The Selling Partner API Developer Guide for Vendors

Contents

What is the Selling Partner API?

The Selling Partner API is a REST-based API that helps Vendors programmatically access their data to automate their business by increasing efficiency, reducing labor requirements, and improving response time to customers.

Selling Partner API HTTP methods

The Selling Partner API supports these HTTP methods (there will be support for more methods in future releases):

1. **GET** – Retrieves resource data or a list of resources.
2. **POST** – Submits an entity to the specified resource, often causing a change in state or side effects on the server.

Selling Partner API endpoints

Selling Partner API endpoints are associated with a particular AWS Region. The AWS Region is important because it is part of the credential scope, which is required for calculating a signature when making a request to the Selling Partner API. See Credential scope.

Selling Partner API endpoints

Geographic region	Endpoint	AWS Region
North America (Canada, US, Mexico, and Brazil marketplaces)	`https://sellingpartnerapi-na.amazon.com`	`us-east-1`
Europe (Spain, UK, France, Germany, Italy, Turkey, U.A.E and India marketplaces)	https://sellingpartnerapi-eu.amazon.com	eu-west-1
Far East (Singapore, Australia, and Japan marketplaces)	`https://sellingpartnerapi-fe.amazon.com`	`us-west-2`

About vendor groups

When you authorize your Selling Partner API application to access your data, you are granting access to the vendor group that is associated with the sign-in credentials for your Vendor Central account. By extension, you are granting access to all of the vendor codes present in the vendor group. Therefore it's important to use the right Vendor Central credentials and vendor group for your Selling Partner API integration.

Using a single vendor group

If you use a single vendor group to manage all of your vendor codes, you can authorize an application to access the data in that vendor group. To do this, register as a developer using the credentials for the Vendor Central account associated with the vendor group.

To be sure that your vendor group contains all of the vendor codes that you want your application to access, check the vendor codes in your vendor group. If you need to, you can add or remove vendor codes from your vendor group at any time.

To check the vendor codes in your vendor group

1. Go to Vendor Central for your marketplace. See Appendix – Vendor Central URIs for a list of URIs by marketplace.
2. Sign in using the credentials for the Vendor Central account with the vendor group that you want your application to access.
3. On the **Settings** menu, click **Contacts**.
4. Review the **Contacts** page to determine if the vendor codes that you want your application to access are present.

To add or remove vendor codes from your vendor group

1. Go to Vendor Central for your marketplace. See Appendix – Vendor Central URIs for a list of URIs by marketplace.
2. Sign in using the credentials for the Vendor Central account with the vendor group for which you want to add or remove vendor codes.
3. Click the **Support** link at the top of the page.
4. On the **Support** page, click the **Contact Us** button.
5. On the **Contact Amazon support** page, click **API Integrations**.
6. In the **Please describe your issue** box, indicate the vendor codes that you want to add or remove from your vendor group.
7. In the **Related vendor or developer ID(s)** box, enter your vendor ID, and click the **Submit** button.

Using multiple vendor groups

Perhaps you use multiple vendor groups to manage your vendor codes. If so, for Selling Partner API integrations we recommend that you either 1) Create a new vendor group containing all of the vendor codes you need, or 2) Choose an existing vendor group and add the vendor codes that you need. You can add or remove vendor codes in a vendor group at any time. However once you have registered as a developer, you cannot change the vendor group associated with that developer. If you want to maintain separate API integrations for different businesses, you can. To do this, register as a developer separately for each vendor code that you want to access. Then you can develop separate API integrations with each registered developer.

To create a new vendor group

1. Go to Vendor Central for your marketplace. See Appendix – Vendor Central URIs for a list of URIs by marketplace.

2. Sign in using the credentials for the Vendor Central account with the vendor group that you want your application to access.
3. Click the **Support** link at the top of the page.
4. On the **Support** page, click the **Contact Us** button.
5. On the **Contact Amazon support** page, click **API Integrations**.
6. In the **Please describe your issue** box, indicate that you would like to create a new vendor group for API integration and provide a list of all the vendor codes to be added to the new vendor group.

Registering as a developer and registering your Selling Partner API application

Here are the steps to register as a developer and register an application for the Selling Partner API.

Step 1. Create an AWS account

You need an AWS account because the Selling Partner API security model uses AWS authentication credentials. If you're not already an AWS customer, you can create a free AWS account. For more information, see AWS Free Tier.

Step 2. Create an AWS Identity and Access Management (IAM) policy

You need an IAM policy to define the permissions for the IAM user that you will create in Step 3. Create an IAM user. The IAM policy will give the IAM user permissions to make calls to the Selling Partner API.

To create an IAM policy

1. Sign in to the AWS Management Console and open the IAM console at `https://console.aws.amazon.com/iam`
2. In the navigation pane at left, click **Policies**.

