Specification: Seller Dynamics – API Documentation
Dated: 15/03/2021

Seller Dynamics – Order API

Important

The information contained within this document is confidential and intended for the use of Objective Associates Ltd only. Permission is required from Objective Associates Ltd prior to disclosure to other parties.

Table of Contents

1. Introduction

This document defines how to use the Seller Dynamics Application Programmers Interface (API) to retrieve and update order information held in the Seller Dynamics system.

2. Connection

The API specification is exposed in Web Service Definition Language (WSDL) and this provides the exposed schema and definitions required to interact with the Seller Dynamics system. The WSDL location is available upon request.

Although the service has been created in C#, due to the autonomous nature of the methods used, the services should be able to be consumed by any language with SOAP libraries. Simply point your system at the WSDL and this should create the framework with which subsequent interactions can be made.

In order to use the API, you will need to pass in the encrypted data string and retailer id that were provided. If you do not have this information, simply contact us, and we will send this out to you. The encrypted data string is tied to your username and password, so if you change the password on this account we will need to re-generate the token for you again.

3. Throttling of Calls

Your calls to the Seller Dynamics API are subject to throttling limits. If your request is throttled then Seller Dynamics will return a HTTP error code 429 “Too many requests”. It is recommended that you code your application in such a way to cope with the HTTP error returned and retry the call later. The throttling limits are ample to allow you to use the API as recommended.

4. Interaction

Only a few method calls are required to interact with the API. Although coded in C#, any high-level programming language can be used.

4.1. Retrieving orders

To retrieve orders from the system, you need to have the encrypted data string and retailer id discussed in Section 2. There are 2 methods used to retrieve orders (GetCustomerOrders and GetCustomerOrdersExtended), each of which take the same six mandatory parameters which are described in the following table:

Parameter name	Parameter type	Description
encryptedLogin	String	This is the encrypted login, provided by your account manager.
retailerId	Guid	This is the account identifier, provided by your account manager.
orderType	Constants.OrderType	This value should specify one of the following constants: PENDING – orders that require to be processed. HISTORIC – orders which have been processed (i.e. dispatched or refunded). FBA – orders which Amazon process on your behalf within their fulfilment centres.
fromDate	DateTime	Orders will be returned which are created in Seller Dynamics on or after this date (see 4.1.4 Calculating Date Ranges).
toDate	DateTime	Orders will be returned which are created in Seller Dynamics on or before this date (see 4.1.4 Calculating Date Ranges).
pageNumber	Int	Indicates the page number being requested. The initial call should start at 1 and then subsequent calls should increment this value based on the data returned in the PaginationObject (see 4.1.3 Pagination).

4.1.1. GetCustomerOrders

Calls to GetCustomerOrders will return the following response:

Parameter name	Parameter type	Description
Pagination	PaginationObject	Returns information about the total number of records available and the number of records returned (see 4.1.3 Pagination).
IsError	Boolean	Specifies whether an error was encountered while processing the request.
ErrorMessage	String	If IsError is true then details about the error is returned in this field.
Customers	List<CustomerOrder>	A list of customer objects meeting the criteria of the request.

Each CustomerOrder object present within the Customers element will contain the following fields:

Parameter name	Type	Description
CustomerMarketplaceIdentifier	String	The marketplace specific identifier for the customer.
CustomerName	String	The customer's name.
CustomerEmail	String	The customer's email address.
CustomerPhone	String	The customer's phone number.
ShippingAddress1	String	Customer shipping address line 1.
ShippingAddress2	String	Customer shipping address line 2.
ShippingAddress3	String	Customer shipping address line 3.
ShippingAddressTown	String	Customer shipping town.
ShippingAddressRegion	String	Customer shipping region.
ShippingAddressPostZIPCode	String	Customer shipping postcode.
ShippingAddressCountry	String	Customer shipping country.
SKU	String	Stock keeping unit value for the product purchased.
EANBarCode	String	The barcode of the item.
ItemTitle	String	The title of the product purchased.
Quantity	Int	The quantity of products purchased.
MarketPlaceName	String	The marketplace where the purchase originated.
MarketPlaceOrderID	String	The marketplace order id.
OrderStatusID	Int	The Seller Dynamics status for the order line. Orders that are "Pending", i.e. those that are not yet processed can have the following numeric values: 1: New 2: In Stock 3: Pending PO 4: On Order 5: Reorder 9: Printed Orders which have been actioned (for example, those that are dispatched or refunded), will have one of the following numeric values: 6: Dispatched 10: Refunded 11: Refunding 12: Dispatching
Status	String	The Seller Dynamics text status of the order line based on the values above, for example New or In Stock.
CostPrice	Decimal	The cost price of the order line based on the product price at the time of purchase multiplied by the quantity purchased.

