amazonpayments™

Managing Orders with MWS APIs

A guide for merchants who want to manage their orders using MWS APIs.

Table of Contents

1 Post-order Management with MWS APIs

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

Using MWS, you can create applications that schedule and download orders for fulfillment, confirm shipment, and cancel and refund orders. These API operations are accessible using a REST like interface.

MWS provides the following features with respect to Checkout by Amazon:

- Order management— you can download order information, obtain payment data, acknowledge orders, cancel orders, confirm shipments, and adjust/refund orders.
- Reports management— you can request a variety of reports as well as query the status of these reports, and then download them.

2 Getting Started with MWS APIs

Follow the following steps to manage orders using MWS APIs.

2.1 Step 1: Register for MWS

If you do not have an account with Amazon Marketplace Web Services (MWS), you should sign up. There are three options available to you when you sign up to use MWS.

1. **You want to access your own Checkout by Amazon seller account with MWS**
   When you sign up to use MWS for your own Amazon seller account, MWS will assign you a developer account identifier. When you make MWS requests, you'll use the developer account credentials associated with your developer account, plus the Merchant ID and the Marketplace ID for your seller account. These credentials appear on the final page of your registration.
2. **You want to use an application to access your Checkout by Amazon seller account with MWS**
   If you want to use an application to access your Amazon seller account, simply choose that option during the MWS registration process. When requested, you enter the developer account identifier for the application. On the last page of the MWS sign-up process, you will see your Merchant ID and Marketplace ID. Follow your application provider's instructions to enter these credentials into the application.
3. **You want to give a developer access to your Checkout by Amazon seller account with MWS**
   You can authorize a third party developer to access your account with MWS. First, the developer must give you his or her MWS developer account identifier. You enter this developer account identifier to authorize the third-party developer to access your account. On the last page of the MWS sign-up process, you will see your Merchant ID and Marketplace ID, which you will then give to your developer so MWS requests can be made on your behalf.

Click here to sign up with MWS.

After registration you will receive your MWS Access Key ID and MWS Secret Key. Please make note of these keys and store them in a secure place. You will use these keys to sign your requests to MWS.

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

2.2 Step 2: Develop MWS Application Using Our Client Libraries

After you register for MWS, you must develop your MWS application using MWS client libraries. You can use this MWS client library to create requests to perform various order management operations.

Following is an example in PHP. You can follow similar steps in other programming languages.

2.2.1 Download the MWS Client Library

Download the MWS PHP client library here . Extract it into a working folder on a computer with PHP 5.2.8 or later and cURL 7.18.0 or later installed.

2.2.2 Configure the Client library with your details

Open the file .config.inc.php at the location <MWS_DIR>/src/MarketplaceWebService/Samples/. In this file, you must configure the following:

2.2.2.1 *MWS access credentials*

You must configure the MWS access credentials you received after registering for MWS here.

In the downloaded client library, replace the "<Your Access Key ID>" parameter within the single quotes with your MWS Access Key ID. Similarly, replace "<Your Secret Access Key>" within the single quotes with your MWS Secret Key. Note that the Access Key ID and Secret Key should be enclosed within single quotes (').

2.2.2.2 *User Agent Header*

All MWS requests must contain a User-Agent header. The application name and version configured here are used in creating this value. Replace "<Your Application Name>" within the single quotes with your own self defined value. This helps you to easily identify your MWS requests. Replace "<Your Application Version or Build Number>" within the single quotes with your version or build number. Note that the Application name and Version should be enclosed within single quotes (').

2.2.2.3 *Merchant and Marketplace ID*

All MWS requests must contain the seller's Amazon Merchant ID and Marketplace ID. Replace "<Your Merchant ID>" within the single quotes with your Amazon Merchant ID, and then replace "<Your Marketplace ID>" within the single quotes with your Amazon Marketplace ID. Note that the Merchant and Marketplace IDs should be enclosed within single quotes (').

After you configure the values and save the the .config.inc.php file, you can start Order Management using the Sample files provided in the folder as explained in the following steps.

2.3 Important Information regarding MWS

Before proceeding to use various MWS API operations, please read the following information.

2.3.1 Amazon Marketplace Web Service endpoints

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

MWS Endpoint	MWS Website
https://mws.amazonservices.com	http://developer.amazonservices.com

2.3.2 Report Formats

Whether a seller receives reports in XML or flat-file format is determined by the seller account configuration. The report type can be set in Seller Central > Settings > Checkout Pipeline Settigns > Order Reports

2.3.3 Feed Formats

You can Acknowledge, Cancel, Confirm Shipment, Refund, or Adjust orders by submitting feeds to MWS. There are two types of feeds: flat text files, which you can build using a spreadsheet application, and XML documents. The different feed formats are explained later in Processing Orders through Feeds, below.

2.3.4 SSL requirements for inbound connections

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

2.3.5 Using the Content-MD5 Header with feed operations

The Content-MD5 header is used as a message-integrity check to verify that the decoded data received is the same data that was sent. When you submit a feed using the SubmitFeed operation, or when you receive a feed report using the GetFeedSubmissionResult operation, you must use a Content-MD5 header. Please see below in this document for more information

When you use the SubmitFeed operation, you must create a Content-MD5 header containing the MD5 hash of the HTTP entity body and include it in your request. This a MWS to compare the MD5 hash you create with the MD5 hash it creates when it receives the feed. This process lets MWS determine if the feed submitted for processing is identical to the feed that was received. This process prevents corrupted descriptive data or pricing product data from appearing on Amazon.

When you use the GetFeedSubmissionResult operation, you must calculate an MD5 hash for the received feeds report and compare that with the Content-MD5 header that is included in the response. If the two match, the report was not corrupted in transmission.

The MWS client libraries provide an easy way to pass in the Content-MD5 header with every MWS request, as long as you send data that has been stored on disk and an MD5 hash has been created for the data. Learn more about working with the Content-MD5 header.

2.3.6 Throttling: Limits to how often you can submit requests

To use MWS successfully, you must understand *throttling*. Throttling is the process of limiting the number of requests you can submit in a given amount of time. A request can include submitting an inventory feed or sending an order report request. Throttling protects the web service from being overwhelmed with requests and make sure all authorized developers have access to the web service.

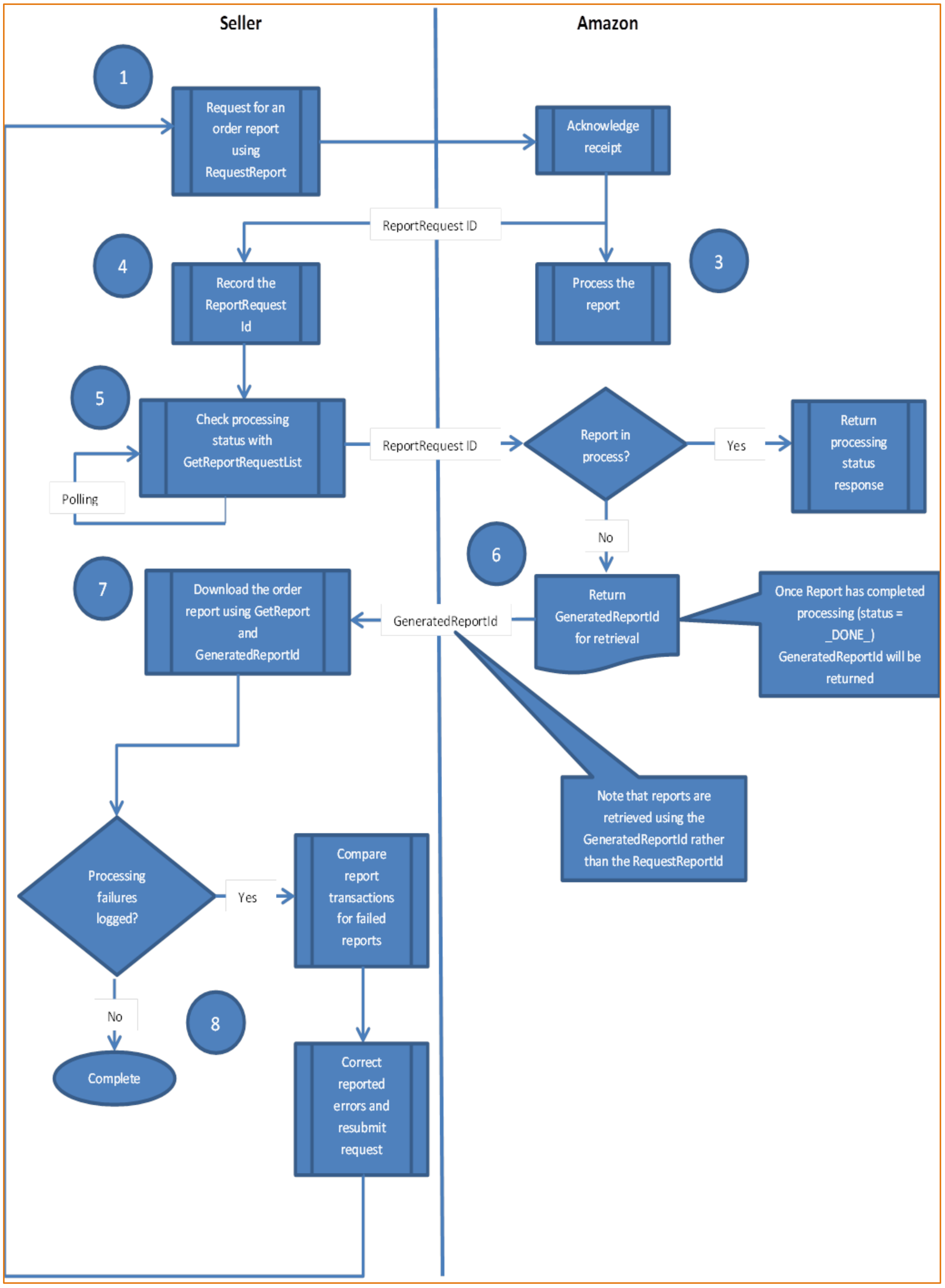
Learn more about throttling and MWS.

2.4 Step 3: Get the Order Report to View Order Details

If your website generates more than 25 Checkout by Amazon orders a day, you might find it more efficient to use Orders Reports to view and manage your orders. An Orders Report is a XML file or a tab-delimited file which provides a summary of the orders you received, along with customer and shipping information you need to fulfill those orders.

To get an Orders Report, you can either request it when you want it, or you can schedule it. Once the report is created (either by request or as a scheduled report), you can download it, open it in an application such as Microsoft Excel (if it's a tab-delimited file), and then manage your orders.

The entire process of the Getting Order Reports process is represented in the following flow chart:


Seller
Amazon
1
Request for an order report using RequestReport
Acknowledge receipt
ReportRequest ID
4
Record the ReportRequest Id
Process the report
3
5
Check processing status with GetReportRequestList
Polling
ReportRequest ID
Report in process?
Yes
Return processing status response
No
6
7
Download the order report using GetReport and GeneratedReportId
GeneratedReportId
Return GeneratedReportId for retrieval
Once Report has completed processing (status = _DONE_) GeneratedReportId will be returned
Note that reports are retrieved using the GeneratedReportId rather than the RequestReportId
Processing failures logged?
Yes
Compare report transactions for failed reports
No
8
Complete
Correct reported errors and resubmit request

1. **Request report** – Seller requests an order report (such as an Order Fulfillment feed) and posts it using RequestReport.
2. **Amazon Payments acknowledges the document** – Amazon Payments returns an acknowledgement receipt that the request was received and provides a ReportRequest ID.
3. **Amazon processes the request** – Amazon processes the request and summarizes the result.
4. **Seller records the ReportRequest ID** – The ReportRequest ID is the ID for the transfer and receipt of the report. This is not the same as the GeneratedReport ID returned when the requested report has completed processing.
5. **Seller checks to see if there is a processing report ready to download** – Seller polls the report processing status by using the GetReportRequestList API and the ReportRequest ID returned in Step 2, above.
6. **Amazon communicates the ReportRequest ID status** – If the report is not fully processed, the seller receives a "pending" status. When the report is completely processed, then the processing status "Done" is returned along with the GeneratedReport ID for retrieving the processing results.
7. **Get the processing report** – Seller uses the GetReport API to retrieve the report using the GeneratedReport ID, returned in Step 6, above.
8. **Seller processing** – If there are errors in the processing report, seller corrects the problems found within the processing report, and then re-sends the request report (Step 1). If there are no errors, processing is complete.