   If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Click **Get Started.**
3. Click the **Create policy** button.
4. Click the **JSON** tab.
5. Paste the following code into the text box, replacing the existing code, and then click **Review policy**.

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": "execute-api:Invoke",
```

```
"Resource": "arn:aws:execute-api:*:*:*"
}
]
}
```

6. On the **Review policy** page, type a **Name** and a **Description** (optional) for the policy that you are creating. We recommend naming your IAM policy, `SellingPartnerAPI`.
7. Review the policy **Summary** to see the permissions that are granted by your policy, then click **Create policy.**

   Your new IAM policy appears in the list.

For more information, see Creating IAM Policies in the AWS documentation.

Step 3. Create an IAM user

You need an IAM user to get AWS access keys for authenticating your calls to the Selling Partner API. We recommend creating a new IAM user exclusively for this purpose.

We associate your IAM user with your Selling Partner API application during registration.

To create an IAM user

1. Sign in to the AWS Management Console and open the IAM console at `https://console.aws.amazon.com/iam`
2. In the navigation pane at left, click **Users** and then click **Add user**.
3. Type the user name for the new user. This is the sign-in name for AWS.
4. Select **Programmatic access** and then click **Next: Permissions**.
5. On the **Set Permissions** page, click **Attach existing policies directly**.
6. From the list of policies, select the policy that you created in Step 2. Create an AWS Identity and Access Management (IAM) policy.

   Tip: Filter on **Customer managed** to make your policy easier to find.
7. Click **Next: Review** to see all of the choices you made up to this point. When you are ready to proceed, click **Create user**.

   The access key ID for your new IAM user is displayed.
8. Click **View** to view the secret access key. To save the access keys, click **Download .csv** and then save the file to a safe location.

   Important: This is your only opportunity to view or download your AWS secret access keys, which you will need to authenticate your calls to the Selling Partner API. Save the access key ID and secret access key in a safe and secure place. **You will not have access to these access keys again after this step.** If you lose your secret access keys you will need to create a new IAM user with its own new set of keys.

9. Click **Close**.
10. In the **User name** column, click your new IAM user and make a note of the User ARN. You will need it in Step 4. Provide your application registration information.

For more information, see Creating an IAM User in Your AWS Account in the AWS documentation.

Step 4. Provide your application registration information

Provide your application information by signing into Vendor Central and opening a Contact Us case.

To provide your application information

1. Go to Vendor Central for your marketplace. See Appendix – Vendor Central URIs for a list of URIs by marketplace.
2. Sign in using the credentials for the vendor account that identifies you as a Selling Partner API developer. For more information, see Registering as a developer and registering your Selling Partner API application.
3. Click the **Support** link at the top of the page.
4. On the **Support** page, click the **Contact Us** button.
5. On the **Contact Amazon support** page, click **API Integration**.
6. Click the **Registration and access** link.
7. Be sure that **Registration and access** is selected, and then click **Send an email**.
8. In the **Subject (required)** box, enter "Register my application".
9. In the **Describe your issue (required)** box, describe your issue as directed. Be sure to enter your application name, the ARN for your IAM user, and indicate either the Direct Fulfillment or Retail role. Click the **Submit** button.

We will review your ticket and send you an email with the resolution. If we approve your registration request, we will register you as a developer and register your application. After registering as a developer, your vendor account identifies you as a Selling Partner API developer. See Viewing your developer information to see the IAM user ARN and LWA credentials for each of your registered applications.

Viewing your developer information

After registering your Selling Partner API application you can view your developer information on the Developer Central page in Vendor Central.

To view your developer information

1. Go to Vendor Central for your marketplace. See Appendix – Vendor Central URIs for a list of URIs by marketplace.

2. Sign in using the credentials for the vendor account that identifies you as a Selling Partner API developer. For more information, see Registering as a developer and registering your Selling Partner API application.

3. Direct your browser to the Vendor Central developer console for your marketplace. To get the URI:

    1. Go to Appendix – Vendor Central URIs to get the Vendor Central domain for your marketplace.

    2. Append the following to the domain: `/sellingpartner/developerconsole`.

    Example URI for the US marketplace:
    `https://vendorcentral.amazon.com/sellingpartner/developerconsole`

    The **Developer Central** page appears, displaying the application name and the IAM user ARN for each of your registered applications.

4. Click the **View** link for the application for which you want to view Login with Amazon (LWA) credentials.

    The **LWA credentials** box appears, displaying the LWA client identifier for the application authorization.

5. Click the arrow next to **Client secret** to see the LWA client secret for the application authorization.

You will need these LWA credentials to request an LWA access token before making a call to the Selling Partner API. For more information, see Step 1. Request a Login with Amazon access token.

Authorizing your application

The authorization model for the Selling Partner API is based on Login with Amazon (LWA), Amazon's implementation of OAuth 2.0. After registering as a developer and registering your Selling Partner API application, you can authorize your application to call the Selling Partner API to access your vendor account information. For more information about accessing vendor account information, see About vendor groups.

To authorize an application

1. Go to Vendor Central for your marketplace. See Appendix – Vendor Central URIs for a list of URIs by marketplace.

2. Direct your browser to the Vendor Central developer console for your marketplace. To get the URI:

    a. Go to Appendix – Vendor Central URIs to get the Vendor Central domain for your marketplace.

    b. Append the following to the domain: `/sellingpartner/developerconsole`.

    Example URI for the US marketplace:
    `https://vendorcentral.amazon.com/sellingpartner/developerconsole`

    The **Developer Central** page appears, displaying the application name and the IAM user ARN for each of your registered applications.

3. In the **Action** column, next to the application that you want to authorize, click **Edit** and then **Authorize**.

4. On the **Authorize application** page, click the **Generate refresh token** button.

   The refresh token displays in the **Refresh token** box.

The refresh token is a long-lived token that you exchange for a short-lived access token. An access token must be included with every request to the Selling Partner API. Once an access token is issued it is valid for one hour. The same access token can be used for multiple API calls, until it expires. See Step 1. Request a Login with Amazon access token.

Generating a Java SDK with LWA token exchange and authentication

A generated Java SDK makes it easy to call Selling Partner API operations. The SDK handles authorization by exchanging a Login with Amazon (LWA) refresh token for an access token, and authentication by signing requests using AWS Signature Version 4.

> These instructions show you how to generate a Java SDK for the Vendor Orders API using Swagger Code Generator on a computer running Microsoft Windows. The process is the same for users of other operating systems such as macOS or Linux, with the replacement of Windows-specific semantics (for example, `C:\`). Although the following procedure is specific to the Vendor Orders operation, you can modify it to generate an SDK for any operation in the Selling Partner API. To do this, copy the swagger file of your choice (rather than vendorOrders.json) into the directory structure. Then modify the code examples, replacing "vendorOrders.json" with the file name of the swagger file that you are using.

Prerequisites

To complete this procedure you will need:

- **vendorOrders.json**. This is the Swagger file that you use to generate the SDK.
- The **sellingpartner-api-aa-java** folder. This folder contains an authorization and authentication library, along with customized templates for the Swagger Code Generator.

These files were included (along with this developer guide) in the SDK that we provided to you.

To generate a Java SDK with LWA token exchange and authentication

1. Install Java 8 or newer, Apache Maven 3.6 or newer, and GNU Wget and make them available in your `$PATH`.

2. Open a command prompt window and go to a directory where you want to download the Swagger Code Generator.

3. Download the latest version of the Swagger Code Generator.

   For example:

```
3. wget https://repo1.maven.org/maven2/io/swagger/swagger-codegen-
   cli/2.4.13/swagger-codegen-cli-2.4.13.jar -O swagger-codegen-cli.jar
```

swagger-codegen-cli.jar downloads to the current directory.

Note: You can also download from maven.org by directing your browser here: https://repo1.maven.org/maven2/io/swagger/swagger-codegen-cli/2.4.13/swagger-codegen-cli-2.4.13.jar

4. Copy **vendorOrders.json** and **swagger-codegen-cli.jar** into a directory structure that makes sense for you. For this example, we'll copy them to `C:\SwaggerToCL`.

5. Generate the SDK against the templates in the **sellingpartner-api-aa-java** folder.

   For example:

   ```
   java -jar C:\SwaggerToCL\swagger-codegen-cli.jar generate -i
   C:\SwaggerToCL\vendorOrders.json -l java -t [path to
   clients\sellingpartner-api-aa-java directory]/resources/swagger-
   codegen/templates/ -o C:\SwaggerToCL\vendorOrders_JavaCL
   ```

   The SDK is copied to `C:\SwaggerToCL\vendorOrders_JavaCL`

6. Build the AA (Authorization and Authentication) library and add it as a dependency of the SDK:

   a. Navigate to the **sellingpartner-api-aa-java** folder and run `mvn package`. This generates a JAR file in a directory containing all of the required dependencies. The folder should be named **sellingpartnerapi-aa-java-1.0-jar-with-dependencies** or something similar.

   b. Install the JAR file in your local Maven repository.

      For example:

      ```
      mvn install:install-file -Dfile=[path to JAR file in "target"
      folder] -DgroupId=com.amazon.sellingpartnerapi
      -DartifactId=sellingpartnerapi-aa-java -Dversion=1.0 -
      Dpackaging=jar
      ```

      For the `groupId`, `artifactId`, and `version` values in the previous example, use the `groupId`, `artifactId`, and `version` values near the top of the **pom.xml** file in the **sellingpartner-api-aa-java** folder.

   c. Add a dependency on the AA library in the pom.xml of the client library:

      ```
      <dependency>
        <groupId>com.amazon.sellingpartnerapi</groupId>
        <artifactId>sellingpartnerapi-aa-java</artifactId>
        <version>1.0</version>
      </dependency>
      ```

After you have generated your SDK you can use it to make calls to the Selling Partner API. See Connecting to the Selling Partner API using a generated Java SDK.

Connecting to the Selling Partner API using a generated Java SDK

These instructions show you how to use a generated Java SDK to make calls to the Selling Partner API. The SDK exposes classes for configuring your AWS and LWA credentials and uses these to exchange LWA tokens and sign requests for you. For more information, see Generating a Java SDK with authorization and authentication.

Step 1. Configure your AWS credentials

Create an instance of `AWSAuthenticationCredentials`, using the following parameters:

Name	Description	Required
accessKeyId	Your AWS access key Id. For more information, see Step 3. Create an IAM user.	Yes
secretKey	Your AWS secret access key. For more information, see Step 3. Create an IAM user.	Yes
region	The AWS region to which you are directing your call. For more information, see Selling Partner API endpoints.	Yes

For example:

```
import com.amazon.SellingPartnerAPIAA.AWSAuthenticationCredentials;


...


AWSAuthenticationCredentials
awsAuthenticationCredentials=AWSAuthenticationCredentials.builder()
  .accessKeyId("myAccessKeyId")
  .secretKey("mySecretId")
```

Step 2. Configure your LWA credentials

Create an instance of `LWAAuthorizationCredentials`, using the following parameters:

Name	Description	Required
clientId	Your LWA client identifier. For more information, see Viewing your developer information.	Yes
clientSecret	Your LWA client secret. For more information, see Viewing your developer information.	Yes
refreshToken	The LWA refresh token. Get this value when the vendor authorizes your application. For more information, see Authorizing your application.	Yes
endpoint	The LWA authentication server URI.	Yes

For example:

```
import com.amazon.SellingPartnerAPIAA.LWAAuthorizationCredentials;


...


LWAAuthorizationCredentials lwaAuthorizationCredentials =
LWAAuthorizationCredentials.builder()
  .clientId("myClientId")
  .clientSecret("myClientSecret")
  .refreshToken("Aztr|...")
  .endpoint("https://api.amazon.com/auth/o2/token")
  .build();
```

Step 3. Create an instance of the Vendor Orders API and call an operation

With your `AWSAuthenticationCredentials` and `LWAAuthorizationCredentials` instances configured you can create an instance of `vendorOrdersApi` and call an operation.

Note: This example is for using an SDK for the Vendor Orders API, generated in Generating a Java SDK with LWA token exchange and authentication. If you generated an SDK for a different API, replace `vendorOrdersApi` in the following example with text that matches your API. For example, use `vendorShippingAPI` for an SKD generated for the Vendor Shipping API.

For example:

```
vendorOrdersApi vendorOrdersApi = new vendorOrdersApi.Builder()
  .awsAuthenticationCredentials(awsAuthenticationCredentials)
  .lwaAuthorizationCredentials(lwaAuthorizationCredentials)
  .endpoint(https://sellingpartnerapi-na.amazon.com)
  .build();
vendorOrdersApi.getMarketplaceParticipations();
```

Generating a Java client library

A generated Java client library can help you construct a URI and add headers for making requests. A Java Client library does not help with authorizing and authenticating your request, however. You still need to write code to request a Login with Amazon access token and create and sign your request.

Note: For most Java developers, the easiest way to connect to the Selling Partner API is to use a generated Java SDK that includes authentication and authorization. For more information, see Generating a Java SDK with LWA token exchange and authentication**.**

These instructions show you how to generate a Java client library for the Vendor Orders API using Swagger Code Generator on a computer running Microsoft Windows. The process is the same for users of other operating systems such as macOS or Linux, with the replacement of Windows-specific semantics (for example, `C:\`). Although the following procedure is specific to the Vendor Orders operation, you can modify it to generate client libraries for any operation in the Selling Partner API. To do this, copy the swagger file of your choice (rather than vendorOrders.json) into the directory structure. Then modify the code examples, replacing "vendorOrders.json" with the file name of the swagger file that you are using.

Prerequisites

To complete this procedure you will need:

- **vendorOrders.json**. This is the Swagger file that you use to generate the SDK.

This file was included (along with this developer guide) in the SDK that we provided to you.

To generate a Java client library

1. Install Java 8 or newer, Apache Maven 3.6 or newer and wget and make them available in your $PATH.
2. Open a command prompt window and go to a directory where you want to download the Swagger Code Generator.
3. Download the latest version of the Swagger Code Generator.

   For example:

   ```
   wget http://central.maven.org/maven2/io/swagger/swagger-codegen-cli/2.4.7/swagger-codegen-cli-2.4.7.jar -O swagger-codegencli.jar
   ```

   swagger-codegen-cli.jar downloads to the current directory.

   Note You can also download from maven.org by directing your browser here: http://central.maven.org/maven2/io/swagger/swagger-codegen-cli/2.4.7/swagger-codegen-cli-2.4.7.jar.
4. Copy **vendorOrders.json** and **swagger-codegen-cli.jar** into a directory structure that makes sense for you. For this example, we'll copy them to `C:\SwaggerToCL`.
5. Generate the client library.

   For example:

```
java -jar C:\SwaggerToCL\swagger-codegen-cli.jar generate -i
C:\SwaggerToCL\vendorOrders.json -l java -o
C:\SwaggerToCL\vendorOrders.json_JavaCL
```

The client library is copied to `C:\SwaggerToCL\vendorOrders_JavaCL`.

Connecting to the Selling Partner API

These instructions show the steps for making a call to the Selling Partner API. The instructions are primarily for developers coding in languages other than Java. For developers coding in Java, we recommend using a generated Java SDK, which helps you create the request and makes authorization and authentication much easier. For more information, see Generating a Java SDK with LWA token exchange and authentication.

Before your application can connect to the Selling Partner API, you must register it and it must be authorized by a selling partner. See Registering your Selling Partner API application and Authorizing your application.

Step 1. Request a Login with Amazon access token

A Login with Amazon (LWA) access token authorizes your application to take certain actions on behalf of a selling partner. An access token expires one hour after it is issued, and must be included with every request to the Selling Partner API.

To request an access token, make a secure HTTP POST to the LWA authentication server (https://api.amazon.com/auth/o2/token) with the following parameters:

Name	Description	Required
grant_type	The type of access grant requested. Must be `refresh_token`.	Yes
refresh_token	Get this value when you authorize your application to access your selling account. See Authorizing your application.	Yes
client_id	Get this value when you register your application. See Viewing your developer information.	Yes
client_secret	Get this value when you register your application. See Viewing your developer information.	Yes

For example:

```
POST /auth/o2/token HTTP/1.1

Host: api.amazon.com

Content-Type: application/x-www-form-urlencoded;charset=UTF-8
grant_type=refresh_token&refresh_token=Atzr|IQEBLzAtAhRPpMJxdwVz2Nn6f2y-
tpJX2DeX...&client_id=foodev&client_secret=Y76SDl2F
```

Tip To avoid getting an untrusted certificate authority (CA) error when calling the LWA authorization server, be sure to update your trust store so that your application trusts the LWA server.

cURL

If you prefer, you can use cURL to request the access token instead. For example:

```
curl -k -X POST -H 'Content-Type: application/x-www-form-urlencoded' -d
'grant_type=refresh_token& refresh_token=Atzr|IQEBLzAtAhRPpMJxdwVz2Nn6f2y-
tpJX2DeX...&client_id=foodev&client_secret=Y76SDl2F
https://api.amazon.com/auth/O2/token
```

Response

A successful response includes the following values:

Name	Description
access_token	The access token. Maximum size: 2048 bytes.
token_type	The type of token returned. Must be `bearer`.
expires_in	The number of seconds before the access token becomes invalid.
refresh_token	The refresh token that you submitted in the request. Maximum size: 2048 bytes.

For example:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "access_token":"Atza|IQEBLjAsAhRmHjNgHpi0U-Dme37rR6CuUpSREXAMPLE",
  "token_type":"bearer",
  "expires_in":3600,
  "refresh_token":"Atzr|IQEBLzAtAhRPpMJxdwVz2Nn6f2y-tpJX2DeXEXAMPLE"
}
```

For more information, see Authorization Code Grant in the Login with Amazon documentation.

Step 2. Construct a Selling Partner API URI

Here are the components of a Selling Partner API URI:

Name	Description	Example
HTTP method	One of the Selling Partner API HTTP methods.	`GET`

Endpoint	A Selling Partner API endpoint.	`https://sellingpartnerapi-eu.amazon.com`
Path	Selling Partner API section/version number of the section/resource	`vendor/orders/v1/purchaseOrders`
Query string	The query parameters.	`?limit={example}&createdAfter={example}&createdBefore={example}&sortOrder={example}&nextToken={example}&includeDetails={example}`

For example:

```
GET //https://sellingpartnerapi-eu.amazon.com/vendor/orders/v1/purchaseOrders?limit={example}&createdAfter={example}&createdBefore={example}&sortOrder={example}&nextToken={example}&includeDetails={example}
```

Note: If you are creating an application using Java, you can use a generated Java client library to help you construct a URI. For more information, see Generating a Java client library.

Step 3. Add headers to the URI

Add headers to the URI that you constructed in Step 2. Construct a Selling Partner API URI.

