Messaging API Use Case Guide

Version: 1.0

Date: 7/15/2020

Contents

What is the Messaging API?

With the Selling Partner API for Messaging (the Messaging API), you can build applications that let sellers send messages to buyers after they place an order. Sellers can send a variety of message types. These include asking order related questions, arranging a delivery, requesting the removal of negative feedback, and more. The Messaging API can send messages to a buyer even if the buyer has not contacted the seller first. You cannot use this API to respond to messages from buyers.

See the Messaging API reference for a list of operations that correspond to the available message types.

Message availability

Not all message types are available in all circumstances. There are a number of factors that can affect message availability, including whether the order has been fulfilled, the type of seller, the number of messages already sent for the order, etc.

Workflow for sending a message

Sending a message is a multi-step process that includes calls to operations in both the Messaging API and the Uploads API. Here is the basic workflow for sending a message:

1. Your application displays the seller's recent orders, and the seller chooses an order for which they want to send a message to the buyer.
2. Your application displays the message types that are available for the order that the seller chooses. Get the available message types by calling the **getMessagingActionsForOrder** operation of the Messaging API.
3. The seller chooses a message type for the order.
4. If custom messages are supported for the message type, the seller inputs a custom message.
5. If attachments are supported for the message type, the seller uploads an attachment to your application. The seller also inputs a file name for the attachment.
6. If the seller uploads an attachment to your application, your application uploads the attachment to an Amazon S3 bucket by doing the following:
    a. Calculating a Content-MD5 hash for the attachment, which is needed for creating an upload destination.
    b. Creating an upload destination by calling the **createUploadDestination** operation of the Uploads API.
    c. Uploading the attachment to the destination.
7. Your application requests that a message be sent from the seller to the buyer by calling the operation that corresponds to the message type that the seller chose in step #3. For example, your application calls the **createConfirmOrderDetails** operation if the seller chose the "Confirm order details" message type.
8. Amazon emails a message from the seller to the buyer.

See Tutorial: Send a message for the detailed workflow.

Terminology

- **Amazon S3 presigned URL**. A URL for an Amazon S3 bucket to which you can upload an object without AWS security credentials or permissions. You get an Amazon S3 presigned URL in Step 7. Create an upload destination.

Tutorial: Send a message

This tutorial shows you how to create an application that lets a seller send a message to a buyer.

Prerequisites

To complete this tutorial, you will need:

- Authorization from the seller for whom you are making calls. See Selling Partner API Developer Guide for more information.

Steps

Step 1. Get the order for which the seller wants to send a message

1. Display a list of the seller's recent orders that the seller can choose from.
2. The seller chooses an order for which they want to send a message to the buyer.
3. Associate an Amazon order identifier with the order that the seller chooses.
4. Save the Amazon order identifier for Step 2. Get a list of available message types.

Step 2. Get a list of available message types

Call the **getMessagingActionsForOrder** operation of the Messaging API, specifying the Amazon order identifier from Step 1. Get the order for which the seller wants to send a message, to get a list of message types that are available for the order.

1. Call the **getMessagingActionsForOrder** operation of the Messaging API, passing the following parameters:

Path Parameter:

Parameter	Description	Required
amazonOrderId	An Amazon order identifier. This specifies the order for which a message is to be sent. Get this value from Step 1. Get the order for which the seller wants to send a message. Type: string	Yes

Query Parameter:

Parameter	Description	Required
marketplaceIds	A marketplace identifier. This specifies the marketplace in which the order was placed. You can specify only one marketplace. Type: array[string]	Yes

Request example:

```
GET https://sellingpartnerapi-na.amazon.com/messaging/v1/orders/902-
6786204-8179805?marketplaceIds=ATVPDKIKX0DER
```

