

Simple Message Notification

User Guide

Issue 01
Date 2024-04-15



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.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <https://www.huawei.com>

Email: support@huawei.com

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 Overview	1
1.1 Simple Message Notification	1
1.2 Service Advantages	1
1.3 Application Scenarios	2
1.4 Accessing and Using SMN	3
1.5 Permissions	3
1.6 SMN and Other Services	5
1.7 Concepts	5
1.8 Region and AZ	6
2 Getting Started	8
2.1 Publishing a Message	8
3 Topic Management	14
3.1 Creating a Topic	14
3.2 Modifying the Display Name of a Topic	15
3.3 Configuring Topic Policies	15
3.3.1 Basic Mode	15
3.4 Adding a Subscription to a Topic	17
3.5 Publishing a Message	18
3.5.1 Introduction	18
3.5.2 Publishing a Text Message	18
3.5.3 Publishing a JSON Message	19
3.5.4 Publishing a Template Message	21
3.6 Deleting a Topic	23
4 Subscription Management	24
4.1 Adding a Subscription	24
4.2 Requesting Subscription Confirmation	26
4.3 Canceling a Subscription	26
4.4 Deleting a Subscription	27
5 Message Template Management	29
6 Permissions Management	32
6.1 Creating a User and Granting SMN Permissions	32

7 Quotas	35
8 FAQs	36
8.1 What Are the Advantages of SMN?	36
8.2 What Protocols Does SMN Support?	36
8.3 What Are the Requirements for an SMN Topic Name?	36
8.4 How Many Topics Can I Create?	36
8.5 How Many Subscriptions Can Be Added to a Topic?	36
8.6 How Many Messages Can Be Published to a Topic?	37
8.7 How Many Message Templates Can I Create?	37
8.8 Can I Add Subscriptions Using Multiple Protocols to a Topic?	37
8.9 Can a Topic Creator Change Subscription Protocols in a Topic?	37
8.10 Can I Change a Subscription Endpoint for a Topic?	37
8.11 Can I Delete a Published Message?	37
8.12 Does SMN Ensure That Messages Are Received by Subscription Endpoints?	37
8.13 Will a Subscriber Receive the Same Message Multiple Times?	38
8.14 Why Do Subscribers Fail to Receive Messages After I Publish Messages to a Topic?	38
8.15 What Can I Do When Pictures in an Email Message Cannot Be Displayed?	38
8.16 How Do I Obtain My Account ID?	40
A Appendix	41
A.1 JSON Message Format	41
A.2 Template Message Format	43
A.3 Messages Using Different Protocols	43
A.4 Traffic Control over Subscription Confirmation	43
A.5 Mappings Between SMN Actions and APIs	44
A.6 HTTP/HTTPS Messages	45
A.6.1 Introduction	45
A.6.2 HTTP or HTTPS Message Format	45
A.6.3 Message Signature Verification	49
A.6.4 Sample Code	51
B Change History	56

1 Overview

1.1 Simple Message Notification

Simple Message Notification (SMN) is a reliable and flexible large-scale message notification service. It enables you to efficiently send messages to various endpoints, such as phone numbers, and email addresses.

SMN offers a publish/subscribe model to achieve one-to-multiple message subscriptions and notifications in a variety of message types. SMN involves two roles: publisher and subscriber. A publisher publishes messages to a topic, and SMN then delivers the messages to subscribers in the topic. The subscribers can be email addresses, phone numbers, and URLs.

A topic is a collection of messages and a logical access point, through which the publisher and the subscriber can interact with each other. Each topic has a unique name. The topic creator can configure topic policies to grant other users or cloud services permissions to perform certain operations to a topic, for example, querying subscriptions or publishing messages.

1.2 Service Advantages

SMN has the following advantages over any traditional messaging systems.

Table 1-1 SMN advantages

Item	SMN	Traditional Messaging System
Simplicity	SMN provides three basic APIs to create topics, add subscriptions, and publish messages and can be quickly integrated with your services. It enables you to send messages in substantial quantity and do not require highly skilled development.	A self-developed messaging system is expensive and takes long time to be integrated with your services. Its APIs are complex and hard to use.
Stability and reliability	SMN stores messages in multiple data centers and supports transparent topic migration. Once a message fails to deliver, SMN saves it in a message queue and tries to deliver it again. If one service node is faulty, your requests are automatically processed by another available node.	A traditional messaging system cannot achieve the stability and reliability required by critical services and does not provide measures to ensure service continuity.
Multiple message types	You publish a message once, and SMN delivers it to endpoints in various message types.	You need to develop separate messaging systems in multiple types to send SMS message, email, HTTP, or HTTPS notifications.
Security	SMN isolates data based on topics and prevents any unauthorized users from accessing message queues, thereby protecting your service data.	Service data is potentially exposed to unauthorized access due to lack of effective protection mechanisms.