The major steps in Order Reports are as follows:

2.4.1 Request a report

First, an order report is requested from MWS using the RequestReport API call. To create a RequestReport API call, you can use the RequestReportSample.php file in the samples provided along with the MWS client library.

Learn more about the RequestReport API operation.

2.4.2 Schedule a report (optional)

Instead of requesting a report each time you want one, you can schedule order report requests using the ManageReportSchedule API call so the reports are returned periodically. This is particularly useful for downloading a report at a set interval, and removes the need for requesting a report, waiting for it to process, and then downloading it. Once you have scheduled a report for a set interval, you can directly download the report after the set interval, as Amazon processes the report and keeps it ready for your download at the end of the set interval.

Learn more about the ManageReportSchedule API.

2.4.3 Check if the processing report is ready for download

Check if the report you requested has been processed using the GetReportRequestList API call. This API call returns all the requests whose processing status is set to "_DONE_", meaning that the RequestReport API call you submitted earlier is processed and the report is ready for download.

Learn more about the GetReportRequestList API.

2.4.4 Download order report

After the request is processed and is ready for download, you can download it using the GetReportList and GetReport API calls. The GetReportList is used to get the unique ReportId associated with the process request. You then can use this ReportId with the GetReport API call to download the report.

Learn more about downloading the processed report.

2.4.5 Input Orders into Your Order Management System

Now that the Order Report is downloaded, you can process the report and upload data into your order management system. If the report is a tab-delimited file, then you can open it in an application such as Microsoft Excel for analysis. The first line of the tab-delimited file is the header row, so it is easier for you to feed the values of the orders into your order management system.

2.5 Step 4: Processing Orders Through Feeds

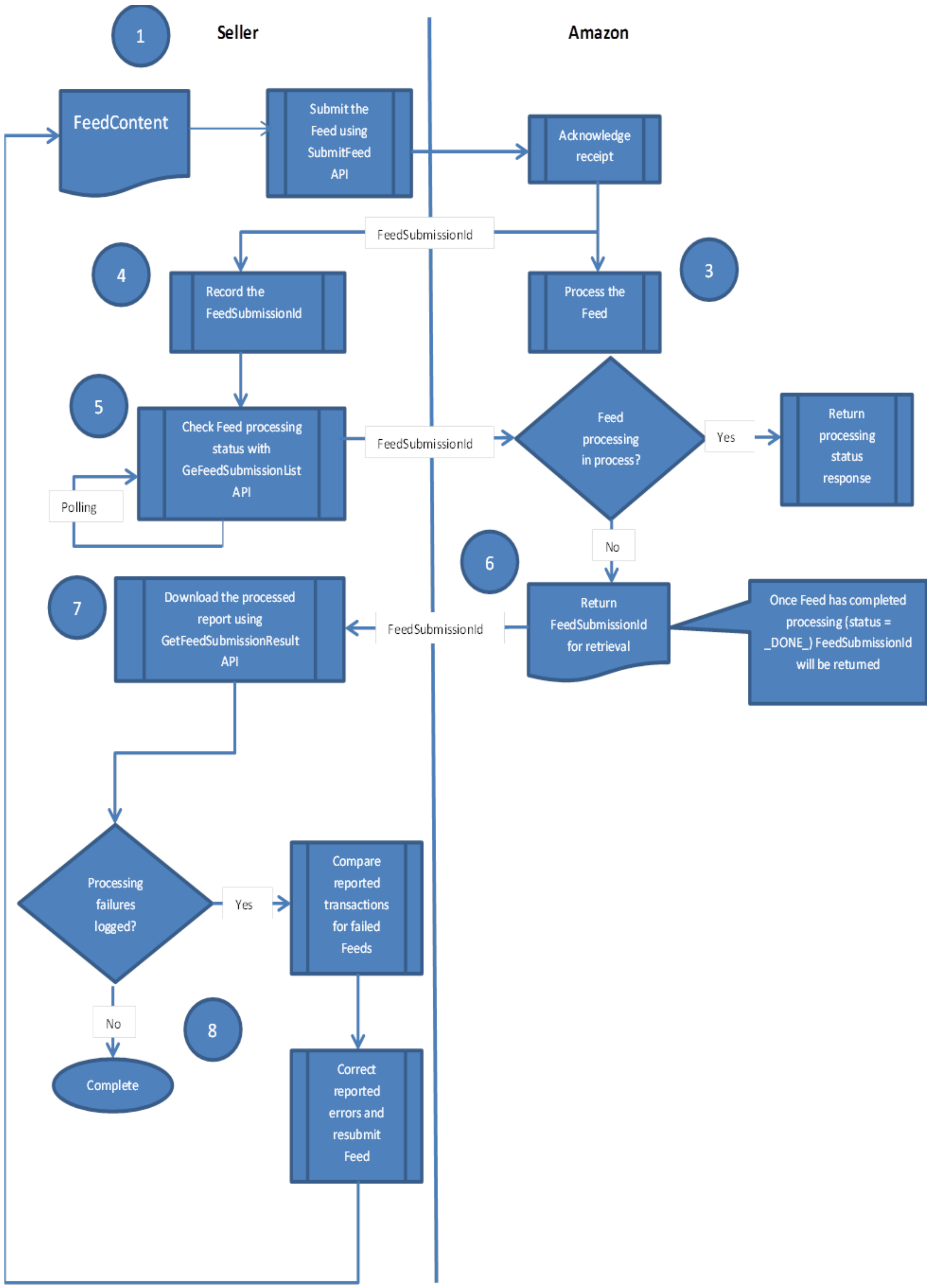
Once you have the orders in you Order Management system, you can then Acknowledge, Cancel, Confirm Shipment, Refund, or Adjust them by submitting feeds to MWS.

The process of processing orders through feeds is represented in the following flow chart:

1. **Feed Content** — Seller constructs a feed for the respective order management he wants to submit and submits the feed using SubmitFeed API.
2. **Amazon Payments acknowledges the document** — Amazon Payments returns an acknowledgement receipt that the request was received and provides a FeedSubmissionId.
3. **Amazon processes the request** — Amazon processes the feed request and summarizes the result.
4. **Seller records the FeedSubmissionId** — The FeedSubmissionId is the ID for the transfer and receipt of the processed feed report. You can use this to download the processed feed report.
5. **Seller checks to see if there is a processing report ready to download** — Seller polls the report processing status using the GetFeedSubmissionList API and the FeedSubmissionId returned in Step 2, above.
6. **Amazon Communicates FeedSubmissionId status** — If the report is not processed, the seller receives a "_PROCESSING_" status. When the report is completely processed, then the

processing status "_DONE_" is returned along with the FeedSubmissionId for retrieving the processing results.

7. **Get the processed feed report** — Seller uses the GetFeedSubmissionResult API to retrieve the processing report using the FeedSubmissionId, returned in Step 6, above.
8. **Seller processing** — If there are errors in the processing report, seller corrects the problems found within the processing report in the feed, and then resubmits the feed (Step 1). If there are no errors, processing is complete.


1
Seller
Amazon
FeedContent
Submit the Feed using SubmitFeed API
Acknowledge receipt
FeedSubmissionId
4
Record the FeedSubmissionId
Process the Feed
3
5
Check Feed processing status with GeFeedSubmissionList API
Polling
FeedSubmissionId
Feed processing in process?
Yes
Return processing status response
No
6
7
Download the processed report using GetFeedSubmissionResult API
FeedSubmissionId
Return FeedSubmissionId for retrieval
Once Feed has completed processing (status = _DONE_) FeedSubmissionId will be returned
Processing failures logged?
Yes
Compare reported transactions for failed Feeds
No
8
Complete
Correct reported errors and resubmit Feed

The steps for order management through feeds are as follows:

2.5.1 Construct an Order feed

This section provides the information required to construct various feeds for managing orders. The basic order management functions and their feeds are:

2.5.1.1 Acknowledge or cancel the order (Order Acknowledgement feed)

The Order Acknowledgement feed allows a seller to acknowledge the success or failure of an order receipt. This is simply the acknowledgement of receipt, and **not** a communication of a fulfillment problem. The acknowledgement feed also permits you to associate their own order ID and order item IDs for future reference within other feeds. Also, you can cancel an order by sending "Failure" as the StatusCode in the Order Acknowledgement feed.

FeedType enumeration used for these feeds are as follows:

- For XML - _POST_ORDER_ACKNOWLEDGEMENT_DATA_
- For Flat File - _POST_FLAT_FILE_ORDER_ACKNOWLEDGEMENT_DATA_

Learn more about the Order Acknowledgement feed and examples.

2.5.1.2 Confirm order shipment (Order Fulfillment feed)

The Order Fulfillment feed allows a seller to update Amazon with information about an order's fulfillment status. Data provided by the seller is used to update the Amazon Payments order information that appears when the buyer checks his account.

This feed is used to confirm to Amazon the shipment of orders as well as to provide the carrier/shipment details that are passed on to the buyer by Amazon.

FeedType enumeration used for these feeds are as follows:

- For XML - _POST_ORDER_FULFILLMENT_DATA_
- For Flat File - _POST_FLAT_FILE_FULFILLMENT_DATA_

Learn more about Order Fulfillment Feed and examples.

2.5.1.3 Refund an order (Order Adjustment feed)

The Order Adjustment feed accepts data from a seller about a refund or adjustment to existing orders. Orders are identified either by the Amazon Payments order ID or by the seller's order ID, if it was previously provided in the Order Acknowledgement feed. Similarly, items within the order can be identified either by the Amazon Payments order item code or by the seller's order item ID, if it was previously provided. Sellers must provide a reason for the adjustment, and the amounts to be adjusted, broken out by price component (principle, shipping, tax, and so on).

FeedType enumeration used for these feeds are as follows:

- For XML - _POST_PAYMENT_ADJUSTMENT_DATA_

- For Flat File - _POST_FLAT_FILE_PAYMENT_ADJUSTMENT_DATA_

Learn more about Order Adjustment Feed and examples.

2.5.1.4 *Submit a feed*

The Order Fulfillment feed allows a seller to update Amazon with information about an order's fulfillment status. Data provided by the seller is used to update the Amazon Payments order information that appears when the buyer checks his account. This feed is used to confirm to Amazon the shipment of orders as well as to provide the carrier/shipment details that are passed on to the buyer by Amazon.

FeedType enumeration used for these feeds are as follows:

- For XML - _POST_ORDER_FULFILLMENT_DATA_
- For Flat File - _POST_FLAT_FILE_FULFILLMENT_DATA_

Learn more about the Order Fulfillment feed and examples.

2.5.1.5 *Refund an order (Order Adjustment feed)*

The Order Adjustment feed accepts data from a seller about a refund or adjustment to existing orders. Orders are identified either by the Amazon Payments order ID or by the seller's order ID, if it was previously provided in the Order Acknowledgement feed. Similarly, items within the order can be identified either by the Amazon Payments order item code or by the seller's order item ID, if it was previously provided. Sellers must provide a reason for the adjustment, and the amounts to be adjusted, broken out by price component (principle, shipping, tax, and so on).

