Amazon Marketplace Web Service (Amazon MWS) Developer Guide

Contents

Amazon Marketplace Web Service (Amazon MWS) Developer Guide

What's New

Below is the history of this document.

Change	Description	Release Date
Tenth Release, API Version 2009-01-01	Added information that **Marketplace** is now an optional parameter and no longer used for authentication. Added information on the new **MarketplaceIdList** parameter added to support multiple marketplaces. Added throttling limit change from 1,000 to 10,000 requests per hour. Corrected typos.	June 2011
Ninth Release, API Version 2009-01-01	Amazon MWS Developer Guide revised to include new information and to include the content formerly residing in the Getting Started Guide. Individual API sections have been removed and placed in their own API section reference files.	January 2011
Eighth Release, API Version 2009-01-01	Added four new report types for FBA to the **ReportType** topic.	October 2010
Seventh Release, API Version 2009-01-01	Rewrote several topics to describe the Amazon MWS registration process and the developer account identifier and credentials. Updated guide to refer to Amazon MWS developer account identifier. Added new report types for Product Ads and FBA.	June 2010

If You Are New to Amazon Marketplace Web Service

What is Amazon MWS?

Amazon Marketplace Web Service (Amazon MWS) is an integrated web service API that helps Amazon sellers to programmatically exchange data on listings, orders, payments, reports, and more. Data integration with Amazon enables high levels of selling automation, which can help sellers grow their business. By using Amazon MWS, sellers can increase selling efficiency, reduce labor requirements, and improve response time to customers.

There are no fees associated with Amazon MWS, but to use the Amazon MWS API you must have an Amazon MWS-eligible seller account and you must register to use Amazon MWS.

What Amazon MWS Provides

With Amazon MWS, you can build applications for your own Amazon seller account. You can also build applications for other sellers to help them manage their online business. Using Amazon MWS you can create applications that look up products for sale, download orders for fulfillment, confirm shipment, and schedule and receive reports. These API operations are accessible by using a REST-like interface.

Amazon MWS provides the following features:

- Inventory management— You can perform batch uploads of inventory, add products, check inventory levels, examine pricing information, and other inventory management tasks.
- Order management— You can download order information, obtain payment data, acknowledge orders, and schedule reports.
- Reports management— You can request a variety of reports as well as query the status of these reports, and then download them.

For Fulfillment by Amazon (FBA) sellers, Amazon MWS also allows you to:

- Create inbound shipments to an Amazon fulfillment center— You can automate the process for creating labels for units you ship to an Amazon fulfillment center.
- Check status of inbound shipments— You can check to see if your shipment has reached a fulfillment center and, if so, whether the shipment has been processed.
- Submit fulfillment orders— By integrating your system with Amazon MWS, you can enable your customers to submit multi-channel fulfillment orders at any time. There is no lag time while you process or batch orders.
- Track and manage outbound shipment requests— Once orders have left an Amazon fulfillment center, you can track shipments and keep your customers aware of arrival times.

Registering for an Amazon MWS-eligible seller account

To be eligible to use Amazon MWS, you must have at least one of the following Amazon accounts:

- Non-individual Amazon seller account
- Amazon WebStore
- Amazon Product Ads
- Checkout by Amazon

Registering to use Amazon MWS

Amazon MWS is a secure environment that uses signatures for authentication and lets sellers delegate calling rights to developers by using the Amazon MWS authorization service. To use Amazon MWS, you must have an Amazon MWS-eligible seller account and you must register as an Amazon MWS developer at one of the following sites:

- **CA:** *http://developer.amazonservices.ca*
- **DE:** *http://developer.amazonservices.de*
- **FR:** *http://developer.amazonservices.fr*
- **JP:** *http://developer.amazonservices.jp*
- **UK:** *http://developer.amazonservices.co.uk*
- **US:** *http://developer.amazonservices.com*

There are three options available when you sign up to use Amazon MWS:

I want to access my own Amazon seller account with MWS.— Select this option when you sign up to use Amazon MWS for your own Amazon seller account, Amazon MWS will assign a developer account identifier to you. When you make Amazon MWS requests, you'll use the developer account credentials that are associated with your developer account, plus the merchant Id for your seller account.

I want to use an application to access my Amazon seller account with MWS.— Select this option if you want to use an application to access your Amazon seller account using Amazon MWS. When you register, you must enter the developer account identifier for the application you will be using. The final page of the Amazon MWS registration process shows your merchant Id. You will use this identifier in the application you use.

