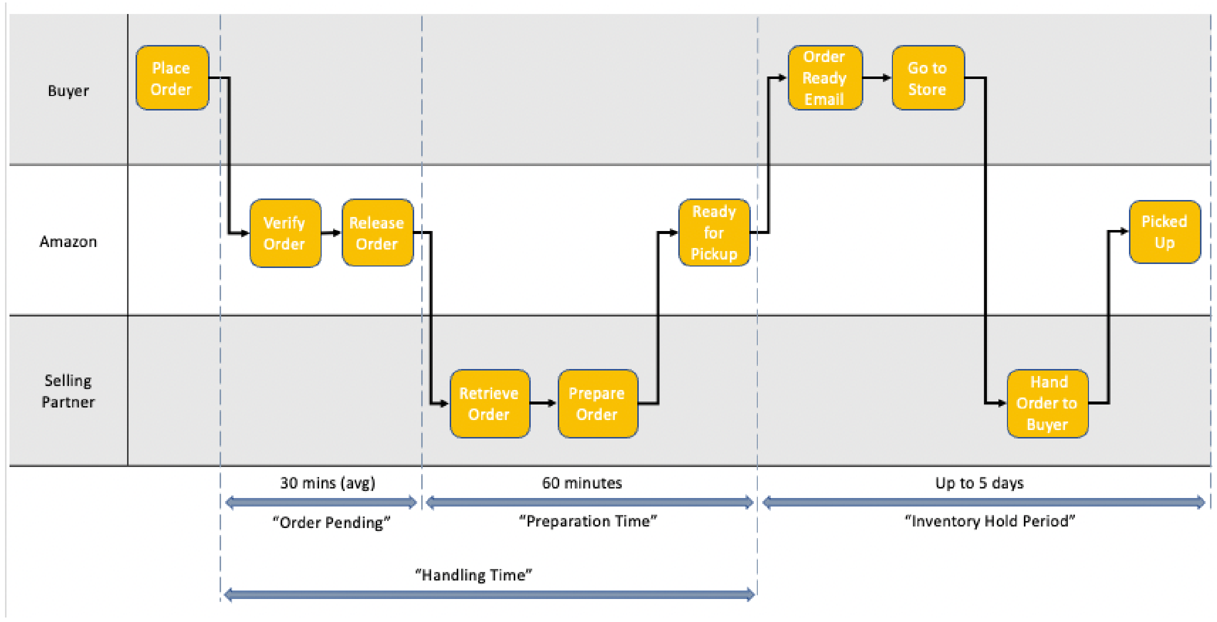
BOPIS Order Management

Introduction

The following document provides a guide to implementing and managing BOPIS Orders through the Amazon Selling Partner API.

BOPIS Order Lifecycle

BOPIS orders have a specific lifecycle, which differs from the typical seller-fulfilled, delivery order model. This is due to the requirement to process orders within a much shorter timeframe (typically within 90 minutes of the order being placed), and a 2-stage process for fulfilling these orders ("Ready for Pickup" and "Picked Up"). The following diagram shows the happy path for a BOPIS order.



Retrieving BOPIS Orders from Amazon

When an order is placed by a buyer on Amazon, it is held in a "Pending" status by Amazon for 30 minutes (average time). During this time, the order and payment details are verified. You can optionally receive partial order details when the order is "Pending" (e.g. order items), if you intend to prepare the order, but the you must not fulfill the order until it has been verified and confirmed by Amazon. The buyer also has an opportunity to cancel the order during this "Pending" status period. Should the buyer do so, no action is required by the you.

Once successfully verified, the order moves to an "Unshipped" status and the full order details are made available to the you. You can either check periodically for new orders ("pull model") or subscribe to receive order notifications via the Selling Partner Notifications API ("push model"). Retrieving order data can be done through the Orders API (synchronous request / response) or via the Reports API (asynchronous report request / process / download).

To retrieve order details from Amazon, the following prerequisites must be met:

- Approval for the **Inventory and Order Tracking** role in your Developer Profile.

- **Inventory and Order Tracking** role selected in the App registration page for your application.

To access buyer and/or shipping address information, you must have approval for the following roles and add these to your App:

- **Direct-to-Consumer Delivery (Restricted)** role is required to access shipping address information.
- **Tax Remittance (Restricted)** role is required to access buyer information.
- **Tax Invoicing (Restricted)** role is required to access buyer information.

Note that Amazon creates a separate BOPIS order for every item and store combination.   If a buyer orders multiple BOPIS items from the same Selling Partner, a separate Amazon order will be created for each BOPIS product. If a buyer orders the same item from separate stores belonging to the same Selling Partner, then a separate Amazon order will be created for each store.

RESTRICTED DATA TOKENS

Before you can retrieve PII (Personally Identifiable Information) from Amazon, you must have completed the Developer Profile form and for this to be reviewed and approved by Amazon.   As part of your Developer Profile form submission, you should review the list of roles available to decide which ones you require.

Once approved, you must add your required roles to your application in Seller Central, then re-authorize the application to generate a new Refresh Token.

To retrieve PII data, you send a Restricted Data Token in your request header instead of an Access Token, using the same **x-amz-access-token** header key.   To first create a Restricted Data Token, you must call the **createRestrictedDataToken** API, passing in an Access Token in the **x-amz-access-token** header key.

Full details on the Token API can be found here and a use-case guide is available here.