FeedType enumeration used for these feeds are as follows:

- For XML - _POST_PAYMENT_ADJUSTMENT_DATA_
- For Flat File - _POST_FLAT_FILE_PAYMENT_ADJUSTMENT_DATA_

Learn more about Order Adjustment feed and examples.

2.6 Submit a Feed

You can submit an XML or flat file using the SubmitFeed operation along with an encrypted header and all required metadata, including a value from the FeedType enumeration.

As with all submissions to MWS, you must also include authentication information. The SubmitFeed operation returns a FeedProcessingId, which you can use to periodically check the status of the feed using the GetFeedSubmissionList operation.

Learn more about SubmitFeed operation

2.7 Check Processing Status of Your Feed

You can check the status of your feed by using the GetFeedSubmissionList operation. If MWS is still processing a request; the FeedProcessingStatus element of the GetFeedSubmissionList operation returns a status of _IN_PROGRESS_. If the processing is complete, a status of _DONE_ is returned.

Learn more about the GetFeedSubmissionList operation.

2.8 View the Status of Your Feed Submission

When the feed processing is complete, that is, marked as _DONE_ in the GetFeedSubmissionList, you can use the GetFeedSubmissionResult operation to receive a processing report describing which records in the feed were successful and which records generated errors. Note that you have to set up a stream that MWS uses to write out the report when you submit the GetFeedSubmissionResult operation. Use the MWS Feeds API section client library code for the GetFeedSubmissionResult operation to create the stream.

After you receive the processing report, you can analyze it, correct any errors in the file or transmission, and then resubmit the feed using the SubmitFeed operation. Repeat the process until there are no errors in the processing report. When the processing report is error-free, the transmission is complete.

Learn more about the GetFeedSubmissionResult operation.

2.9 Using the Instant Order Processing Notification (IOPN) API

MWS APIs do not provide real-time order notification. You must poll MWS APIs to get order-related information. If you want instant updates and then automatically process orders using your order management system (OMS), you can use the Instant Order Processing Notification (IOPN) API. With the Instant Order Processing Notification API, you can be instantly notified of new orders; you can reserve your inventory (remove it from availability to be sold) as soon as you get an Order Ready-to-Ship notification; and you can release your inventory (make it available to be sold) when you receive an Order Canceled notification.

Learn more about Instant Order Processing Notifications.

2.10 What's New in This Document

Ver	Date	Changes
1.0	2011-05-31	Initial Release

3 Appendix A: 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. MWS calculates the MD5 checksum and compares it to the header you sent to make sure that the received feed was not corrupted during transmission. The process is reversed when 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 was not corrupted in transmission.

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

1. Store the feed to be submitted on your computer before transmitting it to 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 PHP code sample illustrates how to compute the Content-MD5 header for a feed submitted to MWS:

```
/**
  * Get the Base64-encoded md5 value of $data. If $data is a memory or temp file stream,
  * this method dumps the contents into a string before calculating the md5. Hence, this
  * method shouldn't be used for large memory streams.
  *
  * @param $data
  * @return unknown_type
  */
  private function getContentMd5($data) {
     $md5Hash = null;
     if (is_string($data)) {
     $md5Hash = md5($data, true);
     } else if (is_resource($data)) {
     // Assume $data is a stream.
     $streamMetadata = stream_get_meta_data($data);
     if ($streamMetadata['stream_type'] === 'MEMORY' || $streamMetadata['stream_type'] ===
'TEMP') {
        $md5Hash = md5(stream_get_contents($data), true);
     } else {
        $md5Hash = md5_file($streamMetadata['uri'], true);
     }
     }
     return Base64_encode($md5Hash);
  }
```

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

```
/**
/**
```

```
* Compares the received Content-MD5 Hash value from the response with a locally calculated
* value based on the contents of the response body. If the received hash
* value doesn't match
* the locally calculated hash value, an exception is raised.

* $streamHandle that needs to be passes to this function is a stream
* opened on the downloaded response for what the Md5 needs to computed
* and compared with the Md5 received in the Service Response
* @param $receivedMd5Hash

* @param $streamHandle
* @return unknown_type
*/
private function verifyContentMd5($receivedMd5Hash, $streamHandle) {

rewind($streamHandle);

$expectedMd5Hash = $thare as follows:>getContentMd5($streamHandle);

rewind($streamHandle);

if (!($receivedMd5Hash === $expectedMd5Hash)) {

 throw new Exception(

    array(

      'Message' => 'Received Content-MD5 = ' . $receivedMd5Hash . ' but expected ' .
       $expectedMd5Hash,

      'ErrorCode' => 'ContentMD5DoesNotMatch'));
}
}
```

4 Appendix B: Throttling and MWS

To use MWS successfully, you must 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 makes sure all authorized developers have access to the web service.

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 a rate, the water will exceed the capacity of the bucket.

To apply this analogy to 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 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 MWS throttling are:

1. *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.
2. *Restore rate* (also called the *recovery rate*) - The rate at which your request quota increases over time, up to the maximum request quota.
3. *Request quota maximum* (also called the *burst rate*) - The maximum size that the request quota can reach.

While most 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 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 will then need to resubmit 10 feed requests once the request quota is restored. As the restore rate is one request every two minutes, it will 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 can automate your process to submit feed requests incrementally.

For example, you can submit 10 feed requests (from your original 25 feeds), and the request quota will still have 5 requests left over. You can then wait 10 minutes, and the restore rate will have increased the request quota to 10 (one request every 2 minutes for 10 minutes gives you 5 new requests).You can then submit 10 more feed requests. For the remaining 5 feed requests, you can wait 10 more minutes and then submit them. If all things go well, you will 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 can slow down the number of requests you make and resubmit requests that initially failed.

4.1 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 42 minutes after the hour.
- Take advantage of times during the day when traffic is likely to be low on MWS, such as early evening or early morning hours.

5 Appendix C: RequestReport API

5.1 Description

The RequestReport operation creates a report request. MWS processes the report request, and when the report is completed, sets the status of the report request to _DONE_. Reports are retained for 90 days. The RequestReport operation has a maximum request quota of 15 and a restore rate of 1 request every 2 minutes. Requests are included within the overall limit of 1,000 requests per seller account per hour.

5.2 ReportType Enumerations for Checkout by Amazon MWS

Report Title	Enumeration Value	Description
Unshipped Orders Report	_GET_FLAT_FILE_ACTIONABLE_ORDER_DATA_	Tab-delimited flat file report that contains only orders that are not confirmed as shipped.
Flat File Order Report	_GET_FLAT_FILE_ORDER_REPORT_DATA_	Tab-delimited flat file order report. For Seller Central sellers only.
Scheduled XML Order Report	_GET_ORDERS_DATA_	Scheduled XML order report. For Seller Central sellers only.

5.3 Request Parameters

Name	Description	Required	Valid Values
ReportType	A value of the ReportType enumeration that indicates the type of report to request. Type: xs:string	Yes	A ReportType enumeration value
StartDate	The start of a date range used for selecting the data to report. Type: xs:datetime	No	Default: Now
EndDate	The end of a date range used for selecting the data to report. Type: xs: datetime	No	Default: Now

5.4 Response Elements

Name	Description
ReportRequestId	A unique identifier for the report request. Type: xs:string
ReportType	The ReportType enumeration value requested.
StartDate	The start of a date range used for selecting the data to report. Type: xs:datetime
EndDate	The end of a date range used for selecting the data to report. Type: xs:datetime
Scheduled	A Boolean value that indicates if a report is scheduled. The value is true if the report

Name	Description
	was scheduled; otherwise it is false. Type: xs:boolean
SubmittedDate	The date when the report was submitted. Type: xs:datetime
ReportProcessingStatus	The processing status of the report.

5.5 Example Using the MWS Client Library

Following is an example using the PHP code library. You can follow similar steps with our C# or Java code libraries to request a report. You can request reports using the RequestReportSample.php file located at <MWS_DIR>/src/MarketplaceWebService/Samples/. Follow the steps given below to request a report:

5.5.1 Configure the RequestReportSample.php for your marketplace

Uncomment the $serviceUrl line corresponding to the country you are registered to sell in, so that the correct endpoint is used for submitting your report request.

For example: U.S. sellers should uncomment the line following the line "//United States:" in the file (that is, $serviceUrl = "https://mws.amazonservices.com")

5.5.2 Configure the parameters for your RequestReport

The type of the report should be configured in the file. This can be done by replacing the following line in the file with the appropriate ReportType enumeration from the above table.

For example, for Orders that need to be shipped:

```
    $request->setReportType('_GET_FLAT_FILE_ACTIONABLE_ORDER_DATA_');
```

The rest of the parameters such as MARKETPLACE_ID, MERCHANT_ID etc. are retrieved from the .config.inc.php file which you configured earlier.

5.5.3 Submit the RequestReport

As all the required changes were made to the RequestReportSample.php file for requesting a report, you can save it now and run it using the command:

```
php RequestReportSample.php
```

5.5.4 Process the service response

If everything is configured correctly, you will see an output which ends with the following service response:

```
Service Response
=============================================================================
        RequestReportResponse
```

```
RequestReportResult
    ReportRequestInfo
        ReportRequestId
            4022565420
        ReportType
            _GET_FLAT_FILE_ACTIONABLE_ORDER_DATA_
        StartDate
            2011-05-13T10:29:47Z
        EndDate
            2011-05-13T10:29:47Z
        SubmittedDate
            2011-05-13T10:29:47Z
        ReportProcessingStatus
            _SUBMITTED_
ResponseMetadata
    RequestId
        9953983f-12f5-4e89-b341-42ba3b6fa71a
```

Please note the ReportRequestId given in the service response. In this case it is "4022565420" as you can see in the response above. Also note that ReportProcessingStatus is "_SUBMITTED_"; otherwise this indicates an error in your RequestReport API call.

6 Appendix D: ManageReportSchedule

6.1 Description

The ManageReportSchedule operation creates, updates, or deletes a report request schedule for a particular report type. Currently, only order reports can be scheduled. By using a combination of the ReportType and Schedule values, MWS determines which action you want to perform.

1. If no combination of ReportType and Schedule exists, then a new report request schedule is created.
2. If the ReportType is already scheduled but with a different Schedule value, then the report request schedule is updated to use the new Schedule value.
3. If you pass in a ReportType and set the Schedule value to _NEVER_ in the request, the report request schedule for that ReportType is deleted.

The ManageReportSchedule operation has a maximum request quota of 10 and a restore rate of 1 request per minute. Requests are included within the overall limit of 1,000 requests per seller account per hour.

The following table shows the ReportType enumeration values you can use with the ManageReportSchedule operation.

ReportType	Enumeration Value Report Title
_GET_ORDERS_DATA_	Scheduled XML Order Report
_GET_FLAT_FILE_ORDERS_DATA_	Scheduled Flat File Order Report

6.2 Schedule Enumeration

Schedule Description	Enumeration Value
Every 15 minutes	_15_MINUTES_
Every 30 minutes	_30_MINUTES_
Every hour	_1_HOUR_
Every 2 hours	_2_HOURS_
Every 4 hours	_4_HOURS_
Every 8 hours	_8_HOURS_
Every 12 hours	_12_HOURS_
Every day	_1_DAY_
Every 2 days	_2_DAYS_
Every 3 days	_72_HOURS_
Every 7 days	_7_DAYS_
Every 14 days	_14_DAYS_
Every 15 days	_15_DAYS_
Every 30 days	_30_DAYS_
Delete a previously created report request schedule	_NEVER_

6.3 Request Parameters

Name	Description	Required	Valid values
ReportType	A value of the ReportType enumeration that indicates the type of report to request. Type: xs:string	Yes	A ReportType enumeration value
Schedule	A value of the Schedule enumeration that indicates how often a report request should be created. Type: xs:string	Yes	A valid Schedule enumeration value Default: None
ScheduledDate	The date when the next report request is scheduled to be submitted. Type: xs:datetime	No	Default: Now value can be no more than 366 days in the future.

