earlier than 8, use the third-party package **commons-codec.jar** to perform Base64 decoding, and replace **byte[] sigByte = Base64.getDecoder().decode(signature);** with **byte[] sigByte = Base64.decodeBase64(signature);** in the preceding code.

The following is an example of the code to create the message verification signature:

```
private static String buildSignMessage(Map<String,String> msg) {
    String type = msg.get("type");
    String message = null;
    if ("Notification".equals(type)){
        message = buildNotificationMessage(msg);
    } else if ("SubscriptionConfirmation".equals(type) ||
        "UnsubscribeConfirmation".equals(type)){
        message = buildSubscriptionMessage(msg);
    }
    return message;
}

private static String buildSubscriptionMessage(Map<String, String> msg) {
    String stringMessage = "message\n";
    stringMessage += msg.get("message") + "\n";
    stringMessage += "message_id\n";
    stringMessage += msg.get("message_id") + "\n";
    stringMessage += "subscribe_url\n";
    stringMessage += msg.get("subscribe_url") + "\n";
    stringMessage += "timestamp\n";
    stringMessage += msg.get("timestamp") + "\n";
    stringMessage += "topic_urn\n";
    stringMessage += msg.get("topic_urn") + "\n";
    stringMessage += "type\n";
    stringMessage += msg.get("type") + "\n";
    return stringMessage;
}

private static String buildNotificationMessage(Map<String, String> msg)
{
    String stringMessage = "message\n";
    stringMessage += msg.get("message").toString() + "\n";
    stringMessage += "message_id\n";
    stringMessage += msg.get("message_id").toString() + "\n";
    if (msg.get("subject") != null){
        stringMessage += "subject\n";
    }
}
```

```
        stringMessage += msg.get("subject").toString() + "\n";
    }
    stringMessage += "timestamp\n";
    stringMessage += msg.get("timestamp").toString() + "\n";
    stringMessage += "topic_urn\n";
    stringMessage += msg.get("topic_urn").toString() + "\n";
    stringMessage += "type\n";
    stringMessage += msg.get("type").toString() + "\n";
    return stringMessage;
}
```

Node.js

```
const fs = require('fs');
const crypto = require('crypto');
const jsrsag = require('jsrsasign');

/**
 * Message signature verification
 * @param pemFile: path for storing the signature file (path for storing the certificate downloaded to your
 local computer)
 * @param signature: signature to be verified
 * @param message: content of the message to be verified
 * @returns {boolean} true: The signature passes the verification. false: The signature fails the verification.
 */
function verifyMessage(pemFile, signature, message) {
    const pubPem = fs.readFileSync(pemFile);
    const verify = crypto.createVerify(signatureAlgorithm(pubPem));
    verify.update(buildSignMessage(message));
    const verifyResult = verify.verify(pubPem, signature, 'base64');
    if (verifyResult) {
        console.log("verify success");
        return true;
    } else {
        console.log('verify failed, result: ' + verifyResult);
        return false;
    }
}

/**
 * Obtain the signature algorithm from the certificate.
 */
function signatureAlgorithm(pubPem) {
    const certObject = new jsrsag.X509();
    certObject.readCertPEM(pubPem.toString());
    let algorithm = certObject.getSignatureAlgorithmField();
    if (algorithm.split('with').length > 1) {
        algorithm = algorithm.split('with')[1] + '-' + algorithm.split('with')[0];
    }
    return algorithm;
}

function buildSignMessage(msg) {
    const type = msg.type;
    let message = "";
    if (type === 'Notification') {
        message = buildNotificationMessage(msg);
    } else if (type === 'SubscriptionConfirmation') {
        message = buildSubscriptionMessage(msg);
    }
    return message;
}

function buildNotificationMessage(msg) {
    let signMessage = 'message\n' + msg.message + '\n';
    signMessage += 'message_id\n' + msg.message_id + '\n';
    if (msg.subject) {
        signMessage += 'subject\n' + msg.subject + '\n';
    }
}
```

```
signMessage += 'timestamp\n' + msg.timestamp + '\n';
signMessage += 'topic_urn\n' + msg.topic_urn + '\n';
signMessage += 'type\n' + msg.type + '\n';
return signMessage;
}

function buildSubscriptionMessage(msg) {
    let signMessage = 'message\n' + msg.message + '\n';
    signMessage += 'message_id\n' + msg.message_id + '\n';
    signMessage += 'subscribe_url\n' + msg.subscribe_url + '\n';
    signMessage += 'timestamp\n' + msg.timestamp + '\n';
    signMessage += 'topic_urn\n' + msg.topic_urn + '\n';
    signMessage += 'type\n' + msg.type + '\n';
    return signMessage;
}
```

 **NOTE**

The sample code has passed the test on Nodejs v14.17.5.