Example : request a Restricted Data Token to retrieve buyer info and shipping address info when calling Get Orders

```
POST /tokens/2021-03-01/restrictedDataToken
{
    "restrictedResources": [{
        "method": "GET",
        "path": "/orders/v0/orders",
        "dataElements": ["buyerInfo", "shippingAddress"]
    }]
}
```

Example : request a Restricted Data Token to retrieve buyer info and shipping address info when calling Get Order, using a Specific Path

```
POST /tokens/2021-03-01/restrictedDataToken
{
    "restrictedResources": [{
        "method": "GET",
        "path": "/orders/v0/orders/202-2484992-1234567",
        "dataElements": ["buyerInfo", "shippingAddress"]
    }]
}
```

Example : request a Restricted Data Token to retrieve buyer info and shipping address info when calling Get Order, using a Generic Path

```
POST /tokens/2021-03-01/restrictedDataToken
{
    "restrictedResources": [{
        "method": "GET",
        "path": "/orders/v0/orders/{orderId}",
        "dataElements": ["buyerInfo", "shippingAddress"]
    }]
}
```

Example : request a Restricted Data Token to retrieve a specific report document containing PII data

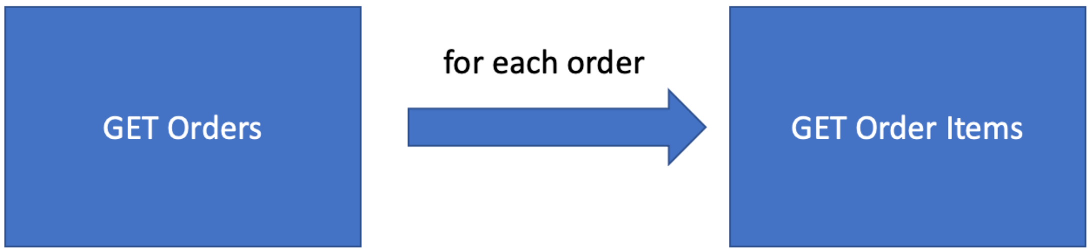
```
POST /tokens/2021-03-01/restrictedDataToken
{
  "restrictedResources": [
    {
      "method": "GET",
      "path": "/reports/2021-06-30/documents/amzn1.spdoc.1.4.eu.abcd1234..."
    }
  ]
}
```

Example : response from a createRestrictedDataToken request

```
{
    "expiresIn": 3600,
    "restrictedDataToken": "Atz.sprdt|ABCDeIFZID/WQ3YIx9..."
}
```

ORDERS API

Order Data can be retrieved synchronously by calling the Orders API. To retrieve your Order Data, periodically call **getOrders** operation (e.g. every 5 minutes, inline with the API Usage Plan). For all returned orders, call the **getOrderItems** operation to retrieve the order item details.



GET ORDERS

Use the getOrders operation to retrieve orders.

Returns orders created or updated during the time frame indicated by your specified parameters. You can also apply a range of filtering criteria to narrow the list of orders returned. If **NextToken** is present, that will be used to retrieve the orders instead of other criteria.

To retrieve PII data returned (e.g. Buyer Name), pass in the Restricted Data Token as the **x-amz-access-token** header. If you don't require PII data, you can pass in the Access Token instead.

Below is a subset of available parameters most relevant to BOPIS. For the full list, see the documentation here.

Parameter	Description	Type	Example	Required
CreatedAfter	A date used for selecting orders created after (or at) a specified time. Only orders placed after the specified time are returned. Either the CreatedAfter parameter or the LastUpdatedAfter parameter is required. Both cannot be empty. The date must be in ISO 8601 format.	string	2022-05-20T12:45:00.000Z	No
OrderStatuses	A list of `OrderStatus` values used to filter the results.	< string > array	Unshipped	No
MarketplaceIds	A list of MarketplaceId values. Used to select orders that were placed in the specified marketplaces. See the Selling Partner API Developer Guide for a complete list of marketplaceId values.	< string > array	A1F83G8C2ARO7P	Yes
IsISPU	When true, this order is marked to be picked up from a store rather than delivered.	boolean	TRUE	No
StoreChainStoreId	The store chain store identifier. Linked to a specific store in a store chain.	string	c97b8f7a-9d5a-48ae-9f8a-c413e51x2y3z	No
MaxResultsPerPage	A number that indicates the maximum number of orders that can be returned per page. Value must be 1 - 100. Default 100.	integer	100	No
NextToken	A string token returned in the response of your previous request.	string		No

Notes :

- To avoid missing any orders, please follow the best practices described in this short video.
- For EU sellers requiring Billing Address data, use Reports API (instead of Orders API) and a report type of **GET_ORDER_REPORT_DATA_INVOICING** or **GET_FLAT_FILE_ORDER_REPORT_DATA_INVOICING.** Ensure that Billing Data fields are enabled via Seller Central for your reports.
- You can add query parameters to the API request, to retrieve just BOPIS orders (*isISPU = true*), or BOPIS orders for a particular store (*StoreChainStoreId = <one of your Supply Source IDs>*)
- **GetOrders** response includes the **IsISPU** flag (to determine if an order is BOPIS), but not the **StoreChainStoreId** (to identify which of your stores the order is for). You must call **GetOrderItems** to retrieve the **StoreChainStoreId**.
- The **ShippingAddress** is the physical address for your pick-up location (i.e. your store address)

Usage Plan:

You can make up to 0.0167 requests per second for the getOrders operation, with a burst capacity of 20.

Rate (requests per second)	Burst
0.0167	20

Example Get Orders request to retrieve Unshipped Orders created after 2023-04-06, from the UK Marketplace

```
GET /orders/v0/orders?CreatedAfter=2023-04-06T00:00:00.000Z
&MarketplaceIds=A1F83G8C2ARO7P&
&OrderStatuses=Unshipped
```

