PAX Cloud Message

API Java SDK – V7.4

Sep 12, 2020



../PaxStore-H5/img/logo_footer.png

Contents

[Introduction: 3](#_Toc19117310)

[Precondition: 3](#_Toc19117311)

[Get Started 3](#_Toc19117312)

[Integrate with SDK 3](#_Toc19117313)

[API Usage 4](#_Toc19117314)

[Send Message to Terminals 5](#_Toc19117315)

[API 5](#_Toc19117316)

[Sample Code 6](#_Toc19117317)

[Query the message arrived number 8](#_Toc19117318)

[API 8](#_Toc19117319)

[Sample Code 9](#_Toc19117320)

[License 9](#_Toc19117321)

## Introduction:

PAX Cloud Message is a service provided by PAXSTORE, it allows ISV to send messages to app of specified terminals.

PAXSTORE exposes RESTful API to let ISV to invoke. And this SDK simplifies the integration.

Before using the service, PAXSTORE need to enable PAX Cloud Message service. PAXSTORE exposes reseller, merchant and terminal related APIs for third-party system convenience. So that the authorized third-party system can do operations for reseller, merchant and terminal without logging into PAXSTORE's admin console.

The exposed API is RESTful style. PAXSTORE provides the Java SDK to simplify the remote API invoke.

## Precondition:

There are some preconditions before using PAX Cloud Message service.

* PAX Cloud Message service is enabled in PAXSTORE
* App supports PAX Cloud Message (Integrate with latest paxstore-3rd-app-android-sdk and claim this app support PAX Cloud Message service when upload in PAXSTORE developer center)
* PAX Cloud Message service is enabled by marketplace administrator at app level in app detail page when send message to non-sandbox terminals
* The PAXSTORE client version installed in the terminal must great than 6.3.3

## Get Started

## Integrate with SDK

Update maven's settings.xml add a new repository <https://dl.bintray.com/paxstore-support/paxstore/>

Update pom.xml add SDK dependency for maven project.

|  |
| --- |
| *<dependency>*  *<groupId>com.pax.market</groupId>*  *<artifactId>paxstore-cloudmsg-java-sdk</artifactId>*  *<version>7.4</version>*  *</dependency>* |

## API Usage

Initialize the API using the constructor like this.

|  |
| --- |
| *CloudMessageServiceApi api = new CloudMessageServiceApi("https://api.whatspos.com/p-market-api", "appkey", "appsecret");* |

Please use the correct app key and secret.

All the APIs in the class *CloudMessageServiceApi* returns a unified result *com.pax.market.trdsys.sdk.message.result.Result*. Below is the structure of class *com.pax.market.api.sdk.java.api.base.dto.Result*

|  |  |  |
| --- | --- | --- |
| Property | Type | Description |
| businessCode | int | The business code, it represents the API invoke result. 0 means invoke the API success, if it is -1 means SDK side validation failed. For other business codes please refer to the message property |
| message | String | The description of businessCode |
| validationErrors | List | Client side validation errors. |
| data | T(generic) | The actual DTO, the structure will be described in each APIs |
| rateLimit | String(int format) | The maximum number of requests you're permitted to make per 10 minutes. |
| rateLimitRemain | String(int format) | The number of requests remaining in the current rate limit window. |
| rateLimitReset | String(long format) | The time at which the current rate limit window resets in UTC epoch millisecond. |

Note: There’s API limitation of Cloud Message API From PAXSTORE version 7.4.

The max number of authenticate failed times is 20 per 30 minutes. And the send message API has 10 times limit per 10 minutes.

Below figure listed the global business codes, those business codes may appear in every result of API call. This document won't list those business codes in the following API chapters when introducing the APIs.

|  |  |  |
| --- | --- | --- |
| Business Code | Message | Description |
| 0 |  | Successful API call. |
| -5 | JSON error! |  |
| -6 | Connection timeout! | Encounter SocketTimeoutException. |
| -7 | Cannot connect to remote server! | The remote server is down or the constructor argument baseUrl is not correct. |
| -8 | Request error! | Please check the error log or send the error log to support. |
| -13 | BaseUrl not correct! | The API BaseUrl may not correct |
| 129 | Authentication failed |  |
| 104 | Client key is missing or invalid |  |
| 108 | Marketplace is not available |  |
| 109 | Marketplace is not active |  |
| 105 | Client key is blocked |  |
| 103 | Access token is invalid |  |
| 102 | Access token is missing |  |
| 101 | Invalid request method | The request method is not correct |
| 113 | Request parameter is missing or invalid |  |
| 429 | Too many request | Request number exceed the maximum number in the current rate limit window |
| 997 | Malformed or illegal request | The JSON in request body is not a valid JSON |
| 998 | Bad request |  |
| 999 | Unknown error | Unknow error, please contact with support. |

## Send Message to Terminals

### API

|  |
| --- |
| *public Result<PushMessageCreateResultDto> createPushMessage(MessageCreateRequest createRequest)* |

The structure of the class *PushMessageCreateResultDto*

|  |  |  |
| --- | --- | --- |
| Property Name | Type | Description |
| msgIdentifier | String | The identifier of the message |

Currently Cloud Message service support 3 types of message.