Response

Response example:


```
{
  "_links":
  {
    "actions": [
      {
        "href": "/messaging/v1/orders/902-6786204-
8179805/messages/confirmCustomizationDetails?marketplaceIds=ATVPDKIKX0
DER",
        "name": "confirmCustomizationDetails"
      },
      {
        "href": "/messaging/v1/orders/902-6786204-
8179805/messages/negativeFeedbackRemoval?marketplaceIds=ATVPDKIKX0DER"
,
        "name": "negativeFeedbackRemoval"
      },
      {
        "href": "/messaging/v1/orders/902-6786204-
8179805/messages/confirmOrderDetails?marketplaceIds=ATVPDKIKX0DER",
        "name": "confirmOrderDetails"
      },
    ],
    "self":
    {
      "href": "/messaging/v1/orders/902-6786204-
8179805?marketplaceIds=ATVPDKIKX0DER"
    }
  },
  "_embedded":
  {
    "actions": [
      {
        "_links":
```

```
        {
          "schema":
          {
            "href": "/messaging/v1/orders/902-6786204-
8179805/messages/confirmCustomizationDetails/schema",
            "name": "confirmCustomizationDetails"
          },
          "self":
          {
            "href": "/messaging/v1/orders/902-6786204-
8179805/messages/confirmCustomizationDetails?marketplaceIds=ATVPDKIKX0
DER",
            "name": "confirmCustomizationDetails"
          }
        },
        "_embedded":
        {
          "schema":
          {
            "_links":
            {
              "self":
              {
                "href": "/messaging/v1/orders/902-6786204-
8179805/messages/confirmCustomizationDetails/schema",
                "name": "confirmCustomizationDetails"
              }
            },
            "type": "object",
            "name": "confirmCustomizationDetails",
            "title": "Confirm customization details",
            "description": "Ask your customer to provide details or
verify the customization input provided (name spelling, imagery,
initials).",
            "properties":
            {
              "attachments":
              {
                "type": "array",
                "items":
                {
                  "type": "object",
                  "required": [],
                  "properties":
                  {
                    "fileName":
                    {
                      "type": "string"
                    },
                    "id":
                    {
                      "type": "string"
```

```
                    }
                  }
                },
                "title": "Add attachment",
                "description": "You can upload text files, PDFs, Word
documents, and these image file types: .jpg, .gif, and .png. The total
size of attachments must be less than 10 MB.",
                "x-ui-field-type": "attachments",
                "maxItems": 5
              },
              "rawMessageBody":
              {
                "type": "string",
                "title": "Explain why you are contacting and any
action you need your customer to take. ",
                "description": "800 character limit. Only links
related to order completion are allowed, no HTML or email addresses.",
                "x-ui-field-type": "text",
                "minLength": 1,
                "maxLength": 800
              }
            },
            "required": [
              "attachments",
              "rawMessageBody"
            ],
            "$schema": "http://json-schema.org/draft-04/schema#",
            "x-ui-hidden": null
          }
        },
        "name": "confirmCustomizationDetails",
        "title": "Confirm customization details"
      },
      {
        "_links":
        {
          "schema":
          {
            "href": "/messaging/v1/orders/902-6786204-
8179805/messages/negativeFeedbackRemoval/schema",
            "name": "negativeFeedbackRemoval"
          },
          "self":
          {
            "href": "/messaging/v1/orders/902-6786204-
8179805/messages/negativeFeedbackRemoval?marketplaceIds=ATVPDKIKX0DER"
,
            "name": "negativeFeedbackRemoval"
          }
        },
        "_embedded":
        {
```

```
          "schema":
          {
            "_links":
            {
              "self":
              {
                "href": "/messaging/v1/orders/902-6786204-
8179805/messages/negativeFeedbackRemoval/schema",
                "name": "negativeFeedbackRemoval"
              }
            },
            "type": "object",
            "name": "negativeFeedbackRemoval",
            "title": "Request to update negative feedback",
            "description": "Ask your customer to consider updating
their seller feedback rating. You may only send this once per order.",
            "properties": {},
            "required": [],
            "$schema": "http://json-schema.org/draft-04/schema#",
            "x-ui-hidden": true
          }
        },
        "name": "negativeFeedbackRemoval",
        "title": "Request to update negative feedback"
      },
      {
        "_links":
        {
          "schema":
          {
            "href": "/messaging/v1/orders/902-6786204-
8179805/messages/confirmOrderDetails/schema",
            "name": "confirmOrderDetails"
          },
          "self":
          {
            "href": "/messaging/v1/orders/902-6786204-
8179805/messages/confirmOrderDetails?marketplaceIds=ATVPDKIKX0DER",
            "name": "confirmOrderDetails"
          }
        },
        "_embedded":
        {
          "schema":
          {
            "_links":
            {
              "self":
              {
                "href": "/messaging/v1/orders/902-6786204-
8179805/messages/confirmOrderDetails/schema",
                "name": "confirmOrderDetails"
```

```
              }
            },
            "type": "object",
            "name": "confirmOrderDetails",
            "title": "Confirm order details",
            "description": "Ask your customer a specific order-related
question prior to shipping their order.",
            "properties":
            {
              "rawMessageBody":
              {
                "type": "string",
                "title": "Explain why you are contacting and any
action you need your customer to take. ",
                "description": "2000 character limit. Only links
related to order completion are allowed, no HTML or email addresses.",
                "x-ui-field-type": "text",
                "minLength": 1,
                "maxLength": 2000
              }
            },
            "required": [
              "rawMessageBody"
            ],
            "$schema": "http://json-schema.org/draft-04/schema#",
            "x-ui-hidden": null
          }
        },
        "name": "confirmOrderDetails",
        "title": "Confirm order details"
      },
    ]
  }
}
```