Example Get Orders response for a BOPIS Order, using a Restricted Data Token

```
{
    "payload": {
        "Orders": [{
                "BuyerInfo": {
                    "BuyerEmail": "xf38cdr7abcdefg@marketplace.amazon.co.uk",
                    "BuyerName": "John Smith"
                },
                "AmazonOrderId": "202-6188802-1234567",
                "EarliestDeliveryDate": "2023-01-28T14:30:00Z",
                "EarliestShipDate": "2023-01-23T14:47:00Z",
                "SalesChannel": "Amazon.co.uk",
                "AutomatedShippingSettings": {
                    "HasAutomatedShippingSettings": false
                },
                "OrderStatus": "Unshipped",
                "NumberOfItemsShipped": 1,
                "OrderType": "StandardOrder",
                "IsPremiumOrder": false,
                "IsPrime": false,
                "FulfillmentChannel": "MFN",
                "NumberOfItemsUnshipped": 0,
                "HasRegulatedItems": false,
                "IsReplacementOrder": "false",
                "IsSoldByAB": false,
                "LatestShipDate": "2023-01-23T14:47:00Z",
                "ShipServiceLevel": "Inst-ISPU",
                "DefaultShipFromLocationAddress": {
                    "StateOrRegion": "Berkshire",
                    "AddressLine1": "null",
                    "PostalCode": "SL1 1YY",
                    "City": "Slough",
                    "CountryCode": "GB",
                    "Name": "null"
                },
                "IsISPU": true,
                "MarketplaceId": "A1F83G8C2ARO7P",
                "LatestDeliveryDate": "2023-01-28T14:30:00Z",
                "PurchaseDate": "2023-01-23T11:48:33Z",
                "ShippingAddress": {
                    "AddressLine3": "",
                    "AddressLine2": "Shoreditch",
                    "AddressLine1": "1 Principal Place",
                    "PostalCode": "EC1A 2FD",
                    "City": "London",
                    "CountryCode": "GB",
                    "Name": "Example Store"
                },
                "IsAccessPointOrder": false,
                "PaymentMethod": "Other",
                "IsBusinessOrder": false,
                "OrderTotal": {
                    "CurrencyCode": "GBP",
                    "Amount": "1.00"
                },
                "PaymentMethodDetails": [
                    "Standard"
                ],
                "IsGlobalExpressEnabled": false,
                "LastUpdateDate": "2023-01-23T16:56:44Z",
                "ShipmentServiceLevelCategory": "Standard"
            }
        ],
        "CreatedBefore": "2023-04-05T16:47:34Z"
    }
}
```

GET ORDER ITEMS

Use the getOrderItems operation to retrieve details on the items within an order.

Returns detailed order item information for the order that you specify, including the **StoreChainStoreId** which represents the Supply Source ID for the store that the order is to be picked up from.

Note, when an order is in the **Pending** state (the order has been placed but Amazon has not yet validated it), the **getOrderItems** operation does not return information about pricing, taxes, shipping charges, gift status or promotions for the order items in the order. After an order leaves the **Pending** state (this occurs once Amazon has validated the order) and enters the Unshipped, Partially Shipped, or Shipped state, the **getOrderItems** operation returns information about pricing, taxes, shipping charges, gift status and promotions for the order items in the order.

Usage Plan:

You can make up to 0.5 requests per second for the getOrderItems operation, with a burst capacity of 30.

Rate (requests per second)	Burst
0.5	30

Example Get Orders response for a BOPIS Order, using a Restricted Data Token

```
{
    "payload": {
        "OrderItems": [{
            "ProductInfo": {
                "NumberOfItems": "1"
            },
            "BuyerInfo": {},
            "ItemTax": {
                "CurrencyCode": "GBP",
                "Amount": "0.00"
            },
            "QuantityShipped": 1,
            "StoreChainStoreId": "d695d132-b9a0-4570-a582-d242d4a1b2c3",
            "BuyerRequestedCancel": {
                "IsBuyerRequestedCancel": "false",
                "BuyerCancelReason": ""
            },
            "ItemPrice": {
                "CurrencyCode": "GBP",
                "Amount": "1.00"
            },
            "ASIN": "B00RKSJ2BS",
            "SellerSKU": "product-10001",
            "Title": "Example Product",
            "IsGift": "false",
            "ConditionSubtypeId": "New",
            "IsTransparency": false,
            "QuantityOrdered": 1,
            "PromotionDiscountTax": {
                "CurrencyCode": "GBP",
                "Amount": "0.00"
            },
            "ConditionId": "New",
            "PromotionDiscount": {
                "CurrencyCode": "GBP",
                "Amount": "0.00"
            },
            "OrderItemId": "34494750123456"
        }],
        "AmazonOrderId": "202-2484992-1234567"
    }
}
```

ORDER REPORTS

You can use Order Reports as an alternative to the Orders API.   Access to Order Reports is provided via the Reports API. There are different types of order reports that you can request, details of which can be found here.

Only the following Order Report types include the additional fields (once enabled via Seller Central) required to identify an order as a BOPIS order:

- GET_ORDER_REPORT_DATA_SHIPPING (NA and EU)
- GET_FLAT_FILE_ORDER_REPORT_DATA_SHIPPING (NA and EU)
- GET_ORDER_REPORT_DATA_INVOICING (EU Only)
- GET_FLAT_FILE_ORDER_REPORT_DATA_INVOICING (EU Only)

Enabling additional fields for your Order Reports

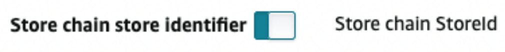
To enable additional fields, login to **Seller Central**, then navigate to **Orders → Order Reports.** Click on the "**Add or remove order report columns**" on the right-hand side of the page.

From here, you can enable the following fields / columns :

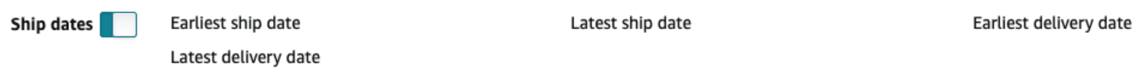
Is In Store Pick Up : this will include a boolean value in your reports indicating if the order is for in store pickup. In the reports, the field is named **IsInStorePickup** (XML) or **is-ispu-order** (Flat File).

Is In Store Pick Up   Is In-store pickup order?

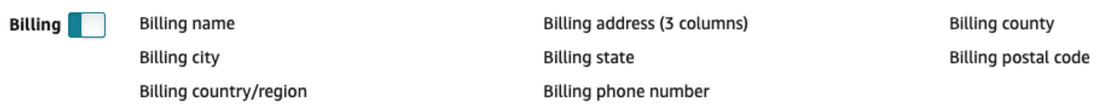
Store chain store identifier : this will include the Supply Source ID, indicating which of your stores this order should be routed to. In the reports, the field is named **StoreChainStoreId** (XML) or **store-chain-store-id** (Flat File).