6.4 Response Elements

Name	Description
Count	A non-negative integer that represents the total number of report requests. Type: tns: nonNegativeInteger
ReportType	The ReportType enumeration value requested.
Schedule	A value of the Schedule enumeration that indicates how often a report request should be created. Type: xs:string
ScheduledDate	The date when the next report request is scheduled to be submitted. Type: xs:datetime

6.5 Example Using the MWS Client Library

Following is an example using the PHP code library. You can follow similar steps with our C# or Java code libraries to request a report. You can request reports using the ManageReportScheduleSample.php file located at <MWS_DIR>/src/MarketplaceWebService/Samples/. Follow the steps given below to Request a report:

6.5.1 Configure the ManageReportScheduleSample.php for your marketplace

Uncomment the $serviceUrl line corresponding to the country you are registered to sell in, so that the correct endpoint is used for submitting your report request.

For example, U.S. sellers should uncomment the line following the line "//United States:" in the file (that is, "$serviceUrl = "https://mws.amazonservices.com";").

6.5.2 Configure the parameters for your ManageReportSchedule

Uncomment the following lines in the file

```
$parameters = array (
   'Marketplace' => MARKETPLACE_ID,
   'Merchant' => MERCHANT_ID,
   'ReportType' => '_GET_ORDERS_DATA_',
   'Schedule' => '_1_HOUR_',
   'ScheduledDate' => new DateTime('now', new DateTimeZone('UTC')),
);
$request = new   MarketplaceWebService_Model_ManageReportScheduleRequest($parameters);
```

The type of the report should be configured in the file. This can be done by replacing the following line in the uncommented lines in the above step with the appropriate ReportType enumeration from the above table.

For example, for orders that need to be shipped:

```
      'ReportType' => '_GET_FLAT_FILE_ACTIONABLE_ORDER_DATA_',
```

The schedule of the report can be changed by replacing the value in the following line with one of the Schedule Enumerations given in the table above

```
'Schedule' => '_1_HOUR_',
```

Uncomment the following line for invoking the ManageReportSchedule API

```
invokeManageReportSchedule($service, $request);
```

The remaining parameters, such as MARKETPLACE_ID, MERCHANT_ID, and so on, are retrieved from the .config.inc.php file which you configured earlier.

Replace the line

```
$reportScheduleList = $manageReportScheduleResult->getReportSchedule();
```

with the following line

```
$reportScheduleList = $manageReportScheduleResult->getReportScheduleList();
```

6.5.3 Submit the ManageReportSchedule

As all the required changes were made to the ManageReportScheduleSample.php file for scheduling a report, you can save it now and run it using the command

```
php ManageReportScheduleSample.php
```

6.5.4 Process the service response

If everything has been configured correctly, you should see an output which ends with the following service response:

```
Service Response
=============================================================================
        ManageReportScheduleResponse
            ManageReportScheduleResult
                Count
                    1
                ReportSchedule
                    ReportType
                        _GET_ORDERS_DATA_
                    Schedule
                        _1_HOUR_
                    ScheduledDate
                        2011-05-19T14:33:43Z
            ResponseMetadata
                RequestId
                    a69ae40d-ef31-43af-8c42-ef97c803ea84
```

If there are errors in scheduling the report, an error message will be given in the service response which must be fixed and resubmitted using the ManageReportSchedule API.

7 Appendix E: GetReportRequestList API

7.1 Description

The GetReportRequestList operation returns a list of report requests that match the query parameters. You can specify query parameters for report status, date range, and report type. The list contains the ReportRequestId for each report request.You can use the ReportRequestId to obtain a report by passing the ID to the GetReport operation.

In the first request, a maximum of 100 report requests are returned. If there are additional report requests to return, HasNext is returned set to true in the response . To retrieve all the results, you can pass the value of the NextToken parameter to call GetReportRequestListByNextToken operation iteratively until HasNext is returned set to false.

The GetReportRequestList operation has a maximum request quota of 10 and a restore rate of 1 request per minute. Requests are included within the overall limit of 1,000 requests per seller account per hour.

7.2 ReportType Enumerations for Checkout by Amazon MWS

Report Title	Enumeration Value	Description
Unshipped Orders Report	_GET_FLAT_FILE_ACTIONABLE_ORDER_DATA_	Tab-delimited flat file report that contains only orders that are not confirmed as shipped.
Flat File Order Report	_GET_FLAT_FILE_ORDER_REPORT_DATA_	Tab-delimited flat file order report. For Seller Central sellers only.
Scheduled XML Order Report	_GET_ORDERS_DATA_	Scheduled XML order report. For Seller Central sellers only.

7.3 Request Parameters

Name	Description	Required	Valid Values
ReportRequestIdList	A structured list of ReportRequestId values. If you pass in ReportRequestId values, other query conditions are ignored. Type: xs:string	No	Default: All
ReportTypeList	A structured list of ReportType enumeration values. Type: xs:string	No	Default: All
MaxCount	A non-negative integer that represents the maximum number of report requests to return. If you specify a number greater than 100, the request is rejected. Type: xs:nonNegativeInteger	No	1-100 Default: 10

Name	Description	Required	Valid Values
RequestedFromDate	The start of a date range used for selecting the data to report, in ISO8601 date format. Type: xs:datetime	No	Default: 90 days ago
RequestedToDate	The end of a date range used for selecting the data to report, in ISO8601 date format. Type: xs: datetime	No	Default: Now

7.4 Response Elements

Name	Description
NextToken	A string token used to pass information to another call. Use the NextToken to call the operation again if the value of HasNext is true. Type: xs:string
HasNext	A Boolean value that indicates whether there are more items to return, requiring additional calls to this operation to retrieve them. The value is true if there are more items to retrieve; otherwise false. Type: xs:boolean
ReportRequestId	A unique identifier for the report request. Type: xs:string
ReportType	The ReportType enumeration value requested.
StartDate	The start of a date range used for selecting the data to report. Type: xs:datetime
EndDate	The end of a date range used for selecting the data to report. Type: xs:datetime
Scheduled	A Boolean value that indicates if a report is scheduled. The value is true if the report was scheduled; otherwise false. Type: xs:boolean
SubmittedDate	The date when the report was submitted. Type: xs:datetime
ReportProcessingStatus	The processing status of the report.

7.5 Example Using the MWS Client Library

Following is an example using the PHP code library. You can follow similar steps with our C# or Java code libraries to request a report. You can call the GetReportRequestList using the GetReportRequestListSample.php file located at <MWS_DIR>/src/MarketplaceWebService/Samples/. Follow the steps given below to get a list of reports that have been processed and marked as "_DONE_":

7.5.1 Configure the GetReportRequestListSample.php for your marketplace

Uncomment the $serviceUrl line corresponding to the country you are registered to sell in, so that the correct endpoint is used for submitting your report request.

For example: U.S. sellers should uncomment the line following the line "//United States:" in the file (that is, "$serviceUrl = "https://mws.amazonservices.com";").

7.5.2 Configure the parameters for your GetReportRequestListSample

Uncomment the following lines in the file:

```
$request = new MarketplaceWebService_Model_GetReportRequestListRequest();
$request->setMarketplace(MARKETPLACE_ID);
$request->setMerchant(MERCHANT_ID);
```

The type of the report requested earlier should be configured. This is done by adding the following lines immediately after the above uncommented lines:

```
$reportTypeList = new MarketplaceWebService_Model_TypeList();
$reportTypeList->setType("_GET_FLAT_FILE_ACTIONABLE_ORDER_DATA_");
$request->setReportTypeList($reportTypeList);
```

As you are only looking for requests which are processed, you supply the ReportProcessingStatus as "_DONE_". This is done by adding the following lines immediately below the earlier added lines:

```
$statusList = new MarketplaceWebService_Model_StatusList();
$statusList->setStatus("_DONE_");
$request->setReportProcessingStatusList($statusList);
```

Uncomment the following line to invoke the API call:

```
invokeGetReportRequestList($service, $request);
```

The remaining parameters, such as MARKETPLACE_ID, MERCHANT_ID, and so on, are from the .config.inc.php file which you configured earlier.

7.5.3 Submit the GetReportRequestListSample

As all the required changes were made to the GetReportRequestListSample.php file for getting the list of requests that are processed, save the file now and then run it using the command:

```
php GetReportRequestListSample.php
```

7.5.4 Process the service response

If everything is configured correctly, you will see an output which ends with a similar service response:

```
Service Response
=============================================================================
        GetReportRequestListResponse
            GetReportRequestListResult
                NextToken

BLmNiyWvjdwD2s1QD6a1FXhjHhmQdImrM48r5nOh7ZQCWzrPs+2MrlUgVMxABufmpeEvkpy05v/XmpfquZpjOFaXqeDl
```

```
wBye3n75Ayilvf34ETTVITqzJur6OQDEMHv5INPHrLJEjIh/tUDd12iBYHLpEZkSyE8qECbB4knvenSSb1Yrc45EQ4iS
l3NO+bgkYbIP4xGILP6UkfXE3FJ7ZO5HO0Cjmg2L
                HasNext

                ReportRequestInfo
                    ReportRequestId
                        4022565420
                    ReportType
                        _GET_FLAT_FILE_ACTIONABLE_ORDER_DATA_
                    StartDate
                        2011-04-05T10:30:05Z
                    EndDate
                        2011-05-13T10:30:05Z
                    SubmittedDate
                        2011-05-13T10:29:47Z
                    ReportProcessingStatus
                        _DONE_
```

Please note the ReportRequestIds given in the service response. Check if the ReportRequestId of your earlier requests is present here. For example, the earlier ReportRequestId 4022565420 is present; therefore you can download the report as it is processed.

8 Appendix F: Downloading the processed report

To download the processed report, the following steps are useful:

8.1 Get the associated ReportId Using GetReportList API

Following is an example using the PHP code library. You can follow similar steps with our C# or Java code libraries to request a report. You can call the GetReportList using the GetReportListSample.php file located at <MWS_DIR>/src/MarketplaceWebService/Samples/. The following changes must be made:

8.1.1 Configure the GetReportListSample.php for your marketplace

Uncomment the $serviceUrl line corresponding to the country you are registered to sell in, so that the correct endpoint is used for submitting your report request.

For example, U.S. sellers should uncomment the line following the line "//United States:" in the file (that is, "$serviceUrl = "https://mws.amazonservices.com";"). And UK sellers should uncomment the line following the line "//United Kingdom:" in the file (that is, "$serviceUrl = "https://mws.amazonservices.co.uk";").

8.1.2 Configure the parameters for your GetReportListSample

Uncomment the following lines in the file:

```
$request = new MarketplaceWebService_Model_GetReportListRequest();
$request->setMarketplace(MARKETPLACE_ID);
$request->setMerchant(MERCHANT_ID);
```

You must provide the ReportRequestId of the report that is processed by adding the following lines immediately below the uncommented lines from step 1:

```
  $reportRequestIdList = new MarketplaceWebService_Model_IdList();
  $reportRequestIdList->setId("<your ReportRequestId>");
  $request->setReportRequestIdList($reportRequestIdList);
```

Uncomment the following to line to invoke the GetReportList API call:

```
invokeGetReportList($service, $request);
```

The remaining parameters, such as MARKETPLACE_ID, MERCHANT_ID, and so on, are retrieved from the .config.inc.php file which you configured earlier.