1.3 Application Scenarios

- **System notifications**

After events or alarms are triggered, SMN can send notifications to specified users by email, SMS message, or HTTP/HTTPS message. For example, Cloud Trace Service (CTS) detects key cloud service operations and uses SMN to notify you and other users.

- **Integrating with cloud services**

SMN can function as a message middleware to directly connect cloud services, improving service efficiency. For example, Cloud Eye does not have to be integrated with Object Storage Service (OBS) to interact with each other.

Instead, they can be connected by SMN, so faults in one service will not affect the other.

- **Off-peak traffic control**

If there is a discrepancy between processing capabilities of the upstream and downstream systems, SMN can cache data to reduce downstream pressure to reduce breakdowns, enhance availability, and mitigate complexity in the system.

1.4 Accessing and Using SMN

You can access the SMN service using a web-based management console and HTTPS-based APIs.

- **Management console**

The management console is a web user interface for you to manage your computing, storage, and other cloud resources. You can log in the management console and select **Simple Message Notification** on the homepage to switch to the SMN console.

- **APIs**

If you want to integrate SMN into a third-party system for secondary development, you can access SMN using APIs. For details, see *Simple Message Notification API Reference*.

1.5 Permissions

You can use Identity and Access Management (IAM) to manage SMN permissions and control access to your resources. IAM provides identity authentication, permissions management, and access control.

You can create IAM users for your employees, and assign permissions to these users on a principle of least privilege (PoLP) basis to control their access to specific resource types. For example, you can create IAM users for software developers and assign specific permissions to allow them to use SMN resources but prevent them from being able to delete resources or perform any high-risk operations.

If your account does not require individual IAM users for permissions management, skip this section.

IAM can be used free of charge. You pay only for the resources in your account.

For more information about IAM, see *Identity and Access Management Service User Guide*.

SMN Permissions

By default, new IAM users do not have any permissions assigned. To assign permissions to these new users, add them to one or more groups, and attach permissions policies or roles to these groups.

SMN is a project-level service deployed and accessed in specific physical regions. When assigning SMN permissions to a user group, specify region-specific projects where the permissions will take effect. If you select **All projects**, the permissions

will be granted for all region-specific projects. When accessing SMN, the users need to switch to a region where they have been authorized to use this service.

Table 1-2 lists all system-defined roles supported by SMN.

Table 1-2 System-defined roles supported by SMN

Role	Description	Type	Dependency
SMN Administrator	Has all permissions for SMN resources.	System-defined role	The Tenant Guest and SMN Administrator roles need to be assigned in the same project.

Table 1-3 lists the common operations supported by each SMN system policy or role. Select the policies or roles as needed.

Table 1-3 Common operations supported by each system-defined policy or role of SMN

Operation	SMN Administrator
Creating a topic	√
Updating a topic	√
Deleting a topic	√
Querying topics	√
Adding a subscription to a topic	√
Adding tags to a topic	√
Configuring topic policies	√
Publishing a message	√
Adding a subscription	√
Requesting subscription confirmation	√
Canceling a subscription	√
Querying subscriptions	√
Creating a message template	√
Modifying a message template	√
Deleting a message template	√
Querying a message template	√

1.6 SMN and Other Services

SMN can be interconnected with other cloud services to provide them with messaging capabilities so that these services can send notifications to tenants or their message processing systems. For details about how to use SMN in other cloud services, see user guides of the related services.

The following are examples of some services using SMN.

- **Cloud Eye**
After an alarm rule is configured, Cloud Eye can use SMN to send alarm notifications to specific users.
- **Object Storage Service (OBS)**
OBS uses SMN to send OBS bucket events to applications for processing.

1.7 Concepts

Protocol

A protocol is a message type. SMN supports the following protocols: SMS, Email, HTTP, and HTTPS.

Publisher

A publisher sends messages to a topic.

Subscriber

A subscriber receives messages delivered from a topic.

When adding a subscription, you can choose protocols as required:

- Email: The endpoint can be one or more email addresses.
- SMS: The endpoint can be one or more phone numbers.
- HTTP or HTTPS: The endpoint can be one or more URLs.

Topic

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

URN

Uniform Resource Names (URNs) are used to identify SMN resources.

- Topic URN
After a topic is created, SMN generates a topic URN composed of the service name, region name, project ID, and topic name to uniquely identify the topic, for example, **urn:smn:region:ffe4fc4c9a54219b60dbaf7b586e132:Mytopic**.