Store chain store identifier
Store chain StoreId

Ship dates : Earliest ship date and Latest ship date contain the same value which provide the date/time by which a Click & Collect order must be marked as "Ready for Pickup".   Earliest delivery date and Latest delivery date contain the same value which provides the last date/time by when a buyer must pick up the order.   Use this information to determine when you should refund an BOPIS order not picked up.


Ship dates
Earliest ship date | Latest ship date | Earliest delivery date
Latest delivery date

Billing (EU Only) : If you require the billing address of the buyer, then enable the Billing fields / columns.   Note that the shipping address provided in order details is the address of your Click & Collect location.   Billing information is provided only in **GET_ORDER_REPORT_DATA_INVOICING** and **GET_FLAT_FILE_ORDER_REPORT_DATA_INVOICING**


Billing
Billing name | Billing address (3 columns) | Billing county
Billing city | Billing state | Billing postal code
Billing country/region | Billing phone number

Requesting an Order Report

There are 2 approaches to requesting Order Reports from Amazon :

1. You submit a new report request every time you want a report (or using your own scheduling controller or logic, or ad-hoc).   Full documentation available here
2. You schedule a report to be generated periodically (e.g. every 15 minutes) via Reports API.   Full documentation available here

Option 1 : Making a new report request

Call **createReports** operation to request a report, specifying the report type, a date range for the period to report on and one or more Amazon Marketplace IDs.   **Requesting a report is an asynchronous process.**   When you submit your request, a **reportId** will be returned, which you can use to query the status of your report request

Usage Plan for **createReports**:

Rate (requests per second)	Burst
0.0.167	15

Example : Request an Order Report (Shipping Data) covering a specific date/time range

```
POST /reports/2021-06-30/reports
{
    "reportType": "GET_FLAT_FILE_ORDER_REPORT_DATA_SHIPPING",
    "dataStartTime": "2023-01-01T09:00:00.000Z",
    "dataEndTime": "2023-01-01T09:15:00.000Z",
    "marketplaceIds":["ATVPDKIKX0DER"]
}
```

Example Response

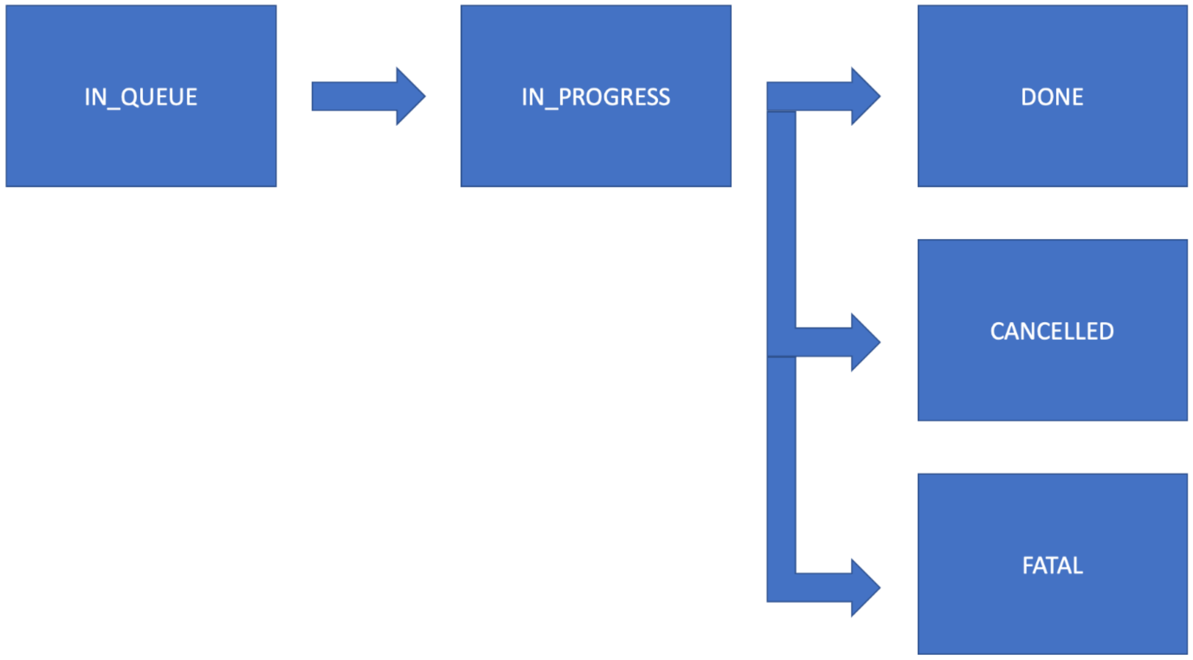
```
{
     "reportId": "128086012345"
}
```

Check status of your Report Request

To check when a report request has been completed by Amazon, you can monitor the status of your report request.   There are 3 ways to do this :

- Using the **Reports API**, periodically check the status of your report request using the **getReport** operation.
- Using the **Reports API**, Periodically check for any new reports using the **getReports** operation and applying a query filter.
- Subscribe to updates via the **Notifications API** using Amazon Simple Queue Service (SQS), to receive **REPORT_PROCESSING_FINISHED** notifications.

A request for an order report will initially be queued (**IN_QUEUE** status).   Amazon will then begin processing the request (**IN_PROGRESS**).   If there are orders to be reported, then the report is completed (**DONE**).   No report is generated if there are no orders (**CANCELLED**) or if the request failed (**FATAL**).



Periodically checking the status of your report request

To periodically check the report status, call **getReport** operation with your report ID.

```
GET /reports/2021-06-30/reports/<your-report-id>
```

Check for the **processingStatus** in the response.   Once **DONE**, a **reportDocumentId** will be included in the response.