8.1.3 Submit the GetReportListSample

As all the required changes were made to the GetReportListSample.php file for getting the list of requests that are processed, you can save it now and run it using the command

```
php GetReportListSample.php
```

8.1.4 Process the service response

If everything is configured correctly, you will see an output which ends with a similar service response:

```
Service Response
=============================================================================
        GetReportListResponse
            GetReportListResult
                HasNext

                ReportInfo
                    ReportId
                        3947794723
                    ReportType
                        _GET_FLAT_FILE_ACTIONABLE_ORDER_DATA_
                    ReportRequestId
                        4022565420
                    AvailableDate
                        2011-05-13T10:30:13Z
                    Acknowledged

            ResponseMetadata
                RequestId
                    25c6077e-7e73-455b-bed5-97db1ad11904
```

Please note the ReportId associated with your ReportRequestId's. For example, the ReportId associated with our ReportRequestId 4022565420 is 3947794723. This ReportId can be passed to the GetReport API and the processed report can be downloaded.

8.2 Downloading the Report Using GetReport

Following is an example using the PHP code library. You can follow similar steps with our C# or Java code libraries to request a report. You can call the GetReport API using the GetReportSample.php file located at <MWS_DIR>/src/MarketplaceWebService/Samples/. The following changes must be made to it:

8.2.1 Configure the GetReportSample.php for your marketplace

Uncomment the $serviceUrl line corresponding to the country you are registered to sell in, so that the correct endpoint is used for submitting your report request.

For example, U.S. sellers should uncomment the line following the line "//United States:" in the file (that is, line no. 36).

8.2.2 Configure the parameters for your GetReportSample

Uncomment the lines 89 – 97 containing the following:

```
$reportId = '<your report id>';
     $parameters = array (
      'Marketplace' => MARKETPLACE_ID,
 'Merchant' => MERCHANT_ID,
     'Report' => @fopen('php://memory', 'rw+'),
     'ReportId' => $reportId,
   );
   $request = new MarketplaceWebService_Model_GetReportRequest($parameters);
```

Replace "<your report id>" within the single quotes in line 89 with the ReportId that you noted from the GetReportList API call earlier. For example, the ReportId noted with the previous call was 3947794723, so the line will be:

```
$reportId = '3947794723';
```

By default the report is saved to memory. However, if you want to save it anywhere else, then replace "php://memory" within the single quotes in line 94 with the path to the file where you want the report to be stored. Note that the path must be accessible to the program for writing; therefore, please check the read-write permissions.

Uncomment the following line to invoke the GetReport API call:

```
invokeGetReport($service, $request);
```

The remaining parameters, such as MARKETPLACE_ID, MERCHANT_ID, and so on, are retrieved from the .config.inc.php file which you configured earlier.

8.2.3 Submit the GetReportSample

As all the required changes were made to the GetReportListSample.php file for getting the list of requests that are processed, you can save it now and run it using the command

```
php GetReportSample.php
```

8.2.4 Process the service response

If everything is configured correctly, you will see an output which ends with a similar service response:

```
Service Response
=============================================================================
        GetReportResponse
            GetReport                ContentMd5                PQ06bIHsYbBF0/p/VR09jQ==
            ResponseMetadata
                RequestId
                    97de5912-0fc8-4ae1-bf75-caf2a5a63cba
        Report Contents
order-id        order-item-id   purchase-date   payments-date   reporting-date promise-date
days-past-promise       buyer-email     buyer-name      buyer-phone-number      sku
product-name    quantity-purchased      quantity-shippedquantity-to-ship        ship-
service-level      recipient-name  ship-address-1 ship-address-2   ship-address-3  ship-city
ship-state      ship-postal-codeship-country    gift-wrap-type  gift-message-text
```

```
xxxx xxxxx xxxxx xxxxx xxxxx xxxxxxxxxxx xxxxxxxxxxxxxxx xx xxxxxxxxxxxxxxxxxxx xxxxx xxxx
xxx
xxxx xxxxx xxxxx xxxxx xxxxx xxxxxxxxxxx xxxxxxxxxxxxxxx xx xxxxxxxxxxxxxxxxxxx xxxxx xxxx
xxx
xxxx xxxxx xxxxx xxxxx xxxxx xxxxxxxxxxx xxxxxxxxxxxxxxx xx xxxxxxxxxxxxxxxxxxx xxxxx xxxx
xxx xxxx xxxxx xxxxx xxxxx xxxxx xxxxxxxxxxx xxxxxxxxxxxxxxx xx xxxxxxxxxxxxxxxxxxx xxxxx
xxxx xxx
```

The ReportContents section of the service response contains the details of the orders that must be shipped. And if you have specified a file in the above steps, the report will be stored there.

9 Appendix G: Order Acknowledgement

9.1 Description

The Order Acknowledgement feed allows a seller to acknowledge the success or failure of order receipt. This is an acknowledgement of receipt, **not** a communication of a fulfillment problem. The acknowledgement feed also permits the seller to associate their own order ID and order item IDs for future reference within other feeds.

9.2 FeedType Enumeration

FeedType enumeration used for both Order Acknowledgment and Order Cancelation is as follows:

Type	Enumeration
XML	_POST_ORDER_ACKNOWLEDGEMENT_DATA_
Flat File (tab-delimited)	_POST_FLAT_FILE_ORDER_ACKNOWLEDGEMENT_DATA_

9.3 Order Acknowledgement Dictionary

Element	Description
AmazonOrderID	Must be specified the first time that the order acknowledgement is sent for this order. If the MerchantOrderID is also specified, Amazon Payments will map the two IDs and the seller can then use their own order ID for subsequent feeds relating to that order. See base XSD for definition.
AmazonOrderItemCode	Must be specified the first time that the order acknowledgement is sent for this order item. If the MerchantOrderItemID is also specified, Amazon Payments will map the two IDs and the seller can then use their own item ID for subsequent feeds relating to that order item. See base XSD for definition.
MerchantOrderID	If the MerchantOrderID is specified with the AmazonOrderID, Amazon Payments Payments will map the two IDs and the seller can then use their own order ID for subsequent feeds relating to that order. See base XSD for definition.
MerchantOrderItemID	If the MerchantOrderItemID is specified with the AmazonOrderItemCode, Amazon Payments will map the two IDs and the seller can then use their own order item ID for subsequent feeds relating to that order item. See base XSD for definition.
StatusCode	The condition of the order receipt.

9.4 XSD

```
<?xml version="1.0"?>
<!— Revision="$Revision: #7 $" —>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
```

```
<xsd:include schemaLocation="amzn-base.xsd"/>
<xsd:element name="OrderAcknowledgement">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="AmazonOrderID"/>
<xsd:element ref="MerchantOrderID" minOccurs="0"/>
<xsd:element name="StatusCode">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:enumeration value="Success"/>
<xsd:enumeration value="Failure"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="Item" minOccurs="0" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="AmazonOrderItemCode"/>
<xsd:element ref="MerchantOrderItemID" minOccurs="0"/>
<xsd:element name="CancelReason" minOccurs="0">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:enumeration value="NoInventory"/>
<xsd:enumeration value="ShippingAddressUndeliverable"/>
<xsd:enumeration value="CustomerExchange"/>
<xsd:enumeration value="BuyerCanceled"/>
<xsd:enumeration value="GeneralAdjustment"/>
<xsd:enumeration value="CarrierCreditDecision"/>
<xsd:enumeration value="RiskAssessmentInformationNotValid"/>
<xsd:enumeration value="CarrierCoverageFailure"/>
<xsd:enumeration value="CustomerReturn"/>
<xsd:enumeration value="MerchandiseNotReceived"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

9.5 Example Feed for Order Acknowledgement

```
<?xml version="1.0"?>
<AmazonEnvelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="amzn-envelope.xsd">
<Header>
<DocumentVersion>1.01</DocumentVersion>
<MerchantIdentifier>My Store</MerchantIdentifier>
</Header>
<MessageType>OrderAcknowledgement</MessageType>
<Message>
<MessageID>1</MessageID>
<OrderAcknowledgement>
<AmazonOrderID>050-1234567-1234567</AmazonOrderID>
<MerchantOrderID>1234567</MerchantOrderID>
<StatusCode>Success</StatusCode>
<Item>
<AmazonOrderItemCode>12345678901234</AmazonOrderItemCode>
<MerchantOrderItemID>1234567</MerchantOrderItemID>
```

```
</Item>
</OrderAcknowledgement>
</Message>
</AmazonEnvelope>
```

9.6 Example Feed for Order Cancellation

```
<?xml version="1.0"?>
<AmazonEnvelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="amzn-envelope.xsd">
<Header>
<DocumentVersion>1.01</DocumentVersion>
<MerchantIdentifier>My Store</MerchantIdentifier>
</Header>
<MessageType>OrderAcknowledgement</MessageType>
<Message>
<MessageID>1</MessageID>
<OrderAcknowledgement>
<AmazonOrderID>050-1234567-1234567</AmazonOrderID>
<MerchantOrderID>1234567</MerchantOrderID>
<StatusCode>Failure</StatusCode>
<Item>
<AmazonOrderItemCode>12345678901234</AmazonOrderItemCode>
<MerchantOrderItemID>1234567</MerchantOrderItemID>
</Item>
</OrderAcknowledgement>
</Message>
</AmazonEnvelope>
```

10 Appendix H: Order Fulfillment

10.1 Description

The Order Fulfillment feed allows a seller to update Amazon with information about an order's fulfillment status. Data provided by the seller is used to update the Amazon Payments order information that appears when the buyer checks his account.

10.2 FeedType Enumeration

FeedType enumeration used for Order Fulfillment is a follows:

Type	Enumeration
XML	_POST_ORDER_FULFILLMENT_DATA_
Flat File (tab-delimited)	_POST_FLAT_FILE_FULFILLMENT_DATA_

10.3 Order Fulfillment Dictionary

Element	Description
AmazonOrderID	Seller can use the Amazon Payments ID or their own ID to identify the order. See base XSD for definition.
AmazonOrderItemCode	Seller can use the Amazon Payments ID or their own ID to identify the item in the order. See base XSD for definition.
CarrierCode	The shipping carrier that delivered the item. Enumerated options. Applicable only to direct-ship merchants.
CarrierName	The shipping carrier that delivered the item. Open string; Sellers can choose CarrierName instead of CarrierCode if the latter does not contain the carrier used. Applicable only to direct-ship merchants
FulfillmentDate	The date that the order or items were actually shipped or picked up, depending on the fulfillment method specified in the order.
MerchantFulfillmentID	A seller-specified unique identifier for the shipment.
MerchantFulfillmentItemID	A seller-specified unique identifier for an item in the shipment.
MerchantOrderID	Seller can use the Amazon Payments ID or their own ID to identify the order. See base XSD for definition
Quantity	If the item has multiple quantities, and portions of that quantity are shipped separately, the quantity is specified here. Applicable only to direct-ship merchants.
ShipperTrackingNumber	The tracking number for the shipment of the item. Applicable only to direct-ship merchants.
ShippingMethod	The shipping method used to deliver the item. Applicable only to direct-ship merchants.

10.4 XSD

```
<?xml version="1.0"?>
<!– Revision="$Revision: #2 $" –>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xsd:include schemaLocation="amzn-base.xsd"/>
<xsd:element name="OrderFulfillment">
<xsd:complexType>
<xsd:sequence>
<xsd:choice>
<xsd:element ref="AmazonOrderID"/>
<xsd:element ref="MerchantOrderID"/>
</xsd:choice>
<xsd:element name="MerchantFulfillmentID" type="IDNumber" minOccurs="0"/>
<xsd:element name="FulfillmentDate" type="xsd:dateTime"/>
<xsd:element name="FulfillmentData" minOccurs="0">
<xsd:complexType>
<xsd:sequence>
<xsd:choice>
<xsd:element ref="CarrierCode"/>
<xsd:element name="CarrierName" type="String"/>
</xsd:choice>
<xsd:element name="ShippingMethod" type="String" minOccurs="0"/>
<xsd:element name="ShipperTrackingNumber" type="String" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="Item" minOccurs="0" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:choice>
<xsd:element ref="AmazonOrderItemCode"/>
<xsd:element ref="MerchantOrderItemID"/>
</xsd:choice>
<xsd:element name="MerchantFulfillmentItemID" type="IDNumber" minOccurs="0"/>
<xsd:element name="Quantity" type="xsd:positiveInteger" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