When you call an API to create a topic, a topic URN will be returned. The topic URN will be used whenever a publisher or subscriber performs operations relating to the topic.

- Subscription URN

After a user subscribes to a topic, SMN will generate a subscription URN composed of the service name, region name, project ID, topic name, and subscription ID, for example,

urn:smn:region:cffe4fc4c9a54219b60dbaf7b586e132:Mytopic:5293b436967f450abc51e0c36347b27a. The URN is displayed on the **Subscriptions** page for subscribers to confirm or cancel a subscription.

Message Template

Message templates contain fixed and changeable content and can be used to send messages quickly. Changeable content is represented with variables. When you publish template messages, the system replaces the variables with the message content you specify.

Template Variable

A message template contains fixed and changeable content. Changeable content is represented with variables. You can specify values for variables when publishing messages using a template.

For example, the template content is **The Arts and Crafts Exposition will be held from *{startdate}* through *{enddate}*. We sincerely invite you to join us..** In the content, *{startdate}* and *{enddate}* are variables.

1.8 Region and AZ

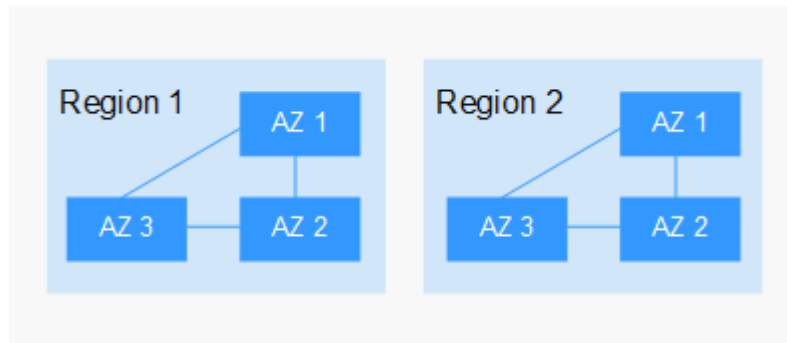
Concept

A region and availability zone (AZ) identify the location of a data center. You can create resources in a specific region and AZ.

- A region is a physical data center, which is completely isolated to improve fault tolerance and stability. The region that is selected during resource creation cannot be changed after the resource is created.
- An AZ is a physical location where resources use independent power supplies and networks. A region contains one or more AZs that are physically isolated but interconnected through internal networks. Because AZs are isolated from each other, any fault that occurs in one AZ will not affect others.

Figure 1-1 shows the relationship between regions and AZs.

Figure 1-1 Regions and AZs



Selecting a Region

Select a region closest to your target users for lower network latency and quick access.

Selecting an AZ

When deploying resources, consider your applications' requirements on disaster recovery (DR) and network latency.

- For high DR capability, deploy resources in different AZs within the same region.
- For lower network latency, deploy resources in the same AZ.

Regions and Endpoints

Before you use an API to call resources, specify its region and endpoint. For more details, see [Regions and Endpoints](#).

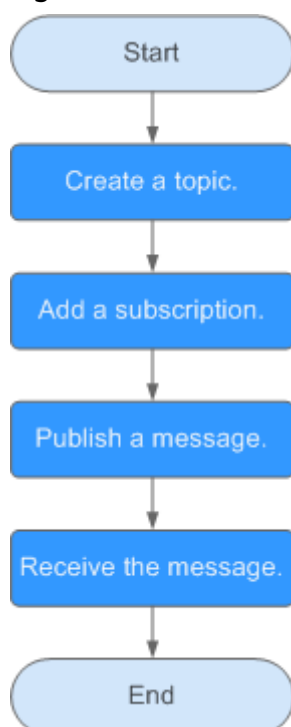
2 Getting Started

2.1 Publishing a Message

After you learn the basic concepts in SMN, you can start to create a topic, add subscriptions to the topic, and publish messages on the SMN console or by calling RESTful APIs provided by SMN.

Figure 2-1 shows the process to publish a message to a topic.

Figure 2-1 Process of publishing a message



Scenarios

To send similar messages repeatedly, create a message template which contains fixed and changeable content. Every time you send messages using the template,

you only have to replace changeable content. For example, your organization holds expositions regularly and needs to notify relevant people of the time, you can create a message template containing date variables and other fixed content.

Step 1. Create 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
4. In the navigation pane, choose **Topics**.
The **Topics** page is displayed.
5. In the upper right corner, click **Create Topic**.
6. Enter a topic name and display name.