Usage Plan for **getReport**:

Rate (requests per second)	Burst
2	15

Example Response

```
{
    "reportType": "GET_FLAT_FILE_ORDER_REPORT_DATA_SHIPPING",
    "processingEndTime": "2023-01-01T09:23:54+00:00",
    "processingStatus": "DONE",
    "marketplaceIds": [
        "A1F83G8C2ARO7P"
    ],
    "reportDocumentId": "amzn1.spdoc.1.4.eu.abcd1234-cc28-42f6-ae78-10e99415a55b...",
    "reportId": "128086012345",
    "dataEndTime": "2023-01-01T09:15:00+00:00",
    "createdTime": "2023-01-01T09:23:21+00:00",
    "processingStartTime": "2023-01-01T09:23:43+00:00",
    "dataStartTime": "2023-01-01T09:00:00+00:00"
}
```

Use this to retrieve the Report Document URL via the **getReportDocument** operation.

Get the Report Document URL

To retrieve a report, you must request the location of the report to download it from.   Use the **getReportDocument** operation to do this.   The **getReportDocument** operation is considered a restricted operation only when a restricted report type is specified. See the list of restricted report types here.   In such cases, you must provide a Restricted Data Token in the **x-amz-access-token** header.

```
GET /reports/2021-06-30/documents/<your-report-document-id>
```

Usage Plan for **getReportDocument**:

Rate (requests per second)	Burst
0.0167	15

When calling the **createRestrictedDataToken** operation to get a Restricted Data Token for the **getReportDocument** operation, the specified restricted resource can contain only a specific path, not a generic path.   For definitions, see Terminology.

```
POST /tokens/2021-03-01/restrictedDataToken
{
    "restrictedResources": [{
        "method": "GET",
        "path": "/reports/2021-06-30/documents/amzn1.spdoc.1.4.eu.abcd1234..."
    }]
}
```

Example Response

```
{
    "reportDocumentId": "amzn1.spdoc.1.4.eu.abcd1234-cc28-42f6-ae78-10e99415a55b...",
    "url": "https://tortuga-prod-eu.s3-eu-west-1.amazonaws.com/%2FNinetyDays/..."
}
```

Extract the URL from the response and make a GET request to this URL to download the report.   Further details on how to retrieve a report are available here.

Option 2 : Scheduling an Order Report

If you prefer to let Amazon schedule the report generation, call the **createReportSchedule** operation, with the report type, period (frequency) and one or more marketplace IDs.

Usage Plan for **createReportSchedule**:

Rate (requests per second)	Burst
0.0222	10

Example : Schedule an Order Report (Shipping Data) every 15 minutes

```
POST /reports/2021-06-30/schedules
{
    "reportType": "GET_FLAT_FILE_ORDER_REPORT_DATA_SHIPPING",
    "period": "PT15M",
    "marketplaceIds":["ATVPDKIKX0DER"]
}
```

Example : (EU Specific) Schedule a Order Report (Invoicing Data) every 15 minutes, which can include Billing Address data

```
POST /reports/2021-06-30/schedules
{
    "reportType": "GET_FLAT_FILE_ORDER_REPORT_DATA_INVOICING",
    "period": "PT15M",
    "marketplaceIds":["ATVPDKIKX0DER"]
}
```

Retrieving a scheduled Order Report

You can periodically run (e.g. every 5 or 10 minutes) a query for available reports using the **getReports** operation and providing specific query filters.   Use this to retrieve newly created reports, since your previous run, to monitor for new reports to download and process.

Usage Plan for **getReports**:

Rate (requests per second)	Burst
0.0222	10

Below is a subset of available parameters most relevant to BOPIS. For the full list, see the documentation here.

Parameter	Description	Type	Example	Required
reportTypes	A list of report types used to filter reports. Refer to Report Type Values for more information. When reportTypes is provided, the other filter parameters (processingStatuses, marketplaceIds, createdSince, createdUntil) and pageSize may also be provided. Either reportTypes or nextToken is required. **Min count** : 1 **Max count** : 10	< string > array	GET_FLAT_FILE_ORDER_REPORT_DATA_SHIPPING	No
processingStatuses	A list of processing statuses used to filter reports. **Min count** : 1	< enum (ProcessingStatuses) > array	DONE	No
marketplaceIds	A list of MarketplaceId values. Used to select orders that were placed in the specified marketplaces. See the Selling Partner API Developer Guide for a complete list of marketplaceId values.	< string > array	A1F83G8C2ARO7P	No
createdSince	The earliest report creation date and time for reports to include in the response, in ISO 8601 date time format. The default is 90 days ago. Reports are retained for a maximum of 90 days.	string (date-time)	2023-04-01T00:00:01.000Z	No
pageSize	The maximum number of reports to return in a single call. **Default** : 10 **Minimum** : 1 **Maximum** : 100	integer	10	No
nextToken	A string token returned in the response of your previous request.	string		No

Example query to retrieve Order Reports (Shipping Data), processed (DONE) since 2023-04-01

```
GET /reports/2021-06-30/reports?
    reportTypes=GET_FLAT_FILE_ORDER_REPORT_DATA_SHIPPING&
    processingStatuses=DONE&
    marketplaceIds=A1F83G8C2ARO7P&
```

```
      createdSince=2023-04-01T00:00:01.000Z
```