10.5 Example Order Fulfillment Feed

```
<?xml version="1.0" encoding="UTF-8"?>
<AmazonEnvelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="amzn-envelope.xsd">
<Header>
<DocumentVersion>1.01</DocumentVersion>
<MerchantIdentifier>My Store</MerchantIdentifier>
</Header>
<MessageType>OrderFulfillment</MessageType>
<Message>
<MessageID>1</MessageID>
<OrderFulfillment>
<MerchantOrderID>1234567</MerchantOrderID>
<MerchantFulfillmentID>1234567</MerchantFulfillmentID>
<FulfillmentDate>2002-05-01T15:36:33-08:00</FulfillmentDate>
<FulfillmentData>
<CarrierCode>UPS</CarrierCode>
```

```
<ShippingMethod>Second Day</ShippingMethod>
<ShipperTrackingNumber>1234567890</ShipperTrackingNumber>
</FulfillmentData>
<Item>
<MerchantOrderItemID>1234567</MerchantOrderItemID>
<MerchantFulfillmentItemID>1234567</MerchantFulfillmentItemID>
<Quantity>2</Quantity>
</Item>
</OrderFulfillment>
</Message>
</AmazonEnvelope>
```

11 Appendix I: Order Adjustment and Partial Cancellation

11.1 Description

The Order Adjustment feed accepts data from a seller about a refund/adjustment to existing orders. Orders re identified either by the Amazon Payments order ID or by the seller's order ID, if it was previously provided in the Order Acknowledgement feed. Similarly, items within the order can be identified either by the Amazon Payments order item code or by the seller's order item ID, if it was previously provided.

Sellers must provide a reason for the adjustment, and the amounts to be adjusted, broken out by price component (principle, shipping, tax, and so on). All adjustments for an order specified within the same adjustment message constitute one "unit of work"; the buyer's credit card is credited one time for the aggregate amount.

Although the adjustment feed allows the seller to charge the buyer additional money, the net amount of the adjustment must be a credit to the buyer.

11.2 FeedType Enumeration

FeedType enumeration used for these feeds is as follows:

Type	Enumeration
XML	_POST_PAYMENT_ADJUSTMENT_DATA_
Flat File (tab-delimited)	_POST_FLAT_FILE_PAYMENT_ADJUSTMENT_DATA_

11.3 Order Adjustment Dictionary

Element	Description
AdjustmentReason	The reason for the refund/adjustment. Can drive fee refund logic for the merchant.
AmazonOrderID	If the MerchantOrderID was specified in the Order Acknowledgement feed, then the seller can use either the Amazon Payments order ID, or their own. If the MerchantOrderID was not specified, AmazonOrderID must be used. See base XSD for definition.
AmazonOrderItemCode	If the MerchantOrderItemID was specified in the Order Acknowledgement feed, then the seller can use either Amazon's order item code, or their own ID. If the MerchantOrderItemID was not MerchantAdjustmentItemID specified, AmazonOrderItemCode must be used. See base XSD for definition. The amounts that the buyer is to be refunded for the item, broken out by type. All amounts are aggregates of the quantity, not unit prices. See base XSD for definition of type. Amounts are signed; positive amounts are refunded to the buyer and negative amounts are charged to the buyer. A merchant-defined identifier for the

Element	Description
	adjustment.
MerchantOrderID	If the MerchantOrderID was specified in the Order Acknowledgement feed, then the seller can use either Amazon's order ID, or their own. If the MerchantOrderID was not specified, AmazonOrderID must be used. See base XSD for definition.
MerchantOrderItemID	If the MerchantOrderItemID was specified in the Order Acknowledgement feed, then the seller can use either the Amazon Payments order item code, or their own ID. If the MerchantOrderItemID was not specified, AmazonOrderItemCode must be used. See base XSD for definition.
PromotionAdjustments	The amounts that the buyer is to be refunded for a promotion, broken out by type. All amounts are aggregates of the quantity, not unit prices. See base XSD for definition of type. Amounts are signed; positive amounts are refunded to the buyer and negative amounts are charged to the buyer.
Quantity	Quantity of items being canceled—used only for partial cancellation.

11.4 XSD

```
<?xml version="1.0"?>
<!— Revision="$Revision: #11 $" —>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xsd:include schemaLocation="amzn-base.xsd"/>
<xsd:element name="OrderAdjustment">
<xsd:complexType>
<xsd:sequence>
<xsd:choice>
<xsd:element ref="AmazonOrderID"/>
<xsd:element ref="MerchantOrderID"/>
</xsd:choice>
<xsd:element name="AdjustedItem" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:choice>
<xsd:element ref="AmazonOrderItemCode"/>
<xsd:element ref="MerchantOrderItemID"/>
</xsd:choice>
<xsd:element name="MerchantAdjustmentItemID" type="StringNotNull"
minOccurs="0"/>
<xsd:element name="AdjustmentReason">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:enumeration value="NoInventory"/>
<xsd:enumeration value="CustomerReturn"/>
<xsd:enumeration value="GeneralAdjustment"/>
<xsd:enumeration value="CouldNotShip"/>
<xsd:enumeration value="DifferentItem"/>
<xsd:enumeration value="Abandoned"/>
<xsd:enumeration value="CustomerCancel"/>
<xsd:enumeration value="PriceError"/>
<xsd:enumeration value="ProductOutofStock"/>
<xsd:enumeration value="CustomerAddressIncorrect"/>
```

```
<xsd:enumeration value="Exchange"/>
<xsd:enumeration value="Other"/>
<xsd:enumeration value="CarrierCreditDecision"/>
<xsd:enumeration value="RiskAssessmentInformationNotValid"/>
<xsd:enumeration value="CarrierCoverageFailure"/>
<xsd:enumeration value="TransactionRecord"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="ItemPriceAdjustments" type="BuyerPrice"/>
<xsd:element name="PromotionAdjustments" minOccurs="0" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="PromotionClaimCode" minOccurs="0"/>
<xsd:element ref="MerchantPromotionID" minOccurs="0"/>
<xsd:element name="Component" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="Type" type="PromotionApplicationType"/>
<xsd:element name="Amount" type="CurrencyAmount"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="DirectPaymentAdjustments" type="DirectPaymentType"
minOccurs="0"/>
<xsd:element name="QuantityCancelled" type="xsd:positiveInteger"
minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

11.5 Example Order Adjustment Feed

```
<?xml version="1.0" encoding="UTF-8"?>
<AmazonEnvelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="amzn-envelope.xsd">
<Header>
<DocumentVersion>1.01</DocumentVersion>
<MerchantIdentifier>My Store</MerchantIdentifier>
</Header>
<MessageType>OrderAdjustment</MessageType>
<Message>
<MessageID>1</MessageID>
<OrderAdjustment>
<MerchantOrderID>1234567</MerchantOrderID>
<AdjustedItem>
<MerchantOrderItemID>1234567</MerchantOrderItemID>
<MerchantAdjustmentItemID>12345</MerchantAdjustmentItemID>
<AdjustmentReason>CustomerReturn</AdjustmentReason>
<ItemPriceAdjustments>
<Component>
<Type>Principal</Type>
<Amount currency="USD">10.00</Amount>
</Component>
<Component>
```

```
<Type>Shipping</Type>
<Amount currency="USD">3.49</Amount>
</Component>
<Component>
<Type>Tax</Type>
<Amount currency="USD">1.29</Amount>
</Component>
<Component>
<Type>Shipping Tax</Type>
<Amount currency="USD">0.24</Amount>
</Component>
</ItemPriceAdjustments>
</AdjustedItem>
</OrderAdjustment>
</Message>
<Message>
<MessageID>2</MessageID>
<OrderAdjustment>
<MerchantOrderID>2345678</MerchantOrderID>
<AdjustedItem>
<MerchantOrderItemID>2345678</MerchantOrderItemID>
<MerchantAdjustmentItemID>23456</MerchantAdjustmentItemID>
<AdjustmentReason>CustomerReturn</AdjustmentReason>
<ItemPriceAdjustments>
<Component>
<Type>Principal</Type>
<Amount currency="USD">10.00</Amount>
</Component>
<Component>
<Type>Tax</Type>
<Amount currency="USD">1.29</Amount>
</Component>
</ItemPriceAdjustments>
<PromotionAdjustments>
<PromotionClaimCode>ABC123</PromotionClaimCode>
<MerchantPromotionID>12345678</MerchantPromotionID>
<Component>
<Type>Principal</Type>
<Amount currency="USD">-1.00</Amount>
</Component>
</PromotionAdjustments>
</AdjustedItem>
</OrderAdjustment>
</Message>
</AmazonEnvelope>
```

12 Appendix K: Order Fulfillment

12.1 Description

The Order Fulfillment feed allows a seller to update Amazon with information about an order's fulfillment status. Data provided by the seller is used to update the Amazon Payments order information that appears when the buyer checks his account.

12.2 FeedType Enumeration

FeedType enumeration used for Order Fulfillment is a follows:

Type	Enumeration
XML	_POST_ORDER_FULFILLMENT_DATA_
Flat File (tab-delimited)	_POST_FLAT_FILE_FULFILLMENT_DATA_

12.3 Order Fulfillment Dictionary

Element	**Description**
AmazonOrderID	Seller can use the Amazon Payments ID or their own ID to identify the order. See base XSD for definition.
AmazonOrderItemCode	Seller can use the Amazon Payments ID or their own ID to identify the item in the order. See base XSD for definition.
CarrierCode	The shipping carrier that delivered the item. Enumerated options. Applicable only to direct-ship merchants.
CarrierName	The shipping carrier that delivered the item. Open string; Sellers can choose CarrierName instead of CarrierCode if the latter does not contain the carrier used. Applicable only to direct-ship merchants
FulfillmentDate	The date that the order or items were actually shipped or picked up, depending on the fulfillment method specified in the order.
MerchantFulfillmentID	A seller-specified unique identifier for the shipment.
MerchantFulfillmentItemID	A seller-specified unique identifier for an item in the shipment.
MerchantOrderID	Seller can use the Amazon Payments ID or their own ID to identify the order. See base XSD for definition
Quantity	If the item has multiple quantities, and portions of that quantity are shipped separately, the quantity is specified here. Applicable only to direct-ship merchants.
ShipperTrackingNumber	The tracking number for the shipment of the item. Applicable only to direct-ship merchants.
ShippingMethod	The shipping method used to deliver the item. Applicable only to direct-ship merchants.

12.4 XSD