I want to give a developer access to my Amazon seller account with MWS.— Select this option when you want to authorize a third-party developer to access your account with Amazon MWS.

Registering as a developer

Once you have completed registration, you receive several important credentials. Your merchant Id, marketplace Id, developer account identifier, AWS access key Id, and Secret Key are displayed on the final page of the Amazon MWS registration process. This information is not e-mailed to you. You should print this page or save it to your hard drive. If you need to see the credentials and identifier again, you can repeat the Amazon MWS registration process. Registering multiple times does not affect your original registration.

If you are developing Amazon MWS applications or providing Amazon MWS-related development services to other sellers, you must provide your developer account identifier to those sellers so that they can authorize you to access their Amazon seller accounts with Amazon MWS.

The following list shows an example of the credentials you receive when you register to access your own seller account using Amazon MWS:

- Developer Account Identifier (a 12-digit identifier): 1234-3214-4321
- Access Key ID (a 20-character, alphanumeric identifier): 022QF0EXAMPLEH9DHM02

• Secret Key (a 40-character identifier): kWcrlEXAMPLEM/LtmEENI/aVmYvHNif5zB+d9+ct

The access key Id is associated with your Amazon MWS registration. You include it in all Amazon MWS requests to identify yourself as the sender of the request. The access key Id is not a secret. To provide proof that you truly are the sender of the request, you must also include a digital signature. For all requests except those generated using the Amazon MWS client libraries, you calculate the signature using your Secret Key. Amazon MWS uses the access key Id in the request to look up your Secret Key and then calculates a digital signature with the key. If the signature Amazon MWS calculates matches the signature you sent, the request is considered authentic. Otherwise, the request fails authentication and is not processed.

Note: Your Secret Key is a secret that only you and Amazon MWS should know. It is important to keep it confidential to protect your account. Never include it in your requests to Amazon MWS, and never e-mail it to anyone. Do not share it outside your organization, even if an inquiry appears to come from Amazon MWS or anyone else at Amazon. No one who legitimately represents Amazon will ever ask you for your Secret Key.

Authorizing a developer or an application to access your account

To authorize a third-party developer, the developer must first give you his or her Amazon MWS developer account identifier. You enter this developer account identifier when you register to authorize the developer to access your account. You must then give the developer your merchant Id so Amazon MWS requests can be made on your behalf. The final page of the Amazon MWS registration process shows your merchant Id and marketplace Id.

Developing an Amazon MWS application

To create applications that use Amazon MWS, you need a development environment for Java, C#, or PHP. If you are a developer who is using Amazon MWS to build software for your own eligible Amazon account, use the email address for that Amazon seller account when you sign up for Amazon MWS, selecting the option to use your own Amazon MWS-eligible seller account with Amazon MWS. For more information, see the topic Signing Up for Amazon MWS. If you are a developer who wants to use Amazon MWS to build professional software for other eligible Amazon sellers, you must first sign up for Amazon MWS with your own Amazon MWS-eligible seller account. On the last page of your Amazon MWS registration, make note of your developer account identifier, as you'll need to provide this number to sellers who want to use your developer services or application.

When your software is ready for testing with additional seller accounts, you can provide sellers with your developer account identifier, which they can use to register forAmazon MWS. The sellers log into their ownAmazon MWS-eligible seller accounts and insert your developer account identifier during the registration process. The sellers need to make note of the merchant Id for their account and pass that information to you. You will then be able to make Amazon MWS requests on their behalf, using their seller credentials and your developer credentials.

For further assistance with building your application, we suggest you use the other available resources for Amazon MWS. For more information, see *Related Resources* .

Note: Please see the FAQ pages on the Amazon MWS portals for additional information about developing Amazon MWS applications for Amazon sellers and the authorization process.

- **CA:** *http://developer.amazonservices.ca*
- **DE:** *http://developer.amazonservices.de*

- **FR:** *http://developer.amazonservices.fr*
- **JP:** *http://developer.amazonservices.jp*
- **UK:** *http://developer.amazonservices.co.uk*
- **US:** *http://developer.amazonservices.com*

What you should know about Amazon Marketplace Web Service (Amazon MWS)

Amazon Marketplace Web Service endpoints

You access Amazon MWS through a URL endpoint for your Amazon marketplace. The following table shows the currently active endpoints for Amazon MWS.