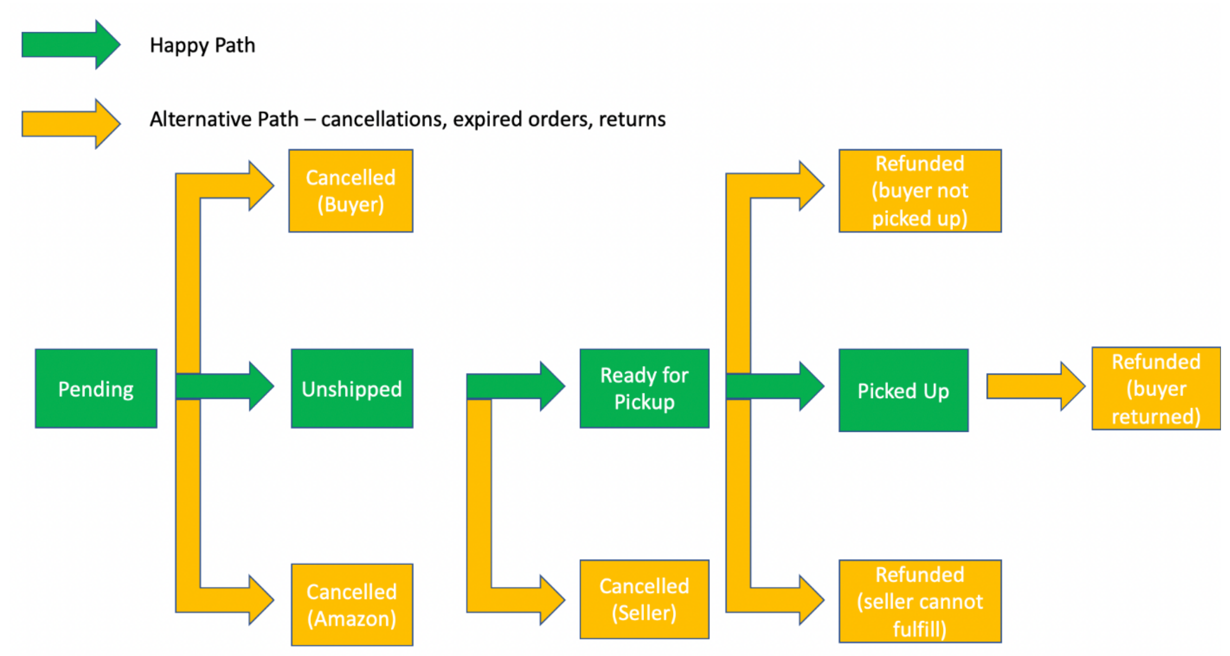
Example response to query

```
{
    "reports": [{
            "reportType": "GET_FLAT_FILE_ORDER_REPORT_DATA_SHIPPING",
            "processingEndTime": "2023-04-24T11:04:06+00:00",
            "processingStatus": "DONE",
            "marketplaceIds": [
                "A1F83G8C2ARO7P"
            ],
            "reportDocumentId": "amzn1.spdoc.1.4.eu.8b757c75...",
            "reportId": "128166012345",
            "dataEndTime": "2023-03-30T21:00:00+00:00",
            "createdTime": "2023-04-24T11:03:47+00:00",
            "processingStartTime": "2023-04-24T11:03:54+00:00",
            "dataStartTime": "2023-01-01T00:00:01+00:00"
        },
        {
            "reportType": "GET_FLAT_FILE_ORDER_REPORT_DATA_SHIPPING",
            "processingEndTime": "2023-04-24T10:57:16+00:00",
            "processingStatus": "DONE",
            "marketplaceIds": [
                "A1F83G8C2ARO7P"
            ],
            "reportDocumentId": "amzn1.spdoc.1.4.eu.053cffad...",
            "reportId": "128159012345",
            "dataEndTime": "2023-03-30T21:00:00+00:00",
            "createdTime": "2023-04-24T10:56:58+00:00",
            "processingStartTime": "2023-04-24T10:57:05+00:00",
            "dataStartTime": "2023-01-01T00:00:01+00:00"
        }
        ...
    ],
    "nextToken": "w4oRvk2mfneudbT..."
}
```

Follow the same steps above (“**Get the Report Document URL”)** to retrieve the Report Document URL from where you can download your Order Report.

PROCESSING ORDERS

Once you have retrieved new order details from Amazon, you must prepare and process these Orders. For BOPIS orders, this is managed through the **Orders API**. The following diagram shows the different order statuses for a BOPIS order.


Happy Path
Alternative Path – cancellations, expired orders, returns
Pending
Unshipped
Cancelled (Buyer)
Cancelled (Amazon)
Ready for Pickup
Cancelled (Seller)
Refunded (buyer not picked up)
Picked Up
Refunded (seller cannot fulfill)
Refunded (buyer returned)

Pending

When an order is placed by a buyer, Amazon will hold the order in a **Pending** state for 30 minutes (average time). During this time, the buyer or Amazon can cancel the order. You should not try to process an order while it is still in the **Pending** state.

Unshipped

After 30 minutes (average time), the order will be moved from **Pending** to **Unshipped** state by Amazon. At this point all the order details are available to retrieve.

Ready for Pickup

Once you have prepared the order, then you must mark it "Ready for Pickup". Use the **updateShipmentStatus** operation for this, setting the **shipmentStatus** to **ReadyForPickup.**

You must prepare the order and mark it "Ready for Pickup" within the Handling Time period (e.g. 90 minutes) that you have configured on your store (Supply Source). If you cannot fufill the order (e.g. no inventory), then you must cancel the order (see section below - "Alternative Path - Cancel Order"). When an order is marked "Ready for Pickup", the buyer is charged at this point.

```
POST /orders/v0/orders/{order-id}/shipment
{
    "marketplaceId": "A1F83G8C2ARO7P",
    "shipmentStatus": "ReadyForPickup"
}
```

Picked Up

When a buyer picks up their order in your store, you must mark the order as "Picked Up". Use the **updateShipmentStatus** operation, setting the **shipmentStatus** to **PickedUp**

```
POST /orders/v0/orders/{order-id}/shipment
{
    "marketplaceId": "A1F83G8C2ARO7P",
    "shipmentStatus": "PickedUp"
}
```

Alternative Path - Cancelled (Buyer)

A buyer can only cancel an order within 30 minutes of placing the order and while it is still in the Pending state. The buyer can immediately cancel the order during this 30 minute window. No action is required by the seller. The order should not be fulfilled. After 30 minutes, the buyer can only request a cancellation by the seller. See **Cancelled (Seller)** below for more detail on this.

Alternative Path - Cancelled (Amazon)

Amazon may determine that the order is not valid (e.g. fraud) and cancel the order during the Pending state. No action is required by the seller. The order should not be fulfilled.

Alternative Path - Cancel (Seller)

Should you receive an order that you cannot fulfill, you must cancel the order within the handling time period configured for your Supply Source (e.g. 90 minutes since order was placed). Orders can be cancelled through Seller Central or by submitting an Order Acknowledgement Feed (**POST_ORDER_ACKNOWLEDGEMENT_DATA** or **POST_FLAT_FILE_ORDER_ACKNOWLEDGEMENT_DATA**) via the **Feeds API**.