1. Save the list of schemas in the `_embedded.actions` array of the response. In the `_embedded.schema` section of each array element you can find a schema for each available message type. Use the `title` value in each schema to create a list of available message types for the seller to choose from in Step 3. Get the seller's choice of message type. Use the schemas to understand how to display and enforce the restrictions for custom messages and attachments that apply for each message type. For more information, see Step 4. Get custom message text from the seller (if supported) and Step 5. Get an attachment from the seller (if supported).
2. Save the list of paths in the `_links.actions` array of the response. The `href` value in each array element contains a path for each available message type. You can use one of these paths in a call the appropriate operation in Step 9. Send a message to the buyer.

Step 3. Get the seller's choice of message type

1. Display a list of available message types for the seller to choose from. Get these from the `title` values that you saved in Step 2. Get a list of available message types.

2. Let the seller choose the message type that they want send, and save it. This choice will affect several subsequent steps, including:
    - Step 4. Get custom message text from the seller (if supported). The choice of message type determines whether a custom message is supported.
    - Step 5. Get an attachment from the seller (if supported). The choice of message type determines whether attachments are supported.
    - Step 9. Send a message to the buyer. The choice of message type determines which operation you will call in Step 9. For example if the seller chooses the "Confirm customization details" message type, you will call the **confirmCustomizationDetails** operation in Step 9.

Step 4. Get custom message text from the seller (if supported)

Some message types permit the seller to include a custom message in the email to the buyer. To find out if a message type supports custom messages, review the Messaging API reference to find the operation that corresponds to the message type. For example, the **createConfirmServiceDetails** operation corresponds to the "Contact Home Service customer" message type. If the operation includes a `text` body parameter, the message type supports custom messages.

1. Determine if the message type that the seller chose in Step 3. Get the seller's choice of message type supports custom messages. If custom messages are not supported, go directly to Step 5. Get an attachment from the seller (if supported). If custom messages are supported, continue with this procedure.
2. Let the seller input the raw text that they want for their custom message.
3. Display and enforce any restrictions on the message text that the seller inputs. You can find the text restrictions in the `_embedded.schema.properties.rawMessageBody` section of the `_embedded.actions` array element that corresponds to the seller's message type. Find this in the response for the **getMessagingActionsForOrder** operation in Step 2. Get a list of available message types.
4. Save the message text and use it as input for Step 9. Send a message to the buyer.

Step 5. Get an attachment from the seller (if supported)

Some message types permit the seller to include a link to an attachment in the email to the buyer. To find out if a message type supports attachments, review the Messaging API reference to find the operation that corresponds to the message type. For example, the **createWarranty** operation corresponds to the "Send Warranty Information" message type. If the operation includes an `attachments` body parameter, the message type supports attachments.

1. Determine if the message type that the seller chose in Step 3. Get the seller's choice of message type supports attachments. If attachments are not supported, go directly to Step 9. Send a message to the buyer. If attachments are supported, continue with this procedure.
2. Let the seller upload an attachment to your application.
3. Display and enforce any restrictions on the attachment that the seller uploads. You can find the attachment restrictions in the `_embedded.schema.properties.attachments` section of the `_embedded.actions` array element that corresponds to the seller's message type. Find this in the response for the **getMessagingActionsForOrder** operation in Step 2. Get a list of available message types.
4. Convert the attachment into an input stream and save it for Step 8. Upload the attachment.
5. Let the seller input a file name for the attachment, including the file extension.