Amazon Marketplace	Amazon MWS Endpoint	Amazon MWS Website
CA	https://mws.amazonservices.ca	http://developer.amazonservices.ca
DE	https://mws.amazonservices.de	http://developer.amazonservices.de
FR	https://mws.amazonservices.fr	http://developer.amazonservices.fr
JP	https://mws.amazonservices.jp	http://developer.amazonservices.jp
UK	https://mws.amazonservices.co.uk	http://developer.amazonservices.co.uk
US	https://mws.amazonservices.com	http://developer.amazonservices.com

SSL requirements for inbound connections

Innbound connections to Amazon MWS endpoints work with SSL version 3 or SSL version 3.1. Inbound connections do not work with SSL version 2.

Throttling: Limits to how often you can submit requests

To use Amazon Marketplace Web Service (Amazon MWS) successfully, you need to understand throttling. Throttling is the process of limiting the number of requests you can submit in a given amount of time. A request can be when you submit an inventory feed or when you make an order report request. Throttling protects the web service from being overwhelmed with requests and ensures all authorized developers have access to the web service.

Amazon MWS uses a variation of the leaky bucket algorithm to meter the web service and implement throttling. The algorithm is based on the analogy where a bucket has a hole in the bottom from which water leaks out at a constant rate. Water can be added to the bucket intermittently, but if too much water is added at once or if water is added at too high an average rate, the water will exceed the capacity of the bucket.

To apply this analogy to Amazon MWS, imagine that the bucket represents the maximum request quota, which is the maximum number of requests you can make at one time. The hole in the bucket represents the restore rate, which is the amount of time it takes to be able to make new requests. So, if you submit too many requests at once, then the bucket overflows and, in the case of Amazon MWS,

throttling occurs. If you fill up the bucket, it takes some time before you can add more water to the bucket since the water leaks from the bucket at a steady rate. So the ability to submit more requests after you have reached the maximum request quota is governed by the restore rate, the time it takes to allow you to make new requests.

The definitions of these three values that control Amazon MWS throttling are:

- Request quota - The number of requests that you can submit at one time without throttling. The request quota decreases with each request you submit, and increases at the restore rate.
- Restore rate (also called the recovery rate) - The rate at which your request quota increases over time, up to the maximum request quota.
- Maximum Request quota (also called the burst rate) - The maximum size that the request quota can reach.

Amazon MWS limits requests to 10,000 total requests per hour for each Amazon seller account and Amazon MWS developer account pair. While most Amazon MWS operations have a maximum request quota of 10 requests and a restore rate of one new request every minute, a few operations allow you to submit more requests initially (higher maximum request quota) but take longer to allow additional requests (a lower restore rate). For example, the `RequestReport` operation has a maximum request quota of 15, but the restore rate is one new request every two minutes.

To apply these ideas, consider this example. Imagine that you want to use the `SubmitFeed` operation to submit 25 inventory update feeds. The `SubmitFeed` operation has a request quota of 15 and a restore rate of one new request every two minutes. If you submit all 25 feed requests at once, your requests will be throttled after 15 requests. You would then have to resubmit 10 feed requests once the request quota had been restored. Since the restore rate is one request every two minutes, it would take 20 minutes for you to be able to submit the remaining 10 feed requests. So, instead of submitting all the requests and having to resubmit the requests that were throttled, you could automate your process to submit feed requests incrementally.

For example, you could submit 10 feed requests (out of your original 25 feeds), and the request quota would still have five requests left over. You could then wait 10 minutes, and the restore rate would have increased the request quota to 10 (one request every two minutes for 10 minutes gives you five new requests). You could then submit 10 more feed requests. For the remaining five feed requests, you could wait ten more minutes and then submit them. If all things go well, you would have submitted all 25 of your inventory feeds in about 20 minutes.

You should consider automating your requests and have a fallback process where, if throttling occurs because you reached the maximum request quota or the web service experienced high traffic volumes, you could slow down the number of requests you make and resubmit requests that initially failed.

Tips on avoiding throttling

There are several things you can do to make sure your feeds and submissions are processed successfully:

- Know the throttling limit of the specific request you are submitting.
- Have a "back off" plan for automatically reducing the number of requests if the service is unavailable. The plan should use the restore rate value to determine when a request should be resubmitted.
- Submit requests at times other than on the hour or on the half hour. For example, submit requests at 11 minutes after the hour or at 41 minutes after the hour.
- Take advantage of times during the day when traffic is likely to be low on Amazon MWS, such as early evening or early morning hours.