You may also need to cancel an order if the buyer has requested a cancellation. This can occur if the order is more than 30 minutes old but still **Unshipped**. If you cancel an order without having received a request from the buyer to do so, then this will counted as a Pre-fulfilment Cancellation in your Seller Metrics. The buyer must request the cancellation through their Amazon account in this situation.

Partial order cancellation is not supported. Use a **CancelReason** of **NoInventory** if you are unable to fufill the order due to lack of inventory. If the buyer has requested for the order to be cancelled, use a **CancelReason** of **BuyerCanceled.** For a full list of **CancelReasons**, refer to the OrderAcknowledgement.xsd file here.

Example POST_ORDER_ACKNOWLEDGEMENT_DATA XML feed for Order Cancellation

```
<?xml version="1.0" encoding="UTF-8"?>
<AmazonEnvelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceS
 <Header>
     <DocumentVersion>1.01</DocumentVersion>
     <MerchantIdentifier>{Merchant-Id}</MerchantIdentifier>
 </Header>
 <MessageType>OrderAcknowledgement</MessageType>
    <Message>
        <MessageID>1</MessageID>
        <OrderAcknowledgement>
            <AmazonOrderID>{Amazon-Order-id}</AmazonOrderID>
            <StatusCode>Failure</StatusCode>
            <Item>
                <AmazonOrderItemCode>{AmazonOrderItemCode}</AmazonOrderItemCode>
                <CancelReason>{REASON}</CancelReason>
            </Item>
        </OrderAcknowledgement>
    </Message>
</AmazonEnvelope>
```

Alternative Path - Refund Order (not picked up)

If an order which is Ready for Pickup has not been picked by the end of the Inventory Hold Period (5 days), then you must refund this order.   This is **not** done automatically by Amazon.   Orders can be refunded through Seller Central or by submitting an Order Adjustment Feed (**POST_PAYMENT_ADJUSTMENT_DATA** or **POST_FLAT_FILE_PAYMENT_ADJUSTMENT_DATA**) via the **Feeds API**.   An Adjustment Reason of **CustomerCancel** should be used.

Example POST_PAYMENT_ADJUSTMENT_DATA XML feed for Order Refund

```
<?xml version="1.0" encoding="utf-8"?>
<AmazonEnvelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceS
    <Header>
        <DocumentVersion>1.01</DocumentVersion>
        <MerchantIdentifier>{Merchant-Id}</MerchantIdentifier>
    </Header>
    <MessageType>OrderAdjustment</MessageType>
    <Message>
        <MessageID>1</MessageID>
        <OrderAdjustment>
            <AmazonOrderID>{Amazon-Order-id}</AmazonOrderID>
            <AdjustedItem>
                <AmazonOrderItemCode>{AmazonOrderItemCode}</AmazonOrderItemCode>
                <AdjustmentReason>{REASON}</AdjustmentReason>
                <ItemPriceAdjustments>
                    <Component>
                        <Type>Principal</Type>
                        <Amount currency="USD">999.99</Amount>
                    </Component>
                    <Component>
                        <Type>Shipping</Type>
                        <Amount currency="USD">3.49</Amount>
                    </Component>
                    <Component>
                        <Type>Tax</Type>
                        <Amount currency="USD">70</Amount>
                    </Component>
                    <Component>
                        <Type>Shipping Tax</Type>
                        <Amount currency="USD">0.24</Amount>
                    </Component>
                </ItemPriceAdjustments>
            </AdjustedItem>
        </OrderAdjustment>
    </Message>
</AmazonEnvelope>
```

Alternative Path - Refund Order (returned)

Buyers may choose to return an order via post to your returns address defined in your Seller Central account, or in-person directly to your store. As above, such orders can be refunded through Seller Central or by submitting an Order Adjustment Feed (**POST_PAYMENT_ADJUSTMENT_DATA** or **POST_FLAT_FILE_PAYMENT_ADJUSTMENT_DATA**) via the **Feeds API**.

Notifications

The **Selling Partner API for Notifications** lets you subscribe to notifications from Amazon that are relevant to your business. Using this API, you can create a destination to receive notifications, subscribe to notifications, delete notification subscriptions, and more. Instead of polling for information, your application can receive information directly from Amazon when an event triggers a notification to which you are subscribed.

Depending on the type, notifications can be received through **Amazon EventBridge** or **Amazon Simple Queue Service (Amazon SQS)**:

- **Amazon EventBridge**: A serverless event bus that connects application data from your own applications, integrated Software-as-a-Service (SaaS) applications, and AWS services. For more information, refer to Amazon EventBridge.
- **Amazon Simple Queue Service (Amazon SQS)**: A fully managed message queuing service for microservices, distributed systems, and serverless applications. For more information, refer to Amazon Simple Queue Service.

The following Notifications can be useful for your BOPIS integration. These are received through **Amazon SQS:**

ORDER_STATUS_CHANGE - sent whenever there is a change in the status of new or existing order availability.

FEED_PROCESSING_FINISHED - sent whenever any feed submitted using the Selling Partner API for Feeds reaches a feed processing status of DONE, CANCELLED, or FATAL.

REPORT_PROCESSING_FINISHED - sent whenever any report that you have requested using the the Selling Partner API for Reports reaches a report processing status of DONE, CANCELLED, or FATAL.

You can find the complete list of available notification types here (Amazon SQS) and here (Amazon EventBridge).

Setting up to receive Notifications (Amazon SQS)

Pre-requisites : Login to your AWS Management Console, navigate to the Amazon SQS service, then create a Standard queue. If you are not familiar with Amazon SQS, this AWS SQS Getting Started guide provides more detail.

Step 1 : Grant the Selling Partner API permission to write to your Amazon SQS queue. Follow the steps here to do this.

Step 2 : Call the **createDestination** operation to create an Amazon Simple Queue Service (SQS) destination. Follow the steps here to complete this. This tells the Selling Partner API about your SQS queue. Note that **createDestination** is a Grantless Operation.