```
<?xml version="1.0"?>
<!— Revision="$Revision: #2 $" —>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xsd:include schemaLocation="amzn-base.xsd"/>
<xsd:element name="OrderFulfillment">
<xsd:complexType>
<xsd:sequence>
<xsd:choice>
<xsd:element ref="AmazonOrderID"/>
<xsd:element ref="MerchantOrderID"/>
</xsd:choice>
<xsd:element name="MerchantFulfillmentID" type="IDNumber" minOccurs="0"/>
<xsd:element name="FulfillmentDate" type="xsd:dateTime"/>
<xsd:element name="FulfillmentData" minOccurs="0">
<xsd:complexType>
<xsd:sequence>
<xsd:choice>
<xsd:element ref="CarrierCode"/>
<xsd:element name="CarrierName" type="String"/>
</xsd:choice>
<xsd:element name="ShippingMethod" type="String" minOccurs="0"/>
<xsd:element name="ShipperTrackingNumber" type="String" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="Item" minOccurs="0" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:choice>
<xsd:element ref="AmazonOrderItemCode"/>
<xsd:element ref="MerchantOrderItemID"/>
</xsd:choice>
<xsd:element name="MerchantFulfillmentItemID" type="IDNumber" minOccurs="0"/>
<xsd:element name="Quantity" type="xsd:positiveInteger" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

12.5 Example Order Fulfillment Feed

```
<?xml version="1.0" encoding="UTF-8"?>
<AmazonEnvelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="amzn-envelope.xsd">
<Header>
<DocumentVersion>1.01</DocumentVersion>
<MerchantIdentifier>My Store</MerchantIdentifier>
</Header>
<MessageType>OrderFulfillment</MessageType>
<Message>
<MessageID>1</MessageID>
<OrderFulfillment>
<MerchantOrderID>1234567</MerchantOrderID>
<MerchantFulfillmentID>1234567</MerchantFulfillmentID>
<FulfillmentDate>2002-05-01T15:36:33-08:00</FulfillmentDate>
<FulfillmentData>
<CarrierCode>UPS</CarrierCode>
```

```
<ShippingMethod>Second Day</ShippingMethod>
<ShipperTrackingNumber>1234567890</ShipperTrackingNumber>
</FulfillmentData>
<Item>
<MerchantOrderItemID>1234567</MerchantOrderItemID>
<MerchantFulfillmentItemID>1234567</MerchantFulfillmentItemID>
<Quantity>2</Quantity>
</Item>
</OrderFulfillment>
</Message>
</AmazonEnvelope>
```

13 Appendix L: Order Adjustment and Partial Cancellation

13.1 Description

The Order Adjustment feed accepts data from a seller about a refund/adjustment to existing orders. Orders can be identified either by the Amazon Payments order ID or by the seller's order ID, if it was previously provided in the Order Acknowledgement feed. Similarly, items within the order can be identified either by the Amazon Payments order item code or by the seller's order item ID, if it was previously provided.

Sellers must provide a reason for the adjustment, and the amounts to be adjusted, broken out by price component (principle, shipping, tax, and so on). All adjustments for an order specified within the same adjustment message constitute one "unit of work"; the buyer's credit card is credited one time for the aggregate amount.

Although the adjustment feed allows the seller to charge the buyer additional money, the net amount of the adjustment must be a credit to the buyer.

13.2 FeedType Enumeration

FeedType enumeration used for these feeds is as follows:

Type	Enumeration
XML	_POST_PAYMENT_ADJUSTMENT_DATA_
Flat File (tab-delimited)	_POST_FLAT_FILE_PAYMENT_ADJUSTMENT_DATA_

13.3 Order Adjustment Dictionary

Element	Description
AdjustmentReason	The reason for the refund/adjustment. Can drive fee refund logic for the merchant.
AmazonOrderID	If the MerchantOrderID was specified in the Order Acknowledgement feed, then the seller can use either the Amazon Payments order ID, or their own. If the MerchantOrderID was not specified, AmazonOrderID must be used. See base XSD for definition.
AmazonOrderItemCode	If the MerchantOrderItemID was specified in the Order Acknowledgement feed, then the seller can use either Amazon's order item code, or their own ID. If the MerchantOrderItemID was not MerchantAdjustmentItemID specified, AmazonOrderItemCode must be used. See base XSD for definition. The amounts that the buyer is to be refunded for the item, broken out by type. All amounts are aggregates of the quantity, not unit prices. See base XSD for definition of type. Amounts are signed; positive amounts are refunded to the buyer and negative amounts are charged to the buyer. A merchant-defined identifier for the

Element	Description
	adjustment.
MerchantOrderID	If the MerchantOrderID was specified in the Order Acknowledgement feed, then the seller can use either Amazon's order ID, or their own. If the MerchantOrderID was not specified, AmazonOrderID must be used. See base XSD for definition.
MerchantOrderItemID	If the MerchantOrderItemID was specified in the Order Acknowledgement feed, then the seller can use either the Amazon Payments order item code, or their own ID. If the MerchantOrderItemID was not specified, AmazonOrderItemCode must be used. See base XSD for definition.
PromotionAdjustments	The amounts that the buyer is to be refunded for a promotion, broken out by type. All amounts are aggregates of the quantity, not unit prices. See base XSD for definition of type. Amounts are signed; positive amounts are refunded to the buyer and negative amounts are charged to the buyer.
Quantity	Quantity of items being canceled—used only for partial cancellation.

13.4 XSD

```
<?xml version="1.0"?>
<!– Revision="$Revision: #11 $" –>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xsd:include schemaLocation="amzn-base.xsd"/>
<xsd:element name="OrderAdjustment">
<xsd:complexType>
<xsd:sequence>
<xsd:choice>
<xsd:element ref="AmazonOrderID"/>
<xsd:element ref="MerchantOrderID"/>
</xsd:choice>
<xsd:element name="AdjustedItem" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:choice>
<xsd:element ref="AmazonOrderItemCode"/>
<xsd:element ref="MerchantOrderItemID"/>
</xsd:choice>
<xsd:element name="MerchantAdjustmentItemID" type="StringNotNull"
minOccurs="0"/>
<xsd:element name="AdjustmentReason">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:enumeration value="NoInventory"/>
<xsd:enumeration value="CustomerReturn"/>
<xsd:enumeration value="GeneralAdjustment"/>
<xsd:enumeration value="CouldNotShip"/>
<xsd:enumeration value="DifferentItem"/>
<xsd:enumeration value="Abandoned"/>
<xsd:enumeration value="CustomerCancel"/>
<xsd:enumeration value="PriceError"/>
<xsd:enumeration value="ProductOutofStock"/>
<xsd:enumeration value="CustomerAddressIncorrect"/>
```

```
<xsd:enumeration value="Exchange"/>
<xsd:enumeration value="Other"/>
<xsd:enumeration value="CarrierCreditDecision"/>
<xsd:enumeration value="RiskAssessmentInformationNotValid"/>
<xsd:enumeration value="CarrierCoverageFailure"/>
<xsd:enumeration value="TransactionRecord"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="ItemPriceAdjustments" type="BuyerPrice"/>
<xsd:element name="PromotionAdjustments" minOccurs="0" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="PromotionClaimCode" minOccurs="0"/>
<xsd:element ref="MerchantPromotionID" minOccurs="0"/>
<xsd:element name="Component" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="Type" type="PromotionApplicationType"/>
<xsd:element name="Amount" type="CurrencyAmount"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="DirectPaymentAdjustments" type="DirectPaymentType"
minOccurs="0"/>
<xsd:element name="QuantityCancelled" type="xsd:positiveInteger"
minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

13.5 Example Order Adjustment Feed

```
<?xml version="1.0" encoding="UTF-8"?>
<AmazonEnvelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="amzn-envelope.xsd">
<Header>
<DocumentVersion>1.01</DocumentVersion>
<MerchantIdentifier>My Store</MerchantIdentifier>
</Header>
<MessageType>OrderAdjustment</MessageType>
<Message>
<MessageID>1</MessageID>
<OrderAdjustment>
<MerchantOrderID>1234567</MerchantOrderID>
<AdjustedItem>
<MerchantOrderItemID>1234567</MerchantOrderItemID>
<MerchantAdjustmentItemID>12345</MerchantAdjustmentItemID>
<AdjustmentReason>CustomerReturn</AdjustmentReason>
<ItemPriceAdjustments>
<Component>
<Type>Principal</Type>
<Amount currency="USD">10.00</Amount>
</Component>
<Component>
```

```
<Type>Shipping</Type>
<Amount currency="USD">3.49</Amount>
</Component>
<Component>
<Type>Tax</Type>
<Amount currency="USD">1.29</Amount>
</Component>
<Component>
<Type>Shipping Tax</Type>
<Amount currency="USD">0.24</Amount>
</Component>
</ItemPriceAdjustments>
</AdjustedItem>
</OrderAdjustment>
</Message>
<Message>
<MessageID>2</MessageID>
<OrderAdjustment>
<MerchantOrderID>2345678</MerchantOrderID>
<AdjustedItem>
<MerchantOrderItemID>2345678</MerchantOrderItemID>
<MerchantAdjustmentItemID>23456</MerchantAdjustmentItemID>
<AdjustmentReason>CustomerReturn</AdjustmentReason>
<ItemPriceAdjustments>
<Component>
<Type>Principal</Type>
<Amount currency="USD">10.00</Amount>
</Component>
<Component>
<Type>Tax</Type>
<Amount currency="USD">1.29</Amount>
</Component>
</ItemPriceAdjustments>
<PromotionAdjustments>
<PromotionClaimCode>ABC123</PromotionClaimCode>
<MerchantPromotionID>12345678</MerchantPromotionID>
<Component>
<Type>Principal</Type>
<Amount currency="USD">-1.00</Amount>
</Component>
</PromotionAdjustments>
</AdjustedItem>
</OrderAdjustment>
</Message>
</AmazonEnvelope>
```

14 Appendix M: SubmitFeed

14.1 Description

The SubmitFeed operation uploads a file and any necessary metadata for processing. Note that you must calculate a Content-MD5 header for the submitted file.

The SubmitFeed operation has a maximum request quota of 15 and a restore rate of one request every two minutes. Requests are included within the overall limit of 1,000 requests per seller account per hour.

Feed size is limited to 2,147,483,647 bytes (2^31 -1) per feed. If you have a large amount of data to submit, you should submit feeds smaller than the feed size limit by breaking up the data, or submit the feeds over a period of time. One good practice is to submit feeds with a size limit of 30,000 records/items or submit feeds over a period of time, such as every few hours.

14.2 Setting the Content-Type for a Feed

Your feeds must be in a valid encoding based on your marketplace and file type, and that encoding must be specified as an HTTP Content-Type header.The following table shows the HTTP Content-Type header you should use for flat files and XML files for each marketplace:

Marketplace	Flat-File Content-Type	XML Content-Type
North America and Europe	Text/tab-separated-values; charset=iso-8859-1	text/xml

14.3 Request Parameters

Name	Description	Required	Valid values
FeedContent	The actual content of the feed itself, in XML or flat file format. You must FeedContent include the in the body of the HTTP request. Type: HTTP-BODY	Yes	Default: none
FeedType	A FeedType enumeration value indicating how the data should be processed. Type: xs:string	Yes	Default: None
PurgeAndReplace	A Boolean value that enables the purge and replace functionality. Set to true to purge and replace the existing data; otherwise false. This value only applies to	No	Default: false

Name	Description	Required	Valid values
	product-related flat file feed types, which do not have a mechanism for specifying purge and replace in the feed body. Use this parameter only in exceptional cases. Usage is throttled to allow only one purge and replace within a 24-hour period. Type: xs:boolean		

14.4 Response Elements

Name	Description
FeedSubmissionId	A unique identifier for the feed submission. Type: xs:string
FeedType	The type of feed submitted. This is the FeedType enumeration SubmitFeed value that was provided to the operation.
SubmittedDate	The date and time when the feed was submitted. Type: xs:datetime
FeedProcessingStatus	The processing status of the feed submission.

14.5 Example Using the MWS Client Library

Following is an example using the PHP code library. You can follow similar steps with our C# or Java code libraries to request a report. You can call the SubmitFeed using the SubmitFeedSample.php file located at <MWS_DIR>/src/MarketplaceWebService/Samples/. Follow the steps given below to submit a feed to MWS:

14.5.1 Configure the SubmitFeedSample.php for your marketplace

Uncomment the $serviceUrl line corresponding to the country you are registered to sell in, so that the correct endpoint is used for submitting your report request.

For example, U.S. sellers should uncomment the line following the line "//United States:" in the file (that is, "$serviceUrl = "https://mws.amazonservices.com";").

14.5.2 Configure the parameters for your SubmitFeedSample

Put in the correct FeedType according to the type of Feed you are sending at the line "<'FeedType' => '_POST_ORDER_FULFILLMENT_DATA_'," replacing "_POST_ORDER_FULFILLMENT_DATA_" within the single quotes. Refer to the various types mentioned in the Order Management Feeds section of Step 4.a for available options. For example, if you are sending an acknowledgement feed, then the FeedType in line 134 of the file should be set as