Table 2-1 Parameter descriptions

Parameter	Description
Topic Name	<p>Topic name, which:</p> <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	<p>Message sender name which can contain up to 192 characters</p> <p>NOTE After you specify a display name, the sender in email messages will be presented as <i>Display name</i><username@example.com>. Otherwise, the sender will be username@example.com.</p>

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 and subscriptions.

Step 2. 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.

4. In the navigation pane, choose **Subscriptions**.
5. In the upper right corner, click **Add Subscription**.
The **Add Subscription** dialog box is displayed.
6. Specify the required subscription information.
 - a. Beside the **Topic Name** box, click **Select Topic**.
 - b. Specify the subscription protocol and endpoints.

Table 2-2 Parameters for adding a subscription

Parameter	Description
Topic Name	Specifies the name of the topic to which messages are published.
Protocol	Specifies the protocol over which messages are sent. Possible values include SMS , HTTP , HTTPS , and Email .
Endpoint	Specifies the subscription endpoint. You can add up to 10 SMS, email, HTTP, or HTTPS endpoints, one in each line. <ul style="list-style-type: none">● SMS: Enter one or more valid phone numbers. The phone number must be preceded by a plus sign (+) and country code. Examples: +8600000000000 +8600000000001● 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

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 [A.4 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 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.

Step 3. Create a Message Template


1. Log in to the management console.
2. Click  on the upper left to select the desired region and project.
3. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
4. In the navigation pane, choose **Message Templates**.
5. In the upper right corner, click **Create Message Template**.
The **Create Message Template** dialog box is displayed.
6. Specify the template name, protocol, and content.

Table 2-3 Parameters required for creating a message template

Parameter	Description
Template Name	Template name, which: <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	Endpoint protocol of the template, which cannot be changed once the template is created The protocol can be Default , SMS , HTTP , HTTPS , or Email . If you do not specify a protocol, Default is used.

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.

Step 4. Publish 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 **Application**, select **Simple Message Notification**.
The SMN console is displayed.
4. In the navigation pane, choose **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.
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 [A.2 Template Message Format](#) 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 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. Click **OK**.

SMN delivers your message to all subscription endpoints. For details about messages for different protocols, see [A.3 Messages Using Different Protocols](#).

Step 5. Receive the Message

Subscription endpoints of different protocols receive different messages.

- Email

Subscription endpoints are email addresses.
Email messages contain the message subject, content, and a link to unsubscribe.
- HTTP/HTTPS

Subscription endpoints are public network URLs. For details, see section "HTTP/HTTPS Messages" in *Simple Message Notification User Guide*.
- SMS

Subscription endpoints are phone numbers.
SMS messages contain only the message content.

3 Topic Management

3.1 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
4. In the navigation pane, choose **Topics**.
The **Topics** page is displayed.
5. In the upper right corner, click **Create Topic**.
6. Enter a topic name and display name.

Table 3-1 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.

Parameter	Description
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</i> <username@example.com>. Otherwise, the sender will be username@example.com .

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

3.2 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

- Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
- In the navigation pane, choose **Topics**.
The **Topics** page is displayed.
- 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.

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*.

- Click **OK**.

3.3 Configuring Topic Policies

3.3.1 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
2. In the navigation pane, choose **Topics**.
The **Topics** page is displayed.
3. Locate a topic, click **More** under **Operation**, and select **Configure Topic Policy**.
4. 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 3-2](#).

Table 3-2 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.
	Specified user accounts	Only specified users have the permission to publish messages to the topic. Users are specified in the following format: urn:csp:iam::domainId:root. <i>domainId</i> indicates the account IDs of the users. 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. 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.
Services that can publish messages to this topic	Example: OBS The services that can publish messages to a topic vary in different regions.	The selected cloud services have operation permissions of the topic. 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.

3.4 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
2. In the navigation pane, choose **Topics**.
The **Topics** page is displayed.
3. Locate the topic that you want to add a subscription to. In the **Operation** column, click **Add Subscription**.
The **Add Subscription** dialog box is displayed.
4. Specify the subscription protocol and endpoints.

Table 3-3 Parameters for adding a subscription

Parameter	Description
Topic Name	Specifies the name of the topic to which messages are published.
Protocol	Specifies the protocol over which messages are sent. Possible values include SMS , HTTP , HTTPS , and Email .
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. The phone number must be preceded by a plus sign (+) and country code. Examples: +8600000000000 +8600000000001• 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

5. 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 [A.4 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 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.

3.5 Publishing a Message

3.5.1 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 [A.6.2 HTTP or HTTPS Message Format](#).