Go

```
package demo

import (
    "bytes"
    "crypto"
    "crypto/rsa"
    "crypto/x509"
    "encoding/base64"
    "encoding/json"
    "encoding/pem"
    "fmt"
    "io/ioutil"
)

type Message struct {
    Signature      string `json:"signature"`
    Subject        *string `json:"subject"`
    TopicUrn       string `json:"topic_urn"`
    MessageId      string `json:"message_id"`
    SignatureVersion string `json:"signature_version"`
    Type           string `json:"type"`
    Message        string `json:"message"`
    SubscribeUrl   string `json:"subscribe_url"`
    UnsubscribeUrl string `json:"unsubscribe_url"`
    SigningCertUrl string `json:"signing_cert_url"`
    Timestamp      string `json:"timestamp"`
}

func VerifyMessage(pemFile string, message string) bool {
    msg := Message{}
    err := json.Unmarshal([]byte(message), &msg)
    if err != nil {
        fmt.Println("Convert json to struct failed")
        return false
    }
    pemContent, err := ioutil.ReadFile(pemFile)
    if err != nil {
        fmt.Println("Read pem file failed")
        return false
    }
    certDerblock, _ := pem.Decode(pemContent)
    if certDerblock == nil {
        fmt.Println("Decode pem file failed")
        return false
    }
    cert, err := x509.ParseCertificate(certDerblock.Bytes)
    if err != nil {
```

```
    fmt.Println("Parse cert failed")
    return false
}

msgString := buildMessage(&msg)
msgHash := crypto.SHA256.New()
msgHash.Write([]byte(msgString))
msgHashSum := msgHash.Sum(nil)

decodeSign, _ := base64.StdEncoding.DecodeString(msg.Signature)

publicKey := cert.PublicKey.(*rsa.PublicKey)
err = rsa.VerifyPKCS1v15(publicKey, crypto.SHA256, msgHashSum, decodeSign)
if err != nil {
    fmt.Println("Verify failed")
    return false
} else {
    fmt.Println("Verify success")
    return true
}
}

func buildMessage(msg *Message) string {
    if msg.Type == "Notification" {
        return buildNotificationMessage(msg)
    } else if msg.Type == "SubscriptionConfirmation" || msg.Type == "UnsubscribeConfirmation" {
        return buildSubscriptionMessage(msg)
    }
    return ""
}

func buildNotificationMessage(msg *Message) string {
    buf := bytes.Buffer{}
    buf.WriteString("message\n" + msg.Message + "\n")
    buf.WriteString("message_id\n" + msg.MessageId + "\n")
    //The Subject field does not exist in msg, and this issue needs to be addressed.
    if msg.Subject != nil {
        buf.WriteString("subject\n" + *msg.Subject + "\n")
    }
    buf.WriteString("timestamp\n" + msg.Timestamp + "\n")
    buf.WriteString("topic_urn\n" + msg.TopicUrn + "\n")
    buf.WriteString("type\n" + msg.Type + "\n")
    return buf.String()
}

func buildSubscriptionMessage(msg *Message) string {
    buf := bytes.Buffer{}
    buf.WriteString("message\n" + msg.Message + "\n")
    buf.WriteString("message_id\n" + msg.MessageId + "\n")
    buf.WriteString("subscribe_url\n" + msg.SubscribeUrl + "\n")
    buf.WriteString("timestamp\n" + msg.Timestamp + "\n")
    buf.WriteString("topic_urn\n" + msg.TopicUrn + "\n")
    buf.WriteString("type\n" + msg.Type + "\n")
    return buf.String()
}
}
```

 **NOTE**

The sample code has passed the test on Go 11.5

B Change History

Released On	Description
2023-06-30	This issue is the seventh official release, which incorporates the following changes: <ul style="list-style-type: none">• Added Monitoring.• Added Adding, Modifying, or Deleting a Topic Tag.
2023-01-31	This issue is the sixth official release, which incorporates the following change: Added Logs .
2022-03-30	This issue is the fifth official release, which incorporates the following changes: <ul style="list-style-type: none">• Removed Application from the supported subscription protocols.• Removed DMS from the supported subscription protocols.
2020-09-30	This issue is the fourth official release, which incorporates the following change: Optimized the entire document.
2019-02-28	This issue is the third official release, which incorporates the following changes: <ul style="list-style-type: none">• Added the "Mobile Application Management" section.• Added the Application protocol.
2018-08-30	This issue is the second official release, which incorporates the following changes: <ul style="list-style-type: none">• Added the SMS subscription protocol.• Added the DMS subscription protocol.
2017-12-31	This issue is the first official release.