Here are the HTTP headers that you include in requests to the Selling Partner API:

Request headers

Name	Description	Required
`host`	The Selling Partner endpoint. See Selling Partner API endpoints	Yes
`x-amz-access-token`	The Login with Amazon access token. See Step 1. Request a Login with Amazon access token.	Yes
`x-amz-date`	The date and time of your request	Yes
`user-agent`	Your application name and version number, platform, and programming language. These help us diagnose and fix problems you might encounter with the service. See Include a User-Agent header in all requests.	Yes

Here is an example of a request to the Selling Partner API with URI and headers but no signing information:

```
GET //https://sellingpartnerapi-eu.amazon.com/vendor/orders/v1/purchaseOrders?limit={example}&createdAfter={example}&createdBefore={example}&sortOrder={example}&nextToken={example}&includeDetails={example} HTTP/1.1

host: https://sellingpartnerapi-eu.amazon.com
```

```
user-agent: My Selling Tool/2.0 (Language=Java/1.8.0.221;
Platform=Windows/10)

x-amz-access-token=Atza|IQEBLjAsAhRmHjNgHpi0U-Dme37rR6CuUpSREXAMPLE

x-amz-date: 20190430T123600Z
```

Note: If you are creating an application using Java, you can use a generated Java client library to help you add headers to the URI. For more information, see Generating a Java client library.

To sign a request, see Step 4. Create and sign your request.

Step 4. Create and sign your request

The Selling Partner API uses Signature Version 4 Signing Process for authenticating requests. When you send HTTP requests to the Selling Partner API, you sign the requests so that we can identify who sent them. You sign requests using your AWS access key, which consists of an access key ID and a secret access key. For information about getting your AWS access key, see Step 3. Create an IAM user.

Note You need to learn how to sign HTTP requests only when you manually create them. When you use the one of the AWS SDKs to calculate signatures for you, the SDK automatically signs the requests with the access key that you specify when you configure it. When you use an SDK you don't need to learn how to sign requests yourself. Java developers, for example, can use AWS4Signer.java from the AWS SDK for Java as a model for calculating a signature. You can find SDKs for other languages in the AWS GitHub repository.

To create and sign your request, complete the following:

1. Create a canonical request

   Follow the instructions in Task 1: Create a Canonical Request for Signature Version 4 in the AWS documentation, using this guidance:

   - See Step 3. Add headers to the URI for an example of an unsigned request to start with when you create your canonical request.
   - Use SHA-256 for the hash algorithm.
   - Do not put authentication information in the query parameters. Put it in the Authorization header.

2. Create a string to sign

   Follow the instructions in Task 2: Create a String to Sign for Signature Version 4 in the AWS documentation, using this guidance:

   - The algorithm designation value is `AWS4-HMAC-SHA256`
   - To determine the credential scope, see Credential scope.

3. Calculate the signature

   Follow the instructions in Task 3: Calculate the Signature for AWS Signature Version 4 in the AWS documentation.

4. Add the signing information

Follow the instructions in Task 4: Add the Signature to the HTTP Request in the AWS documentation, using this guidance:

- Do not add signing information to the query string. Add it to the Authorization header.
- See Authorization header for details about creating an Authorization header.

The following example shows what a request might look like after you've added the signing information to it using the Authorization header.

```
GET //https://sellingpartnerapi-
eu.amazon.com/vendor/orders/v1/purchaseOrders?limit={example}&created
After={example}&createdBefore={example}&sortOrder={example}&nextToke
n={example}&includeDetails={example} HTTP/1.1

Authorization: AWS4-HMAC-SHA256 Credential=AKIDEXAMPLE/20190430/us-
east- 1/execute-api/aws4_request, SignedHeaders=host;user-agent;x-
amz-access-token,
Signature=5d672d79c15b13162d9279b0855cfba6789a8edb4c82c400e06b5924aE
XAMPLE

host: https://sellingpartnerapi-eu.amazon.com

user-agent: My Selling Tool/2.0 (Language=Java/1.8.0.221;
Platform=Windows/10)

x-amz-access-token=Atza|IQEBLjAsAhRmHjNgHpi0U-Dme37rR6CuUpSREXAMPLE

x-amz-date: 20190430T123600Z
```

Credential scope

The credential scope is a component of the “string to sign” that you create when you sign a request to the Selling Partner API. See Step 4. Create and sign your request.

Credential scope is represented by a slash-separated string of dimensions, as shown in the following table:

Dimension	Description	Example
Date	An eight-digit string representing the year (YYYY), month (MM), and day (DD) of the request.	`20190430`
AWS region	The region you are sending the request to. See Selling Partner API endpoints.	`us-east-1`
Service	The service you are requesting. You can find this value in the endpoint. See Selling Partner API endpoints.	`execute-api`
Termination string	A special termination string. For AWS Signature Version 4, the value is `aws4_request`	`aws4_request`

For example:

```
20190430/us-east-1/execute-api/aws4_request
```

Important The date that you use as part of your credential scope must match the date of your request, as specified in the **x-amz-date** header. For details, see Handling Dates in Signature Version 4 in the AWS documentation.