3.5.2 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
2. In the navigation pane, choose **Topics**.
The **Topics** page is displayed.
3. In the topic list, locate the topic that you need to publish a message to and click **Publish Message** in the **Operation** column.
4. Configure the required parameters based on [Table 3-4](#).
The topic name is provided by default and cannot be changed.

Table 3-4 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
Message	Specifies the message content. The message content cannot be left blank nor exceed 256 KB.

5. Click **OK**.
SMN delivers your message to all subscription endpoints. For details about the messages received by each endpoint, see [A.3 Messages Using Different Protocols](#).

3.5.3 Publishing a JSON Message

Scenarios

In a JSON message, you can specify different message content for different protocols, including SMS, email, 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
2. In the navigation pane, choose **Topics**.
The **Topics** page is displayed.
3. In the topic list, locate the topic that you need to publish a message to and click **Publish Message** in the **Operation** column.
4. Configure parameters by referring to [Table 3-5](#).

Table 3-5 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 5 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 [A.1 JSON Message Format](#).
 - If you choose to automatically generate the JSON message, proceed with steps [5](#) through [8](#).
5. Click **Generate JSON Message**.
 6. 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 [A.1 JSON Message Format](#).
 7. Click **OK**.
 8. 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 [A.1 JSON Message Format](#).

9. Click **OK**.

SMN delivers your message to all subscription endpoints. For details about the messages received by each endpoint, see [A.3 Messages Using Different Protocols](#).

3.5.4 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**, **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 [5 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
4. In the navigation pane, choose **Message Templates**.
5. In the upper right corner, click **Create Message Template**. For details, see [Creating a Message Template](#) in [5 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 **Application**, select **Simple Message Notification**.
The SMN console is displayed.
4. In the navigation pane, choose **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.
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 [A.2 Template Message Format](#) 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 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. Click **OK**.

SMN delivers your message to all subscription endpoints. For details about messages for different protocols, see [A.3 Messages Using Different Protocols](#).

3.6 Deleting a Topic

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

 **NOTE**

Deleting a topic deletes all its subscriptions.

4 Subscription Management

4.1 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
4. In the navigation pane, choose **Subscriptions**.
5. In the upper right corner, click **Add Subscription**.
The **Add Subscription** dialog box is displayed.
6. Specify the required subscription information.
 - a. Beside the **Topic Name** box, click **Select Topic**.
 - b. Specify the subscription protocol and endpoints.

Table 4-1 Parameters for adding a subscription

Parameter	Description
Topic Name	Specifies the name of the topic to which messages are published.
Protocol	Specifies the protocol over which messages are sent. Possible values include SMS , HTTP , HTTPS , and Email .
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. The phone number must be preceded by a plus sign (+) and country code. Examples: +8600000000000 +8600000000001• 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

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 [A.4 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 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.

4.2 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 [A.4 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
4. In the navigation pane, choose **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.
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 [A.4 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.

4.3 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 [A.6.1 Introduction](#).

4.4 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
4. In the navigation pane, choose **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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
4. In the navigation pane on the left, choose **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**.

5 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. Select **Simple Message Notification** under **Application**.
The SMN console is displayed.
4. In the navigation pane, choose **Message Templates**.
5. In the upper right corner, click **Create Message Template**.
The **Create Message Template** dialog box is displayed.
6. Specify the template name, protocol, and content.

Table 5-1 Parameters required for creating a message template

Parameter	Description
Template Name	Template name, which: <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	Endpoint protocol of the template, which cannot be changed once the template is created The protocol can be Default , SMS , HTTP , HTTPS , or Email . If you do not specify a protocol, Default is used.
Content	Template content Use <i>{xxx}</i> as the placeholder to create a template. When you use this template to send messages, replace <i>{xxx}</i> with specific content. <i>xxx</i> must start with a letter or digit and can contain up to 21 characters, including only letters, digits, hyphens (-), periods (.), and underscores (_). The message template must meet the following requirements: <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.

6 Permissions Management

6.1 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 an account or cloud service to perform efficient O&M on your SMN resources.