Using the Amazon MWS client libraries

Each Amazon MWS API section has its own client library that contains code for doing many common tasks when working with Amazon MWS. By using the Amazon MWS client library, you save time and you know the request you send is correctly formatted. For example, the Amazon MWS client library performs the following tasks for you:

- Request Signature - creates a valid request HMAC-SHA signature. Each request must have a valid signature or the request is rejected. A request signature is calculated using your Secret Access Key, which is a shared secret, given to you when you registered, and known only to you and Amazon MWS.
- Timestamp - adds a timestamp on each request you submit. Each request must contain the timestamp of the request.
- Requests - builds a valid request for you based on the operation you select and the parameters you enter.
- User-Agent header - creates the optional User-Agent header.
- Stream - creates a stream you use to receive downloaded reports when using the `GetReport` operation.

If you create your own client library

You can create your own client library for use with Amazon MWS. Your code should construct and sign a request in the format expected by Amazon MWS, and then you parse the resulting XML response.

You access Amazon MWS by following these steps:

1. Determine the correct Amazon MWS endpoint to use.
2. Determine the throttling limits for the operation you want to submit.
3. Construct a query string for the request.
4. Sign the query string and create the request.
5. Send the correctly formatted URL request and an HTTP header containing the User-Agent header to the endpoint for your Amazon marketplace.
6. Parse the response.

Request Format

Amazon MWS supports query requests for calling web service actions. Query requests are simple HTTP requests, using the GET or POST method with query parameters in the URL or HTTP body, respectively. Amazon MWS requires the use of HTTPS in order to prevent third-party eavesdropping on your communication with Amazon.

Each of the HTTP header lines must be terminated with a carriage return and a line feed. Query requests must contain an Action parameter to indicate the action to be performed. The response is an XML document.

Creating the Canonicalized Query String

To create an Amazon MWS query request, you first construct a query string with the query information. You then sign this query string and include it in the request submission. All parameters must be in natural-byte order when calculating the signature. The string consists of:

- The HTTP action. This value is most often **POST**.

- The domain name of the request, such as https://mws.amazonservices.com/. For a list of endpoints for each Amazon marketplace, see the Amazon MWS Endpoints section in this guide. After the endpoint is a forward slash (/), which separates the endpoint from the parameters.
- *AWSAccessKeyId*—Your Amazon MWS account is identified by your access key Id, which Amazon MWS uses to look up your Secret Access Key.
- *Action*—The action you want to perform on the endpoint, such as the operation `GetFeedSubmissionResult`.
- *Parameters*—Any required and optional request parameters.
- *Marketplace*—A deprecated parameter once used for authentication but no longer required nor used.
- *MarketplaceIdList*—An optional structured list of marketplace Ids for supporting sellers registered in multiple marketplaces. For example, two marketplace Ids would be formatted as: &MarketplaceIdList.Id.1=ATVPDKIKX0DER&MarketplaceIdList.Id.2=A1F83G8C2ARO7P
- *Merchant* or *SellerId*—Your seller or merchant Id.
- *SignatureMethod*—The HMAC hash algorithm you are using to calculate your signature, either **HmacSHA256** or **HmacSHA1**.
- *SignatureVersion*—Which signature version is being used. This is Amazon MWS-specific information that tells Amazon MWS the algorithm you used to form the string that is the basis of the signature. For Amazon MWS, this value is currently **SignatureVersion=2**.
- *Timestamp*—Each request must contain the timestamp of the request. Depending on the API function you're using, you can provide an expiration date and time for the request instead of or in addition to the timestamp.
- *Version*—The version of the API section being called.

To create the query string to be signed, do the following:

1. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when Content-Type is application/x-www-form-urlencoded).
2. URL encode the parameter name and values according to the following rules:
   - Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).
   - Percent encode all other characters with %XY, where X and Y are hex characters 0-9 and uppercase A-F.
   - Percent encode extended UTF-8 characters in the form %XY%ZA....
   - Percent encode the space character as %20 (and not +, as common encoding schemes do).
3. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII character 61), even if the parameter value is empty.
4. Separate the name-value pairs with an ampersand (&) (ASCII code 38).
5. Create the string to sign according to the following pseudo-grammar (the "\n" represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" +
    ValueOfHostHeaderInLowercase + "\n" +
    HTTPRequestURI + "\n" +
    CanonicalizedQueryString <from the preceding step>
```

   The HTTPRequestURI component is the HTTP absolute path component of the URI up to, but not including, the query string. If the HTTPRequestURI is empty, use a forward slash (/).

The following example is a query string for a `GetFeedSubmissionResult` request. Note that there are no spaces or line breaks in the sorted parameter string.