6. Save the file name and use it as input for Step 9. Send a message to the buyer.

Step 6. Calculate a Content-MD5 hash

Calculate a Content-MD5 hash for the attachment from Step 5. Get an attachment from the seller (if supported). You will need it to create an upload destination in Step 7. Create an upload destination.

The Java sample code in this step contains logic for calculating a Content-MD5 hash. This sample code uses the Apache HTTP client. You can also find this sample code on the Selling Partner API GitHub site here: https://github.com/amzn/amazon-marketplace-api-sdk/blob/master/sample%20codes/SellingPartnerApiMessagingUploadsSolicitationsSampleCode/util/CreateMD5.java.

1. Use the following as input for the sample code:
    - The attachment input stream from Step 5. Get an attachment from the seller (if supported), is the argument for the `fis` parameter of the `computeContentMD5Value` method of the `CreateMD5` class.
2. Save the `md5Content` value to pass in with the **Content-MD5** parameter in Step 7. Create an upload destination.

Sample Java code

```
package io.swagger.client.util;

import java.io.FileInputStream;
import java.io.IOException;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class CreateMD5
{

  /** Calculate content MD5 hash values for feeds stored on disk.
   */
  public static String computeContentMD5Value(FileInputStream fis)
  throws IOException,
  NoSuchAlgorithmException
  {

    DigestInputStream dis = new DigestInputStream(fis,
        MessageDigest.getInstance("MD5"));

    byte[]buffer = new byte[8192];
    while (dis.read(buffer) > 0);

    String md5Content = new
String(org.apache.commons.codec.binary.Base64
        .encodeBase64(dis.getMessageDigest().digest()));

    // Effectively resets the stream to be beginning of the file
    // via a FileChannel.
```

```
    fis.getChannel().position(0);

    return md5Content;
  }
}
```

Step 7. Create an upload destination

Call the **createUploadDestination** operation of the Uploads API to create an upload destination for your attachment, in the form of an Amazon S3 presigned URL. For definitions, see Terminology.

1. Call the **createUploadDestination** operation of the Uploads API, passing the following parameters:

Query parameters:

Parameter	Description	Required
marketplaceIds	A marketplace identifier. This specifies the marketplace where the upload will be available. You can specify only one marketplace. Type: array[string]	Yes
Content-MD5	The Content-MD5 hash that you calculated in Step 6. Calculate a Content-MD5 hash. Type: string	Yes

Request example:

```
GET https://sellingpartnerapi-
na.amazon.com/uploads/v1/uploadDestinations?marketplaceIds=ATVPDKIKX0D
ER&Content-MD5=1jnGuPv7xwPiat6NesKL%2Fw%3D%3D
```

Response

Response example:

```
{
  "payload":
  {
    "uploadDestinationId": "4370d51b-2954-445c-9f30-620eed36b9de",
    "url": "https://s3.amazonaws.com/buyer-seller-messaging-test-
draft-attachment-namarketplace/%2F-21/4370d51b-2954-445c-9f30-
620eed36b9de?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20200623T125000Z&X-Amz-SignedHeaders=content-md5%3Bhost%3Bx-amz-
server-side-encryption&X-Amz-Expires=900&X-Amz-
Credential=AKIAW5VUA47EMB4ZFAGS%2F20200623%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-
Signature=5351ed97518f3ce7a7734a60f54c7fa0a31995e8797dc6fdd498856be0f0
5671",
    "headers":
```

```
    {
      "x-amz-server-side-encryption": "aws:kms",
      "Content-MD5": "1jnGuPv7xwPiat6NesKL/w=="
    }
  }
}
```

2. Save the following values:
    - **url** and **headers**. Use these values in Step 8. Upload the attachment.
    - **uploadDestinationId**. Use this value in Step 9. Send a message to the buyer.

Step 8. Upload the attachment

Upload to an Amazon S3 bucket the attachment that the seller uploaded to your application in Step 5. Get an attachment from the seller (if supported).

The Java sample code in this step contains logic for uploading an attachment. This sample code uses the Apache HTTP client. You can also find this sample code on the Selling Partner API GitHub site here: https://github.com/amzn/amazon-marketplace-api-sdk/blob/master/sample%20codes/SellingPartnerApiMessagingUploadsSolicitationsSampleCode/util/UploadFileToDestination.java.