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

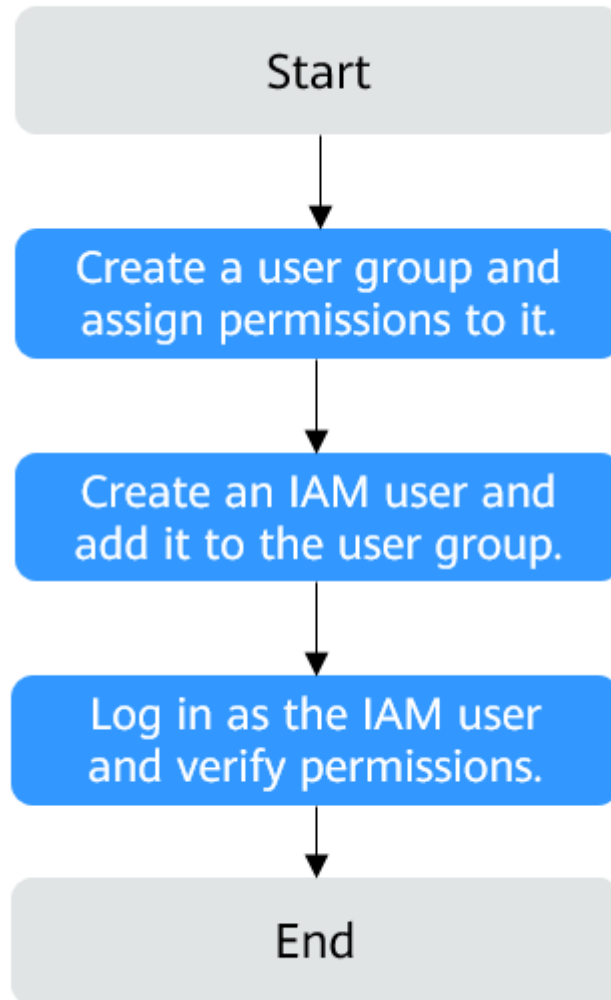
This section describes the procedure for granting permissions (see [Figure 6-1](#)).

Prerequisites

Learn about the system permissions (see [1.5 Permissions](#)) supported by SMN and choose policies or roles according to your requirements. For system permissions of other cloud services, see [Permissions](#).

Process Flow

Figure 6-1 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 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. In the upper right corner of the page, click  .
The **Service Quota** page is displayed.
3. 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?

The system does not support online quota adjustment. If you need to adjust a quota, contact the operations administrator.

Before contacting the operations administrator, make sure that the following information has been obtained:

- Account name, which can be obtained by performing the following operations:
Log in to the management console using the cloud account, click the username in the upper right corner, select **My Credentials** from the drop-down list, and obtain the account name on the **My Credentials** page.
- Quota information, which includes service name, quota type, and required quota

8 FAQs

8.1 What Are the Advantages of SMN?

SMN has the following advantages:

- It does not require many development and maintenance resources, reducing your message notification costs.
- It is highly reliable and scalable.
- It can be quickly deployed and is easy to use.

8.2 What Protocols Does SMN Support?

SMN supports the following protocols:

- Email: Messages are sent to subscribers' email addresses by email.
- SMS: Messages are sent to subscribers' phone numbers by SMS message.
- HTTP or HTTPS: Messages are sent to subscription URLs by HTTP or HTTPS request. SMN only supports public network URLs and public IP addresses.

8.3 What Are the Requirements for an SMN Topic Name?

See [Table 3-1](#).

8.4 How Many Topics Can I Create?

By default, you can create 3,000 topics.

8.5 How Many Subscriptions Can Be Added to a Topic?

By default, 10,000 subscriptions can be added to a topic.

8.6 How Many Messages Can Be Published to a Topic?

There is no limit on the number of messages that can be published to a topic.

8.7 How Many Message Templates Can I Create?

By default, you can create 100 message templates.

8.8 Can I Add Subscriptions Using Multiple Protocols to a Topic?

Yes. A topic supports HTTP, HTTPS, email, and SMS subscriptions.

8.9 Can a Topic Creator Change Subscription Protocols in a Topic?

No. A topic creator cannot change subscription protocols after subscriptions are added to the topic.

8.10 Can I Change a Subscription Endpoint for a Topic?

No. If necessary, you can delete the endpoint and add a new one.

8.11 Can I Delete a Published Message?

No. You cannot delete a message once it is published.

8.12 Does SMN Ensure That Messages Are Received by Subscription Endpoints?

SMN pushes messages to subscription endpoints asynchronously, which does not ensure the timeliness of message delivery. If your service requires quasi-real-time message delivery, exercise caution whether to use SMN.

If a subscription endpoint is accessible, it will receive all messages delivered by SMN.

If an endpoint is inaccessible, SMN saves the undelivered message in a message queue and tries to deliver it six more times. If the message still fails to be delivered, SMN discards it and does not send the information to the publisher that the message delivery failed.

The interval for re-sending an undelivered message varies depending on the length of the message queue. Usually, an undelivered message is processed within

several hours. If a queue has too many undelivered messages, those messages will be processed within a day.

8.13 Will a Subscriber Receive the Same Message Multiple Times?

A subscriber will only receive a message once. SMN re-sends a message only when there is a network or device failure.