* Notification Type  
  For notification message type the message body contains a title and content like below

|  |
| --- |
| {  "title":"This is message title",  "content":"This is message content" } |

* Data Type  
  Data type message body is a JSON format string
* Mix Type Mix type message is the combination of notification type message and data type message.

After send message API will return the result, if the property businessCode of result is 0 mean the message is created successfully.

### Sample Code

* Send notification type message to terminal sn00001,sn00002 and print the msgIdentifier if success

|  |
| --- |
| *MessageCreateRequest request = new MessageCreateRequest(); request.setMsgType(MsgType.Notification); MsgContent msgContent = new MsgContent(); Notification notification = new Notification(); notification.setTitle("This is title of notification"); notification.setContent("This is content of notification"); msgContent.setNotification(notification); request.setContent(msgContent); request.setSerialNos(new String[]{"sn00001", "sn00002"}); Result<PushMessageCreateResultDto> result = getMessageApi().createPushMessage(request); if(result.getData()!=null) {  System.out.println(result.getData().getMsgIdentifier()); }* |

* Send data type message to terminal sn00001,sn00002 and print the msgIdentify if success

|  |
| --- |
| *MessageCreateRequest request = new MessageCreateRequest(); request.setMsgType(MsgType.Data); MsgContent msgContent = new MsgContent(); JSONObject data = new JSONObject(); data.put("date", "12/7/2019"); data.put("titile", "title"); JSONObject addressNode = new JSONObject(); addressNode.put("city", "jiangsu suzh"); addressNode.put("postCode", "215000"); data.put("address", addressNode); msgContent.setData(data); request.setContent(msgContent); request.setSerialNos(new String[]{"sn00001", "sn00002"}); Result<PushMessageCreateResultDto> result = getMessageApi().createPushMessage(request); if(result.getData()!=null) {  System.out.println(result.getData().getMsgIdentifier()); }* |

* Send mix type message to terminal sn00001,sn00002 and print the msgIdentify if success

|  |
| --- |
| *MessageCreateRequest request = new MessageCreateRequest(); request.setMsgType(MsgType.Mix); MsgContent msgContent = new MsgContent(); Notification notification = new Notification(); notification.setTitle("This is title of notification"); notification.setContent("This is content of notification"); msgContent.setNotification(notification); JSONObject data = new JSONObject(); data.put("date", "12/7/2019"); data.put("titile", "title"); JSONObject addressNode = new JSONObject(); addressNode.put("city", "jiangsu suzh"); addressNode.put("postCode", "215000"); data.put("address", addressNode); msgContent.setData(data); request.setContent(msgContent); request.setSerialNos(new String[]{"sn00001", "sn00002"}); Result<PushMessageCreateResultDto> result = getMessageApi().createPushMessage(request); if(result.getData()!=null) {  System.out.println(result.getData().getMsgIdentifier()); }* |

**Possible client validation errors**

> Parameter createRequest is mandatory!  
> Parameter serialNos's max size is 1000!

**Possible business codes**

|  |  |  |
| --- | --- | --- |
| Business Code | Message | Description |
| 2609 | Serial No is mandatory |  |
| 2610 | Max terminal Id size is 1000 |  |
| 2621 | Duplicated serial numbers |  |
| 2613 | Message type in invalid |  |
| 2611 | Message content is mandatory |  |
| 2617 | Message data is mandatory | If the message type is data or mix type, the data property of create reqeust is mandatory |
| 2618 | Message data is invalid | If the message type is data or mix type, make sure the data is a json object |
| 2612 | Message content is too long |  |
| 2619 | No valid serial numbers | Make sure the terminal has installed the app and the push channel has switched to cloud push channel |

## Query the message arrived number

### API

|  |
| --- |
| *public Result<QueryArriveRateDto> queryArriveRate(String messageIdentifier)* |

This API is used to query the message arrived terminal number. The parameter messageIdentifier is returned after created message success. The effective time of the message is 24 hours, and query the message arrive terminal number when message is still effective is not allowed. The structure of QueryArriveRateDto like below

|  |  |  |
| --- | --- | --- |
| Property Name | Type | Description |
| arrivedNumber | Integer | The number of terminal of the message arrived |
| arrivedRate | String | The percentage of the arrived terminal number |

### Sample Code

|  |
| --- |
| *Result<QueryArriveRateDto> result = getMessageApi().queryArriveRate("b8ddeca556dd4f25b1a1143a3f5e8855"); if(result.getBusinessCode() == 0) {  System.out.println(result.getData().getArrivedNumber());  System.out.println(result.getData().getArrivedRate()); }* |

**Possible client validation errors**

> Parameter messageIdentifier is mandatory!  
> Parameter messageIdentifier's length must be 32!

**Possible business codes**

|  |  |  |
| --- | --- | --- |
| Business Code | Message | Description |
| 2615 | Invalid message identifier |  |
| 2620 | Message sent failed |  |
| 2616 | Arrived rate is not available now, please try later |  |

## License

See the [Apache 2.0 license](file:///C:\Users\Administrator\Downloads\LICENSE) file for details.

Copyright 2018 PAX Computer Technology(Shenzhen) CO., LTD ("PAX")  
  
Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at following link.  
  
 http://www.apache.org/licenses/LICENSE-2.0  
  
Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an "AS IS" BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the License for the specific language governing permissions and  
limitations under the License.