```
POST
mws.amazonservices.com
/
AWSAccessKeyId=AKIAFJPPO5KLY6G4XO7Q&Action=GetFeedSubmissionResult&FeedSubmiss
ionId=4321011681&Marketplace=ATVPDKIKX0DER&Merchant=A3F1LGRLCQDI4D&SignatureMe
thod=HmacSHA256&SignatureVersion=2&Timestamp=2011-02-04T23%3A08%3A19Z&Version=
2009-01-01
```

This is the string that you sign and then include in your URL request. The steps that show how to sign the query request string are in the section "Signing the query request."

Timestamps

The timestamp (or expiration time) you use in the request must be a *dateTime* object. A best practice is to provide the timestamp in the Coordinated Universal Time (Greenwich Mean Time) time zone format, such as "2009-03-03T18:12:22Z" or "2009-02-23T18:12:22.093-07:00". The *Timestamp* attribute must contain the client's machine time in ISO8601 format; requests with a timestamp significantly different (15 minutes) than the receiving machine's clock will be rejected to help prevent replay attacks. For more information about timestamps in XML, see *http://www.w3.org/TR/xmlschema-2/#dateTime*.

Every Amazon MWS response includes a Date header in its HTTP response that you can use to check whether your local machine's time matches our server's time, such as Date: Tue, 24 Mar 2009 20:34:28 GMT You can also load the Amazon MWS address *https://mws.amazonservices.com/* in any Web browser (no request is needed) to receive a response with the current Amazon MWS server time:

```
<?xml version="1.0"?>
<PingResponse>
  <Timestamp timestamp="2009-03-24T20:29:19:22Z"/>
</PingResponse>
```

Here are a few additional considerations when working with timestamps:

- In order to allow Amazon MWS to extend the content of the PingResponse, any software you write to parse the *Timestamp* should not break if sibling XML tags start to appear. Generally, you should ignore unknown tags in any XML Amazon MWS sends you, as per the web architectural principle in Section 5.2 of *http://www.w3.org/TR/webarch/*.
- If you specify a timestamp (instead of an expiration time), the request automatically expires 15 minutes after the timestamp. In other words, Amazon MWS does not process a request if the timestamp is more than 15 minutes earlier than the current time on Amazon MWS servers. Make sure your server's time is set correctly.
- If you are using .NET, you must not send overly specific timestamps, due to different interpretations of how extra time precision should be dropped. To avoid overly specific timestamps, manually construct *dateTime* objects with no more than millisecond precision.

Creating the User-Agent header

The User-Agent header is used to identify your application, its version number, and programming language. The User-Agent header information enables Amazon MWS to identify problems with particular

applications, application versions, and programming languages. The User-Agent header is not required, but it is a best practice to include one with each Amazon MWS request.

The Amazon MWS client libraries provide an easy-to-use method for passing the User-Agent header with every Amazon MWS request. When you initialize an Amazon MWS client library, you add the Application or Company Name and the Version Number. Other HTTP libraries also provide easy methods for constructing User-Agent headers, but if you have any difficulties creating the header, please request assistance from Amazon MWS.

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 example illustrates a minimally acceptable User-Agent header.

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

If you are a third-party application integrator, you might want to use a User-Agent header like the following.

```
      My Desktop Seller Tool/2.0 (Language=Java/1.6.0.11;
Platform=Windows/XP)
```

If you are a large seller who is integrating through your own IT department, you might want create a User-Agent header like the following, so Amazon MWS could help you troubleshoot using the Host attribute.

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

To specify additional attributes, use the format AttributeName=Value;, separating each name value pair with a semicolon. Should you wish 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. Amazon MWS will reject a User-Agent header if it is longer than 500 characters.

Signing the query request

The request signature is part of the authentication process for identifying and verifying who is sending a request. It is used as the value for the *Signature* parameter in the request URL you construct. Amazon MWS verifies both the identity of the sender and whether the sender is registered to use Amazon MWS. Authentication is performed using your access key Id to locate your Secret Key, which you use to create the request signature. If verification fails, the request is not processed. Note that if you are using one of the Amazon MWS client libraries to submit requests, you do not need to calculate your signature or time stamp.

1. Create a query request as described the the previous section called "Creating a query request." The following is an example of a query request:

```
POST
mws.amazonservices.com
/
AWSAccessKeyId=0PExampleR2&Action=SubmitFeed&FeedType=_POST_INVENTORY_AVAIL
ABILITY_DATA_&Marketplace=ATExampleER&Merchant=A1ExampleE6&SignatureMethod=
HmacSHA256&SignatureVersion=2&Timestamp=2009-08-20T01%3A10%3A27.607Z&Versio
n=2009-01-01
```

2. Calculate an RFC 2104-compliant HMAC with the string you just created, use your Secret Key as the key, and use HmacSHA256 or HmacSHA1 as the hash algorithm.
3. Convert the resulting value to base64.
4. Use the resulting value as the value of the *Signature* request parameter.

Creating the URL

The URL contains the following parts:

- https://
- The marketplace-specific web service endpoint you want to access
- Parameters that were included in the query request string, plus the calculated signature:
  - AWSAccessKeyId
  - Action
  - MarketplaceId list
  - SignatureMethod
  - SignatureVersion
  - Timestamp
  - Version
  - Signature
- User-Agent header

The following is an example of a complete request URL that you could submit: The actual request should not contain white space or line breaks.

```
https://mws.amazonservices.com/AWSAccessKeyId=AKIAFJPPO5KLY6G4XO7Q&Action=GetFeedSubmis
sionResult&FeedSubmissionId=4321011681&Marketplace=ATVPDKIKX0DER&Merchant=A3F1LGRLCQDI4
D&SignatureMethod=HmacSHA256&SignatureVersion=2&Timestamp=2011-02-04T23%3A08%3A19Z&Vers
ion=2009-01-0&Signature=WhateverTheSignatureWas1HTTP/1.1Host:mws.amazonservices.comX-Am
azon-User-Agent:AmazonJavascriptApp/1.0(Language=Javascript)Content-Type:text/xml
```

Working with Content-MD5 headers and MD5 checksums

MD5 is an algorithm for computing a 128-bit "digest" (or hash) of arbitrary-length data with a high degree of confidence that any alterations in the data will be reflected in alterations in the digest. You create a Content-MD5 header when you submit a feed. Amazon MWS calculates the MD5 checksum and compares it to the header you sent to ensure that the received feed has not been corrupted in transmission. The process is reversed when Amazon MWS sends a report; the Content-MD5 header is sent with the report and you calculate the MD5 checksum and compare it to the header Amazon sent to make sure the report you received has not been corrupted in transmission.

The basic process for sending a feed with a Content-MD5 header to Amazon MWS is as follows:

1. Store the feed to be submitted on disk before transmitting it to Amazon MWS.
2. Compute the Content-MD5 of the file and store it in a companion file.
3. Create a `SubmitFeed` request, pass in the stored Content-MD5, and attach the file contents in a stream.
4. Submit the request.

The following Java code sample illustrates how to compute the Content-MD5 header for a feed submitted to Amazon:

```
    /**
* Calculate content MD5 header values for feeds stored on disk.
*/
public static String computeContentMD5HeaderValue( 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;
}
```

The following Java code sample illustrates how to compute the MD5 checksum for a report that is downloaded:

```
  /**
  * Consume the stream and return its Base-64 encoded MD5 checksum.
  */
  public static String computeContentMD5Header(InputStream inputStream) {
  // Consume the stream to compute the MD5 as a side effect.
  DigestInputStream s;
  try {
  s = new DigestInputStream( inputStream,
  MessageDigest.getInstance("MD5"));
  // drain the buffer, as the digest is computed as a side-effect
  byte[] buffer = new byte[8192];
  while(s.read(buffer) > 0);
  return new String(

org.apache.commons.codec.binary.Base64.encodeBase64(s.getMessageDigest().digest()),

  "UTF-8");
  } catch (NoSuchAlgorithmException e) {
  throw new RuntimeException(e);
  } catch (IOException e) {
  throw new RuntimeException(e);
```

```
  }
  }
```

Response format

In response to an action request, Amazon MWS returns an XML file that contains the results of the request. If a request is successful, the response is returned with the data requested. The following example shows a successful response.