8.14 Why Do Subscribers Fail to Receive Messages After I Publish Messages to a Topic?

If you have verified that the subscription endpoints are normal, the most possible reason is that the subscribers have not confirmed their subscriptions or the confirmation messages and messages published have been blocked.

Check whether the message is blocked and processed as a junk message.

If the problem persists, contact technical support.

SMS Message

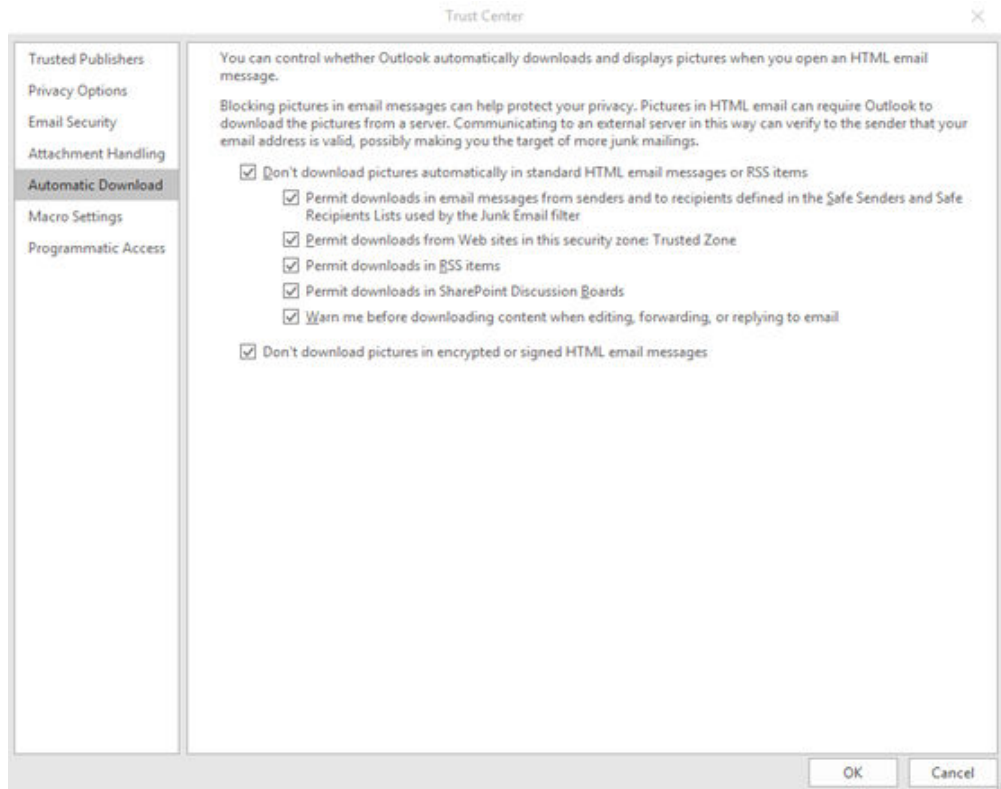
- Confirmation messages
Check whether the messages have been blocked.
- Notification messages
Check whether the subscription has been confirmed or whether the message has been blocked.

8.15 What Can I Do When Pictures in an Email Message Cannot Be Displayed?

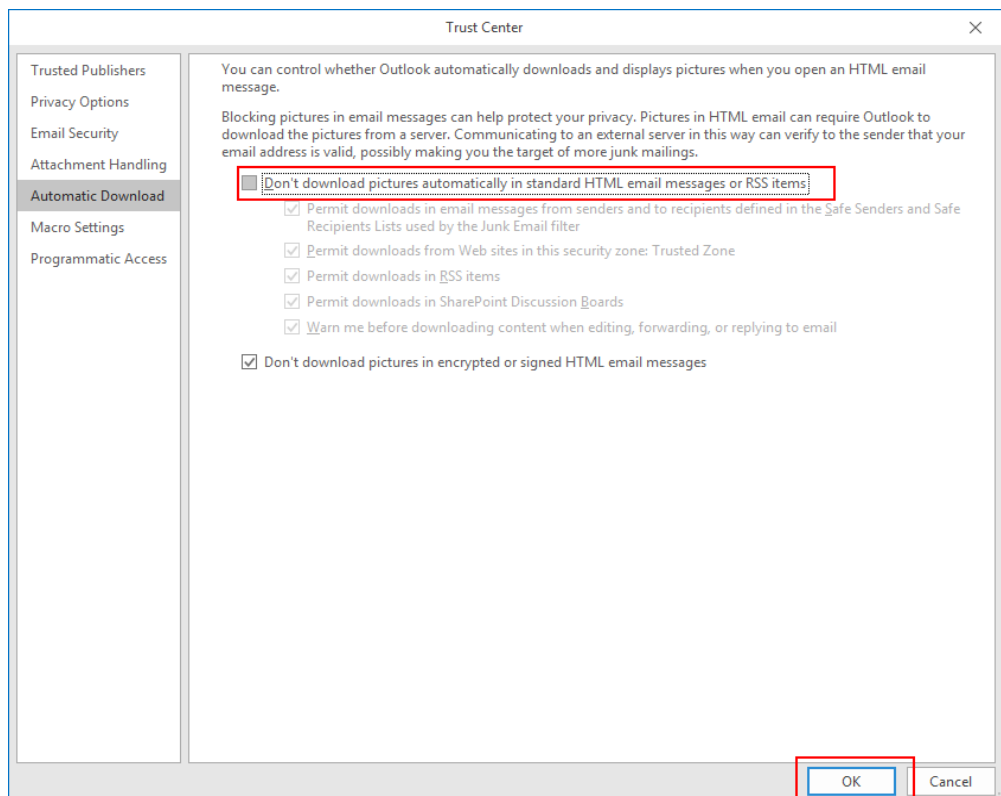
To protect your privacy, some mailboxes do not automatically download pictures from the Internet by default. If pictures in an email cannot be displayed, you need to enable automatic picture download.