- Use the following as input to the sample code:
    - Use the **url** value that you saved in Step 7. Create an upload destination as the argument for `URL(uploadDestinationPayload.getPayload().getUrl())` in the `UploadFileToDestination` class.
    - Use the **headers** value that you saved in Step 7. Create an upload destination as input for `(Map < String, String > )uploadDestinationPayload.getPayload().getHeaders()` in the `UploadFileToDestination` class.
    - Use the attachment input stream that you saved in Step 4. Get custom message text from the seller (if supported) and use it as input for the `out.write()` method of the `UploadFileToDestination` class.

Sample Java code

```
package io.swagger.client.util;

import java.io.IOException;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Map;

import io.swagger.client.model.CreateUploadDestinationResponse;

public class UploadFileToDestination
{

  private static String requestMethod = "PUT";
  private static String contentType = "Content-Type";
```

```
  private static String contentValue = "application/text";
  private static URL url;
  private static Map < String,
  String > headers;

  /* Upload file to destination using the response returned in
  CreateUploadDestination. Reference
https://docs.aws.amazon.com/AmazonS3/latest/dev/PresignedUrlUploadObje
ctJavaSDK.html)
   */

   @ SuppressWarnings("unchecked")
  public static void uploadFileInDestination(
    CreateUploadDestinationResponse uploadDestinationPayload)
  throws IOException
  {

    url = new URL(uploadDestinationPayload.getPayload().getUrl());
    headers = (Map < String, String >
)uploadDestinationPayload.getPayload()
    .getHeaders();

    // Create the connection and use it to upload the new object using
the
    // pre-signed URL.
    HttpURLConnection connection =
(HttpURLConnection)url.openConnection();
    connection.setDoOutput(true);
    connection.setRequestMethod(requestMethod);
    connection.setRequestProperty(contentType, contentValue);
    // headers that are returned in createDestination call
    for (Map.Entry < String, String > headersEntry:
headers.entrySet())
    {
      connection.setRequestProperty(headersEntry.getKey(),
        headersEntry.getValue());
    }
    OutputStreamWriter out = new OutputStreamWriter(
        connection.getOutputStream());
    out.write("Text of the file");
    out.close();
    // Check the HTTP response code. To complete the upload and make
the
    // object available,
    // you must interact with the connection object in some way.
    connection.getResponseCode();
    System.out.println("HTTP response code: " +
connection.getResponseCode()
       + connection.getResponseMessage());
    connection.disconnect();
  }
}
```

The attachment is uploaded to the Amazon S3 bucket.

Step 9. Send a message to the buyer

To send a message from the seller to the buyer you need to call the operation that corresponds to the message type that the seller chose in Step 2. Get a list of available message types. For example, if the seller chose the "Confirm customization details" message type, you would call the **confirmCustomizationDetails** operation. See the Messaging API reference for the list of operations that are available. In this example, we will call the **confirmCustomizationDetails** operation.

1. Call the **confirmCustomizationDetails** operation of the Messaging API, passing the following parameters:

Path parameters:

Parameter	Description	Required
amazonOrderId	An Amazon order identifier. This specifies the order for which a message is to be sent. Use the Amazon order identifier that you saved in Step 1. Get the order for which the seller wants to send a message. Type: string	Yes

Query Parameter:

Parameter	Description	Required
marketplaceIds	A marketplace identifier. This specifies the marketplace in which the order was placed. You can specify only one marketplace. Type: array[string]	Yes

Body Parameters:

Parameter	Description	Required
text	The custom message to be included in the email to the buyer. Use the message text from Step 4. Get custom message text from the seller (if supported). Type: string	Yes

Parameter	Description	Required
attachments	Attachments to include in the message to the buyer • **uploadDestinationId**. The identifier of the upload destination. Use the **uploadDestinationId** value from Step 7. Create an upload destination. • **fileName**. The name of the file, including the extension. This file name appears in the email message. Use the **fileName** value from Step 5. Get an attachment from the seller (if supported). Type: object	Yes

Request example:

```
GET https://sellingpartnerapi-na.amazon.com/messaging/v1/orders/902-
6786204-
8179805/messages/confirmCustomizationDetails?marketplaceIds=ATVPDKIKX0
DER
{
  "text": "This is the message from the seller to the buyer that will
appear in the email to the buyer.",
  "attachments": [
    {
      "uploadDestinationId": "4370d51b-2954-445c-9f30-EXAMPLE6b9de",
      "fileName": "example.txt"
    }
  ]
}
```

Response

None.

2. Amazon emails the seller's message to the buyer.