```
           <?xml version="1.0"?>
            <RequestReportResponse
xmlns="http://mws.amazonservices.com/doc/2009-01-01/">
              <RequestReportResult>
                <ReportRequestInfo>
                  <ReportRequestId>2291326454</ReportRequestId>
                  <ReportType>_GET_MERCHANT_LISTINGS_DATA_</ReportType>
                  <StartDate>2009-01-21T02:10:39+00:00</StartDate>
                  <EndDate>2009-02-13T02:10:39+00:00</EndDate>
                  <Scheduled>false</Scheduled>
                  <SubmittedDate>2009-02-20T02:10:39+00:00</SubmittedDate>


<ReportProcessingStatus>_SUBMITTED_</ReportProcessingStatus>
                </ReportRequestInfo>
              </RequestReportResult>
              <ResponseMetadata>
               <RequestId>88faca76-b600-46d2-b53c-0c8c4533e43a</RequestId>

              </ResponseMetadata>
            </RequestReportResponse>
```

If a request is unsuccessful, the main response element is **ErrorResponse**, irrespective of the action requested. This element contains one or more **Error** child elements. Each **Error** includes:

- An error code that identifies the type of error that occurred.
- A message code that describes the error condition in a human-readable form.
- An error type, identifying either the receiver or the sender as the originator of the error.

The following example shows an error response:

```
            <ErrorResponse
xmlns="http://mws.amazonservices.com/doc/2009-01-01/">
              <Error>
                <Type>Sender</Type>
                <Code>InvalidClientTokenId</Code>
                <Message>The AWS Access Key Id you provided does not exist
 in our records.</Message>