The following uses Outlook 2019 as an example to describe how to enable automatic picture download. Procedures are as follows:

1. Open Outlook and click **File** in the upper left corner.
2. Click **Options**.
The **Options** page is displayed.
3. Click **Trust Center** and **Settings**.
The **Trust Center** page is displayed.



4. On the **Automatic Download** tab, deselect **Don't download pictures automatically in standard HTML email messages or RSS items**, and click **OK**.



 NOTE

For other types of mailboxes, search for "**How Do I Enable Automatic Picture Download**" through your browser.

8.16 How Do I Obtain My Account ID?

1. Log in to the management console.
2. Point to the username in the upper right corner and click **Basic Information**.
3. On the **Account Info** page, click **Manage** beside **Security Credentials**.
4. Obtain the account ID.

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 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.
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.

Action	API	Function
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.6 HTTP/HTTPS Messages

A.6.1 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/HTTP message is composed of a message header and body, which are illustrated in detail in this topic.

A.6.2 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 A.6.3 Message Signature Verification .
topic_urn	Indicates the URN of the topic the message belongs to.
message_id	Indicates the unique message ID.

Parameter	Description
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+Rs88yuyMHq1k4h3jUkcDZ6HCqTqdpJ8nrLcdqETyyEiOQyTszJdU05z
+LhfE8jerCCdSbL4zeInVkydHh0kcCRWmORye0/EuQ/gLC1UIXwvUsqbUCPnBRhNFXOeXMOppCzK
+d04xjy4QHd1H/bHxgsY3AlTe0gCFT068Zru7OK6w9aQaY44mXnN3OWGmBmoHyFab5TRXLSQNz/9u/
Vj646cQMMaI0PPQ30QzGYD0MtZgDZi12m8jMTHAnMkTEcbLaEgaqmaoEnATSpEcpFKNXv2skwk7rsVakMOI
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://console.xxx.com/smn/
SMN_region01_b3974c411807498bb532b3cd6cd65d91.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 A.6.3 Message Signature Verification .

Parameter	Description
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/
Vj646cQMMaI0PPQ30QzGYD0MtzgDZi12m8jMTHAnMkTEcbLaEgaqmaoEnATSpEcspFKNXv2skwk7rsVakMOI
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://console.xxx.com/smn/
SMN_region01_b3974c411807498bb532b3cd6cd65d91.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 .

Parameter	Description
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 A.6.3 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/gLC1UIXwvUsqbUCPnBRhNFXOeXMOPPCzK
+d04xjy4QHd1H/bHxgsY3AlTe0gCFT068Zru7OK6w9aQaY44mXnN3OWGmBmoHyFab5TRXLSQNZ/9u/
Vj646cQMMal0PPQ30QzGYD0MtgDZi12m8jMTHAnMkTEcbLaEgaqmaoEnATSpEcspFKNXv2skwk7rsVakMOI
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://console.xxx.com/smn/
SMN_region01_b3974c411807498bb532b3cd6cd65d91.pem",
  "timestamp": "2019-08-12T22:40:56Z"
}
```

A.6.3 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.6.4 Sample Code

Java

Verify **signing_cert_url**, **signature** that obtained in [A.6.2 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 jsrsasig = 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
2024-04-15	This issue is the first official release.