For more information, see Step 4. Create and sign your request.

Authorization header

The Authorization header contains the signing information for a request. Although the header is named "Authorization", the signing information is used for authentication.

Here are the components of an Authorization header:

Header component	Description
The algorithm used for signing	The hash algorithm used throughout the signing process. The Selling Partner API requires SHA-256. You specify this in Step 4. Create and sign your request.
`Credential`	Your AWS access key ID plus the Credential scope. You get your AWS access key ID in see Step 3. Create an IAM user.
`SignedHeaders`	A list of all the HTTP headers that you included with the signed request. For an example, see Step 3. Add headers to the URI.
`Signature`	The signature calculated in Step 4. Create and sign your request.

For example:

```
Authorization: AWS4-HMAC-SHA256 Credential=AKIDEXAMPLE/20190430/us-east-
1/execute-api/aws4_request, SignedHeaders=host;user-agent;x-amz-access-
token;x-amz-date,
Signature=5d672d79c15b13162d9279b0855cfba6789a8edb4c82c400e06b5924aEXAMPLE
```

For more information, see Step 4. Create and sign your request.

Response format

In response to an HTTP request, the Selling Partner API returns response headers and a JSON response message.

Response headers

Name	Description
`Content-Length`	Standard HTTP response header.
`Content-Type`	Standard HTTP response header.
`Date`	Standard HTTP response header.
`x-amzn-ErrorType`	Error type.
`x-amzn-RequestId`	Request identifier. Include this if you contact us for support.

Success response

If your request is successful, the Selling Partner API returns the data requested. Here is an example of a successful response to a call to the getOrders API.

```
HTTP/1.1 200 OK

Content-Length: 368

Content-Type: application/json Date: Thu, 07 Jun 2019 22:23:31 GMT

x-amzn-RequestId: 6875f61f-6aa1-11e8-98c6-9b9a3a7283a4


{
  "pagination": {
    "nextToken": "2YgYW55IGNhcm5hbCBwbGVhc3VyZS4"
  },
  "orders": [
    {
      "order": {
        "purchaseOrderNumber": " L8266355",
        "orderDetails": {
          "purchaseOrderDate": "2019-07-16T19:17:34.304Z",
          "purchaseOrderType": "RegularOrder",
          "paymentMethod": "Invoice",
          "buyingParty": {
            "partyId": "NAG1"
          },
          "sellingParty": {
            "partyId": "999US"
          },
          "shipToParty": {
            "partyId": "NAG1"
          },
          "billToParty": {
            "partyId": "NAG1"
          },
          "items": [
            {
              "itemSequenceNumber": "00001",
              "amazonProductIdentifier": " ABC123434",
              "vendorProductIdentifier": "028877454078",
              "title": "Baby Dove Baby Wipes Rich Moisture (50 Pieces)",
              "orderedQuantity": {
                "amount": "10",
                "unitOfMeasure": "Cases",
                "unitSize": "5"
              },
```

```
              "isBackOrderAllowed": "N",
              "netCost": {
                "amount": "10.2",
                "currencyCode": "USD"
              }
            }
          ]
        }
      }
    }
  ]
}
```

Error response

If your request is unsuccessful, the Selling Partner API returns an error response. Here are the elements of the response message in an error response:

Response message

Element	Description	Required
code	HTTP status code.	Yes
message	Explanation of the error condition.	Yes
details	Link to additional information.	No.

Here is an example of an error response:

```
HTTP/1.1 400 Bad Request
Content-Length: 117
Content-Type: application/json
Date: Fri, 08 Jun 2019 00:48:28 GMT
x-amzn-ErrorType: ValidationException
x-amzn-RequestId: a8c8d99a-6ab5-11e8-b0f8-19363980175b

{
  "code": "400",
  "message": "The request could not be understood by the server due to
malformed syntax."
}
```

Throttling: limits to how often you can submit requests

Throttling is the process of limiting the number of requests that your application can make to the Selling Partner API over a given period of time. Throttling protects the web service from being overwhelmed with requests and helps to ensure that all authorized applications have access to the web service.

Throttling limits are based on the volume of requests over time that your application makes to the Selling Partner API on behalf of a specific vendor account. Throttling limits are tracked per individual API service.

Throttling limits for all operations are:

- **Rate** (requests per second): 1
- **Burst**: 1

Include a User-Agent header in all requests

A User-Agent header identifies your application, its version number, and the platform and programming language that you are using. You must include a User-Agent header with every request that you submit to the Selling Partner API. Doing this helps us to more effectively diagnose and fix problems, helping to improve your experience using the Selling Partner API.

To create a User-Agent header, begin with the name of your application, followed by a forward slash, followed by the version of the application, followed by a space, an opening parenthesis, the Language name value pair, and a closing parenthesis. The Language parameter is a required attribute, but you can add additional attributes separated by semicolons.