<Detail>com.amazonservices.mws.model.Error$Detail@17b6643</Detail>
              </Error>
              <RequestID>b7afc6c3-6f75-4707-bcf4-0475ad23162c</RequestID>
            </ErrorResponse>
```

Error Messages

If you have a problem with Amazon MWS, record the **RequestId** and **Timestamp** of your request. When you call for technical support, Amazon uses your **RequestId** and **Timestamp** to find the specific example of your issue. This information will help reduce the time required to resolve the issue.

The following table shows common Amazon MWS error code examples and possible solutions to the error.

Error Code Examples	Reason	How to Troubleshoot
"POST /?AWSAccessKeyId=AKIAJSTDR2444BJQ &AWSAccountId=458080 &Marketplace=ATVPDKIKX0DER &Merchant=AC28N11YUQ &SignatureVersion=2&Version=2009-01-01 &RequestId=2d093e-0408-4517 -9685-474d1a0a8e9e &CustomerId=A2AR6RWNQ &NamespaceUri=http %3A%2F%2Fmws.amazonservices.com %2Fdoc%2F2009-01-01%2F &ServiceName=MarketplaceWebService &Action=GetReportList &**ErrorCode=ServiceUnavailable** &ErrorFault=Receiver HTTP/0.0" 503 296 "-" "UST/1.0 (Language=PHP/5.2.14; MWSClientVersion=2009-07-02; Platform=Linux infong 2.4 #1 SMP Wed Nov 4 21:12:12 UTC 2009 i686 GNU/Linux/Linux infong 2.4 #1 SMP Wed Nov 4 21:12:12 UTC 2009 i686 GNU/Linux/Linux infong 2.4 #1 SMP Wed Nov 4 21:12:12 UTC 2009 i686 GNU/Linux)"	This 503 error indicates that the Amazon MWS service is unavailable. When you use the client library, the response is parsed and a MarketplacewebService exception with all data included is thrown.	Retry the request.
"POST /?AWSAccessKeyId=AKIVUUN IIMFTA &AWSAccountId=7948 &Marketplace=ATVPDKIKX0DER &Merchant=ASH1H4EF &SignatureVersion=2 &Version=2009-01-01 &RequestId=260-0116-41fa-91d0 -7bc98359c694 &CustomerId=ASH14EF &NamespaceUri=http%3A%2F %2Fmws.amazonservices.com %2Fdoc%2F2009-01-01%2F &ServiceName=MarketplaceWebService &Action=GetReportRequestList &**ErrorCode=RequestThrottled** &ErrorFault=Sender HTTP/0.0" 503 309 "-" "null"	This 503 error indicates that your request is being throttled. When you use the client library, the response is parsed and a MarketplacewebService exception with all data included is thrown.	Check the throttling limit for the type of request you are submitting. Set up retry logic to resend the request when the appropriate amount of time has passed to prevent throttling.
javax.net.ssl.SSLException: java.lang.RuntimeException: Unexpected error: java.security. InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty	The client tried to communicate with the Amazon MWS endpoint, but it could not verify our SSL certificate or it could not find the certificate store on the client machine to use for verification.	If you get this exception, you need to add Amazon MWS certificates to your client trust store. Consult the Java documentation regarding setting up and configuring Trust Stores in Java.

Error Code Examples	Reason	How to Troubleshoot
<ErrorResponse xmlns="http://mws.amazonservices.com/doc/2009-01-01/"> <Error><Type>Sender</Type> <Code>UserAgentHeaderMalformed</Code> <Message>Problem with required MWS User-Agent header (e.g. "MyAppName/build123 (Language=Java/1.2)"): Encountered " <EOF>" at column 116. Was expecting: "=" ...</Message><Detail/> </Error><RequestID>21f197f6-24b7-4b7b-94fc-55fa34056d34 </RequestID></ErrorResponse>	The User-Agent header sent with the request was not in a valid format.	Build the User-Agent header using code from the Amazon MWS client library or see the documentation for an acceptable format for the User-Agent header.
the Content-MD5 HTTP header you passed for your feed (1B2M2Y8AsgTpgAmY7PhCfg==) did not match the Content-MD5 we calculated for your feed (3cldK7kqMxK6orwvXXdzSQ==)	The Content-MD5 value **1B2M2Y8AsgTpgAmY7PhCfg==** corresponds to the empty string. The MD5 provider instance used to calculate the Content-MD5 is not able to read any bytes from the stream.	A possible solution is to export the document as a string and construct the MemoryStream with this string's bytes. MemoryStream stream = new MemoryStream(new UTF8Encoding() .GetBytes(xmlDocument.ToString())); Then request.ContentMD5 = MarketplaceWebServiceClient .CalculateContentMD5(stream)

Error Messages

The following table describes Amazon MWS error messages. Additional errors, which might be returned due to problems with your feeds, are detailed in the Seller Central Help topics.

Error Message	Description
AccessDenied	Client tried connecting to MWS through HTTP rather than HTTPS.
AccessToFeedProcessingResultDenied	Insufficient privileges to access the feed processing result.
AccessToReportDenied	Insufficient privileges to access the requested report.
ContentMD5Missing	The Content-MD5 header value was missing.
ContentMD5DoesNotMatch	The calculated MD5 hash value does not match the provided Content-MD5 value.
FeedCanceled	Returned for a request for a processing report of a canceled feed.
FeedProcessingResultNoLongerAvailable	The feed processing result is no longer available for download.
FeedProcessingResultNotReady	Processing report not yet generated.
InputDataError	Feed content contained errors.
InternalError	Unspecified server error occurred.
InvalidFeedSubmissionId	Provided Feed Submission Id was invalid.
InvalidFeedType	Submitted Feed Type was invalid.
InvalidParameterValue	Provided query parameter was invalid. For example, the format of the Timestamp parameter was malformed.

Error Message	Description
InvalidQueryParameter	Superfluous parameter submitted.
InvalidReportId	Provided Report Id was invalid.
InvalidReportType	Submitted Report Type was invalid.
InvalidScheduleFrequency	Submitted schedule frequency was invalid.
MissingClientTokenId	The merchant Id parameter was empty or missing.
MissingParameter	Required parameter was missing from the query.
ReportNoLongerAvailable	The specified report is no longer available for download.
ReportNotReady	Report not yet generated.
SignatureDoesNotMatch	The provided request signature does not match the server's calculated signature value.
UserAgentHeaderLanguageAttributeMissing	The User-Agent header Language attribute was missing.
UserAgentHeaderMalformed	The User-Agent value did not comply with the expected format.
UserAgentHeaderMaximumLengthExceeded	The User-Agent value exceeded 500 characters.
UserAgentHeaderMissing	The User-Agent header value was missing.

Related Resources

For details about the schemas for the various feed types, see the guide *Selling on Amazon Guide to XML*

Amazon maintains a community-based forum for developers to discuss technical questions related to Amazon MWS.

http://www.amazonsellercommunity.com/forums/forum.jspa?forumID=43

The primary web page for information about Amazon Marketplace Web Service, including links to the developer documentation and client libraries:

- **CA:** *http://developer.amazonservices.ca*
- **DE:** *http://developer.amazonservices.de*
- **FR:** *http://developer.amazonservices.fr*
- **JP:** *http://developer.amazonservices.jp*
- **UK:** *http://developer.amazonservices.co.uk*
- **US:** *http://developer.amazonservices.com*