Example createDestination operation

```
POST /notifications/v1/destinations
{
    "name": "MyDestinationName",
    "resourceSpecification": {
        "sqs": {
            "arn": "arn:aws:sqs:eu-west-1:123456780000:MySQSQueueName"
        }
    }
}
```

Example createDestination response

```
{
    "payload": {
        "resource": {
            "sqs": {
                "arn": "arn:aws:sqs:eu-west-1:123456780000:MySQSQueueName"
            },
            "eventBridge": null
        },
        "destinationId": "a9a36dd9-6458-4329-9f42-ed4ecxxyyzzz",
        "name": "MyDestinationName"
    }
}
```

Step 3 : Create a subscription to a notification type, to be delivered to the destination that you created in the previous step, using the **createSubscription** operation.

Example createSubscription operation for FEED_PROCESSING_FINISHED notifications

```
POST /notifications/v1/subscriptions/FEED_PROCESSING_FINISHED
{
  "payloadVersion": "1.0",
  "destinationId": "a9a36dd9-6458-4329-9f42-ed4ecxxyyzzz",
}
```

Example createSubscription response

```
{
    "payload": {
        "subscriptionId": "0c279d70-f5df-4b5b-985e-xy8fdaabbccc",
        "destinationId": "a9a36dd9-6458-4329-9f42-ed4ecxxyyzzz",
        "payloadVersion": "1.0"
    }
}
```

Notification Structure

Selling Partner notifications are in JSON format. Each notification contains a **Payload** object, which contains the actionable data of the notification. Notification Type, in combination with **PayloadVersion**, determines the structure of the **Payload** object.

A Selling Partner notification with **NotificationVersion**=1.0 contain the following properties:

Object	Description	Type
NotificationVersion	The notification version. This controls the structure of the notification.	string
NotificationType	The notification type. **NotificationType**, combined with **PayloadVersion,** controls the structure of the **Payload object**.	string
PayloadVersion	The payload version. **PayloadVersion**, combined with **NotificationType,** controls the structure of the **Payload object**.	string
EventTime	The date and time (in UTC) that the event which triggered the notification occurred.	string
Payload	The actionable data of the notification. The structure of the **Payload** is determined by **NotificationType**, in combination with **PayloadVersion**.	JSON object For more information, see Notifications.
NotificationMetadata	The notification metadata. This includes the following objects: **ApplicationId** – The identifier for the application that uses the notifications. Type = string **SubscriptionId** - A unique identifier for the subscription which resulted in this notification. Type = string **PublishTime** - The date and time (in UTC) that the notification was sent. Type = string **NotificationId** - A unique identifier for this notification instance. Type = string	JSON object

For each order, you will receive an **ORDER_STATUS_CHANGE** notification for each order status change. When a new order is succesfully placed, an notification with OrderStatus of “**UpComing**” will be received, followed by another with a “**Pending**” OrderStatus, then an “**Unshipped**” OrderStatus. When you mark the order as “Ready for Pickup”, you will receive a notification with a “**Shipped**” OrderStatus. Examples shown below for “Pending” and “**Unshipped**” notifications :

Example “Pending” ORDER_STATUS_CHANGE notification

```
{
  "NotificationVersion" : "1.0",
  "NotificationType" : "ORDER_STATUS_CHANGE",
  "PayloadVersion" : "1.0",
  "EventTime" : "2023-04-27T13:13:14.325Z",
  "Payload" : {
    "OrderStatusChangeNotification" : {
      "SellerId" : "AXXXXXXXXXXXXX",
      "MarketplaceId" : "A1F83G8C2ARO7P",
      "AmazonOrderId" : "202-0199662-1234567",
      "PurchaseDate" : 1682600949435,
      "OrderStatus" : "Pending",
      "DestinationPostalCode" : null,
      "SupplySourceId" : "d695d132-b9a0-4570-a582-ed4ecxxyyzzz",
      "OrderItemId" : "29084056211234:",
      "SellerSKU" : "product-123",
      "Quantity" : 1,
      "FulfillmentChannel" : "MFN"
    }
  },
  "NotificationMetadata" : {
    "ApplicationId" : "amzn1.sp.solution.3327cd53-cb71-4a0a-a80e-ed4ecxxyyzzz",
    "SubscriptionId" : "a400a588-ca35-45c8-9382-ed4ecxxyyzzz",
    "PublishTime" : "2023-04-27T13:13:14.801Z",
    "NotificationId" : "024ac7f3-5684-4484-968e-ed4ecxxyyzzz"
  }
}
```

Example “Unshipped” ORDER_STATUS_CHANGE notification

```
{
  "NotificationVersion" : "1.0",
  "NotificationType" : "ORDER_STATUS_CHANGE",
  "PayloadVersion" : "1.0",
  "EventTime" : "2023-04-27T13:38:59.229Z",
  "Payload" : {
    "OrderStatusChangeNotification" : {
      "SellerId" : "AXXXXXXXXXXXXX",
      "MarketplaceId" : "A1F83G8C2ARO7P",
      "AmazonOrderId" : "202-0199662-1234567",
      "PurchaseDate" : 1682600949435,
      "OrderStatus" : "Unshipped",
      "DestinationPostalCode" : null,
      "SupplySourceId" : "d695d132-b9a0-4570-a582-ed4ecxxyyzzz",
      "OrderItemId" : "29084056211234:",
      "SellerSKU" : "product-123",
      "Quantity" : 1,
      "FulfillmentChannel" : "MFN"
    }
  },
  "NotificationMetadata" : {
    "ApplicationId" : "amzn1.sp.solution.3327cd53-cb71-4a0a-a80e-ed4ecxxyyzzz",
    "SubscriptionId" : "a400a588-ca35-45c8-9382-ed4ecxxyyzzz",
    "PublishTime" : "2023-04-27T13:39:00.584Z",
    "NotificationId" : "22065c8e-bc6b-4a19-b8ea-ed4ecxxyyzzz"
  }
}
```