The following pseudocode illustrates a minimally acceptable User-Agent header:

```
AppId/AppVersionId (Language=LanguageNameAndOptionallyVersion)
```

Here is an example of a User-Agent header that might be used by an external application integrator:

```
My Selling Tool/2.0 (Language=Java/1.8.0.221; Platform=Windows/10)
```

If you are a large vendor who is integrating through your own IT department, you might want create a User-Agent header like the following, so the Selling Partner API could help you troubleshoot using the Host attribute:

```
MyCompanyName/build1611 (Language=Perl; Host=jane.desktop.example.com)
```

To specify additional attributes, use the format `AttributeName=Value;`, separating each name value pair with a semicolon. If you need to use a backslash (\), quote it with another backslash (\\). Similarly, quote a forward slash in the application name (\/), an opening parenthesis in the application version (\(), an equal sign in the attribute name (\=), and both a closing parenthesis (\)), and a semicolon (\;) in attribute values.

Because the User-Agent header is transmitted in every request, it is a good practice to limit the size of the header. The Selling Partner API will reject a User-Agent header if it is longer than 500 characters.

The Selling Partner API sandbox

The Selling Partner API provides a sandbox environment that allows you to test your applications without affecting production data or triggering real-world events. Making sandbox calls to the Selling Partner API is identical to making production calls except you direct the calls to the Selling Partner API sandbox endpoints. The authorization and authentications models are the same for sandbox calls and production calls. Calling the sandbox endpoints returns static, mocked responses for all Selling Partner APIs. You can refer to these mocked responses in the Swagger model JSON file for the API that you want to call. For more information, see How to make a sandbox call to the Selling Partner API.

The Selling Partner API sandbox works like many mocking frameworks, in that it uses pattern matching to return a specified response when the specified parameters are present. A developer receives a response defined in the `x-amazon-spds-sandbox-behaviors` object when they send a request that matches the specified parameters. If the API requires any parameters that aren't specified in the `x-amazon-spds-sandbox-behaviors` object, the sandbox provides the response regardless of the parameter values in the request, as long as the request is valid.

Important: The sandbox is for testing functionality, not scalability testing. Calls to sandbox endpoints are subject to these throttling limits: **rate** = five requests per second; **burst** = 15. For more information about throttling see Throttling: limits to how often you can submit requests.

For the Public Release version of the Selling Partner API we will include input validation for sandbox calls so you receive realistic error messages for invalid requests.

How to make a sandbox call to the Selling Partner API

Prerequisites

To complete this procedure you will need a Swagger model JSON file (for example, vendorOrders.json).

Swagger model JSON files were included (along with this developer guide) in the SDK that we provided to you.

Step 1. Check the JSON file for request parameters

1. Open the JSON file for the API that you want to call.
2. Search for "x-amazon-spds-sandbox-behaviors".

   The `x-amazon-spds-sandbox-behaviors` object of the JSON file contains request and response examples for sandbox calls to the API. If the request example contains parameters, use them in the following step.

Step 2. Make a sandbox call to an API

Make a sandbox call to an API in the same way you would make a production call, with these differences:

1. If the request object in the `x-amazon-spds-sandbox-behaviors` object of the JSON file contains one or more parameter/value pairs, specify these in your call.
2. Direct your call to one of the Selling Partner API sandbox endpoints.

You should receive a response that matches the payload object contained in the `x-amazon-spds-sandbox-behaviors` object of the JSON file.

Selling Partner API sandbox endpoints

The Selling Partner API has sandbox endpoints for the North America, Europe, and Far East selling regions. For more information, see The Selling Partner API sandbox.

Selling region	Endpoint	AWS Region
North America (Canada, US, Mexico, and Brazil marketplaces)	https://sandbox.sellingpartnerapi-na.amazon.com	us-east-1
Europe (Spain, UK, France, Germany, Italy, Turkey, U.A.E, and India marketplaces)	https://sandbox.sellingpartnerapi-eu.amazon.com	eu-west-1
Far East (Singapore, Australia, and Japan marketplaces)	https://sandbox.sellingpartnerapi-fe.amazon.com	us-west-2

Appendix – Vendor Central URIs

Here are the Vendor Central URIs by marketplace.

North America

Marketplace	Vendor Central URI
Canada	https://vendorcentral.amazon.ca
US	https://vendorcentral.amazon.com
Mexico	https://vendorcentral.amazon.com.mx

Europe

Marketplace	Vendor Central URI
Spain	https://vendorcentral.amazon.es
UK	https://vendorcentral.amazon.co.uk
France	https://vendorcentral.amazon.fr
Germany	https://vendorcentral.amazon.de
Italy	https://vendorcentral.amazon.it
India	https://www.vendorcentral.in