```
'FeedType' => ' _POST_ORDER_ACKNOWLEDGEMENT_DATA_',
```

Construct your XML feed based on the various types of feeds explained in the Order Management Feeds section. Once you have feed constructed, replace the lines after "$feed = <<<EOD" and before "EOD;".

The remaining parameters, such as MARKETPLACE_ID, MERCHANT_ID, and so on, are retrieved from the .config.inc.php file which you configured earlier.

14.5.3 Submit the SubmitFeedSample

As all the required changes were made to the SubmitFeedSample.php file for submitting the feed for processing, you can save it now and run it using the command:

```
php SubmitFeedSample.php
```

14.5.4 Process the service response

If everything is configured correctly, you will see an output which ends with a similar service response:

```
Service Response
=============================================================================
        SubmitFeedResponse
            SubmitFeedResult
                FeedSubmissionInfo
                    FeedSubmissionId
                        4023466790
                    FeedType
                        _POST_ORDER_ACKNOWLEDGEMENT_DATA_
                    SubmittedDate
                        2011-05-13T18:18:44Z
                    FeedProcessingStatus
                        _SUBMITTED_
            ResponseMetadata
                RequestId
                    638774a9-d1be-4914-8caf-d5e9e681c524
```

Please note the FeedSubmissionId given in the service response. In this case it is "4023466790". Also note that FeedProcessingStatus is "_SUBMITTED_"; otherwise, this indicates an error in the SubmitFeed API call.

15 Appendix N: GetFeedSubmissionList

15.1 Description

The GetFeedSubmissionList operation returns a list of feed submissions submitted in the previous 90 days that match the query parameters. Use this operation to determine the status of a feed submission by passing in the FeedProcessingId that was returned by the SubmitFeed operation.

The GetFeedSubmissionList request can return a maximum of 100 results. If there are additional results to return, HasNext is returned in the response with a true value. To retrieve all the results, you can pass the value of the NextToken parameter to the GetFeedSubmissionListByNextToken operation and repeat until HasNext is false.

The GetFeedSubmissionList operation has a maximum request quota of 10 and a restore rate of one request per minute. Requests are included within the overall limit of 1,000 requests per seller account per hour.

15.2 Request Parameters

Name	Description	Required	Valid values
FeedSubmissionIdList	A structured list of FeedSubmmissionId values. If you pass in FeedSubmmissionId values in a request, other query conditions are ignored. Type: xs:string	No	Default: All
MaxCount	A non-negative integer that indicates the maximum number of feed submissions to return in the list. If you specify a number greater than 100, the request is rejected. Type: xs:nonNegativeInteger	No	Default: 10
FeedTypeList	A structured list of one or more FeedType enumeration values by which to filter the list of feed submissions.	No	Default: All feed types

Name	Description	Required	Valid values
	Type: xs:string		
FeedProcessingStatusList	A structured list of one or more feed processing statuses by which to filter the list of feed submissions. Type: xs:string	No	Default: All feed types _SUBMITTED_, _IN_PROGRESS_, _CANCELLED_, _DONE_
SubmittedFromDate	The earliest submission date that you are looking for, in ISO8601 date format. xs:datetime	No	Default: 30 days ago
SubmittedToDate	The latest submission date that you are looking for, in ISO8601 date format. Type: xs:datetime	No	Default: Now

15.3 Response Elements

Name	Description
NextToken	A generated string used to pass information to another call. Pass the NextToken value to the GetFeedSubmissionListByNextToken operation if the value of HasNext is true. Type: xs:string
HasNext	A Boolean value that indicates whether there are more items to retrieve, requiring additional requests to GetGetFeedSubmissionListByNextToken to retrieve them. The value true means there are more items to retrieve; otherwise false. Type: xs:boolean
FeedSubmissionId	A unique identifier for the feed submission. Type: xs:string
FeedType	The type of feed submitted. This is the FeedType enumeration value that was provided to the SubmitFeed operation.
SubmittedDate	The date and time when the feed was submitted. Type: xs:datetime
FeedProcessingStatus	The processing status of the feed submission.

15.4 Example using the MWS Client Library

Following is an example using the PHP code library. You can follow similar steps with our C# or Java code libraries to request a report. You can call the GetFeedSubmissionList using the

GetFeedSubmissionListSample.php file located at <MWS_DIR>/src/MarketplaceWebService/Samples/. Follow the steps given below to submit a feed to MWS:

15.4.1 Configure the GetFeedSubmissionListSample.php for your marketplace

Uncomment the $serviceUrl line corresponding to the country you are registered to sell in, so that the correct endpoint is used for submitting your report request.

For example, U.S. sellers should uncomment the line following the line "//United States:" in the file (that is, "$serviceUrl = "https://mws.amazonservices.com";").

15.4.2 Configure the parameters for your GetFeedSubmissionListSample

Uncomment the following lines:

```
$parameters = array (
  'Marketplace' => MARKETPLACE_ID,
  'Merchant' => MERCHANT_ID,
  'FeedProcessingStatusList' => array ('Status' => array ('_SUBMITTED_')),
);

$request = new MarketplaceWebService_Model_GetFeedSubmissionListRequest($parameters)
```

Replace the FeedProcessingStatus in the above parameter from _SUBMITTED_ to _DONE_, as shown below:

```
'FeedProcessingStatusList' => array ('Status' => array ('_DONE_')),
```

Uncomment the following line to invoke the GetFeedSubmissionListRequest

```
invokeGetFeedSubmissionList($service, $request);
```

15.4.3 Submit the GetFeedSubmissionListSample

As all the required changes were made to the GetFeedSubmissionListSample.php file for getting the list of requests that are processed, you can save it now and run it using the command:

```
php GetFeedSubmissionListSample.php
```

15.5 Process the Service Response

If everything is configured correctly, you will see an output which ends with a similar service response:

```
Service Response
=============================================================================
        GetFeedSubmissionListResponse
```

```
            GetFeedSubmissionListResult
                NextToken

hTN/2Z4JUrwD2s1QD6a1FXaanQXDlc/XM48r5nOh7ZQbKJggJnjqn6jqg5ME44F3glRdx/DTLB8aO3icJZeOivEjssjz
s0t/OdV+RyiztljCv/LqsB0EzxybTwcKfw+682haZBfsyOEAyQB17i4ScnoXTW2mZ9sdQjlAMtCf+rkzXvmYAfGnc476
IMcuvsqB5QNwUej+Jm7PZjTnVM/HGQ==

                HasNext

                    1

                FeedSubmissionInfo

                    FeedSubmissionId

                        4035585452

                    FeedType

                        _POST_ORDER_ACKNOWLEDGEMENT_DATA_

                    SubmittedDate

                        2011-05-18T11:17:47Z

                    FeedProcessingStatus

                        _DONE_

                    StartedProcessingDate

                        2011-05-18T11:18:12Z

                    CompletedProcessingDate

                        2011-05-18T11:20:35Z

           ResponseMetadata

                RequestId

                    3cbd5d0c-ee29-40b5-be02-2609749b033a
```

Please note the FeedSubmissionId given in the service response. In this case it is "4035585452". Also note that FeedProcessingStatus is "_DONE_"; therefore, you can now download the processing report.

16 Appendix O: GetFeedSubmissionResult

16.1 Description

The GetFeedSubmissionResult operation returns the feed processing report and the Content-MD5 header for the returned HTTP body.

You should compute the MD5 hash of the HTTP body of the report that MWS returned to you, and compare that with the Content-MD5 header value that is returned. If the computed hash value and the returned hash value do not match, the report body was corrupted during transmission; you should therefore should discard the result and retry the request up to three more times. Please notify MWS if you receive a corrupted report body.

The GetFeedSubmissionResult operation has a maximum request quota of 15 and a restore rate of one request per minute. Requests are included within the overall limit of 1,000 requests per seller account per hour.

16.2 Request Parameters

Name	**Description**	**Required**	**Valid values**
FeedSubmissionId	The identifier of the feed submission you are requesting a feed processing report for. You can get the FeedSubmissionId for a feed using the GetFeedSubmissionList operation. Type: xs:string	Yes	A FeedSubmissionId value.

16.3 Response Elements

The GetFeedSubmissionResult operation returns the feed processing report and the Content-MD5 header for the returned HTTP body.

16.4 Example Using the MWS Client Library

Following is an example using the PHP code library. You can follow similar steps with our C# or Java code libraries to request a report. You can call the GetFeedSubmissionResult using the GetFeedSubmissionResultSample.php file located at <MWS_DIR>/src/MarketplaceWebService/Samples/. Follow the steps given below to submit a feed to MWS:

16.4.1 Configure the GetFeedSubmissionResultSample.php for your marketplace

Uncomment the $serviceUrl line corresponding to the country you are registered to sell in, so that the correct endpoint is used for submitting your report request.

For example, U.S. sellers should uncomment the line following the line "//United States:" in the file (that is, "$serviceUrl = "https://mws.amazonservices.com";").

16.5 Configure the Parameters for Your GetFeedSubmissionResultSample

Uncomment the following lines

```
$parameters = array (

    'Marketplace' => MARKETPLACE_ID,

    'Merchant' => MERCHANT_ID,

    'FeedSubmissionId' => '<Feed Submission Id>',

    'FeedSubmissionResult' => @fopen('php://memory', 'rw+'),

  );
$request = new MarketplaceWebService_Model_GetFeedSubmissionResultRequest($parameters);
```

Replace "<Feed Submission Id>" within the single quotes in the above paramaters with the FeedSubmissionId that you have collected from the earlier GetFeedSubmissionList api call. For example, The FeedSubmissionId that you noted down in our previous call was *4035585452, so the line would be*

```
'FeedSubmissionId' => '4035585452',
```

By default the Report is saved to memory. However if you want to save it anywhere else replace "php://memory" within the single quotes in the following line

```
'FeedSubmissionResult' => @fopen('php://memory', 'rw+'),
```

with the path to the file where you want the report to be stored. Note that the path must be accessible to the program for writing; therefore, please check the read-write permissions.

Uncomment the following line to invoke the GetFeedSubmissionResult api call

```
invokeGetFeedSubmissionResult($service, $request);
```

16.6 Submit the GetFeedSubmissionResultSample

As all the required changes were made to the GetFeedSubmissionResultSample.php file for downloading the processed feed result, you can save it now and run it using the command:

```
php GetFeedSubmissionResultSample.php
```

16.7 Process the Service Response

If everything is configured correctly, you will see an output which ends with a similar service response:

```
Service Response
=============================================================================
        GetFeedSubmissionResultResponse

            GetFeedSubmissionResult
                ContentMd5
                h2JDkIYr15Y1KrcgtXQLQw==

            ResponseMetadata

                RequestId

                    f53f731e-cf72-4a17-8193-b533a1a71966
```

The processed report will be saved at the location mentioned at the path mentioned in the configuration above.

Note: Check the ContentMD5 of the report to see if it matches with the value given in the service response to avoid any damage during download.