PricePaid	String	The price paid for the order line in the retailer's currency.
PostagePaid	String	The postage paid for the order line in the retailer's currency.
TaxPaid	Decimal	The amount of tax paid for the order line in the retailer's currency.
ShippingTaxPaid	Decimal	The amount of postage tax paid for the order line in the retailer's currency.
TotalTaxPaid	Decimal	The total tax paid for the order line.
TaxRate	Decimal	The tax rate applied to the order line.
IsPaid	Boolean	Indicates whether payment has been received.
ShippingMethodRequested	String	The shipping method requested on the marketplace.
PostageMethodName	String	The name of the Seller Dynamics assigned postage method.
PurchaseOrderNumber	String	The supplier purchase order number of an item.
PurchaseDate	DateTime	Date of purchase.
ItemShipDate	DateTime	The date the item was shipped.
TrackingCode	String	The courier tracking code associated with the order.
GoodsPurchasedID	Guid	The Seller Dynamics Guid for the order line.
SupplierID	Guid	The Seller Dynamics Guid for the item supplier.
PurchaseID	Guid	The SellerDynamics Guid for the entire order.
IsError	Boolean	Defines whether an error was encountered.
ErrorMessage	String	The error message encountered.
RecordVersion	Byte[]	A byte array used for time stamping.

4.1.2. GetCustomerOrdersExtended

Calls to GetCustomerOrdersExtended will return the following response:

Parameter name	**Parameter type**	**Description**
Pagination	PaginationObject	Returns information about the total number of records available and the number of records returned (see 4.1.3 Pagination).
IsError	Boolean	Specifies whether an error was encountered while processing the request.
ErrorMessage	String	If IsError is true then details about the error is returned in this field.
Customers	List<CustomerOrderExtended>	A list of customer objects meeting the criteria of the request.

Each CustomerOrderExtended object will contain all of fields returned by GetCustomerOrders (see 4.1.1) and additionally the following:

Parameter name	Type	Description
BillingAddress1	String	Customer billing address line 1.
BillingAddress2	String	Customer billing address line 2.
BillingAddress3	String	Customer billing address line 3.
BillingAddressTown	String	Customer billing town.
BillingAddressRegion	String	Customer billing region.
BillingAddressPostZIPCode	String	Customer billing postcode.
BillingAddressCountry	String	Customer billing country.
OrderPostagePaid	Decimal	The total postage paid for the order in the retailer's currency. This will be the sum of postage paid on all order lines.
CurrencyCode	String	The ISO code for the currency used to purchase the order (e.g. GBP, EUR, etc).
ConvertedPricePaid	Decimal	The price paid for the order line in the marketplace currency.
ConvertedTaxPaid	Decimal	The amount of postage tax paid for the order line in the marketplace currency.
ConvertedPostagePaid	Decimal	The postage paid for the order line in the marketplace currency.
ConvertedShippingTaxPaid	Decimal	The amount of postage tax paid for the order line in the marketplace currency.
ConvertedOrderPostagePaid	Decimal	The total postage paid for the order in the marketplace currency. This will be the sum of postage paid on all order lines.
MarketPlaceOrderItemID	String	The marketplace order line id.
EbaySalesRecordNumber	String	If the order was placed on eBay then this field will contain the eBay Sales Record Number value.
PayPalTransactionID	String	The PayPal purchase transaction identifier (applicable to eBay orders only).
Flag	String	The general purpose Seller Dynamics value associated with the product being purchased.

4.1.3. Pagination

The Pagination object returned by calls to GetCustomerOrders and GetCustomerOrdersExtended provides the following parameters:

Parameter name	Type	Description
PageNumber	Int	The page number passed in the request (e.g. 1).
PageSize	Int	The number of orders returned in the response. Note that a maximum of 50 orders can be returned in a single call.
RecordsAffected	Int	The total number of orders available which meet the request criteria.

By using pagination effectively, you can ensure that all available records are downloaded. The following C# example demonstrates this:

```
int pageNo = 1;
int totalOrders = 0;
bool hasErrored = false;
bool hasFinished = false;
do
{
        // Request a page of orders.
        SellerDynamicsAPI.CustomerOrders response = client.GetCustomerOrders
        (EncryptedLogin, RetailerID, SellerDynamicsAPI.OrderType.PENDING, fromDate, toDate,
        pageNo);

        // Success?
        if (ordersResponse.IsError)
        {
                // No.
                hasErrored = true;
        }
        else
        {
                // Yes.
                if (ordersResponse.Customers.Count() > 0)
                {
                        // Process your orders here...
                }

                // More pages to request?
                totalOrders += ordersResponse.Pagination.PageSize;
                if (totalOrders >= ordersResponse.Pagination.RecordsAffected)
                {
                        // No.
                        hasFinished = true;
                }
                else
                {
                        // Yes.
                        pageNo++;
                }
        }
} while (!hasErrored && !hasFinished);
```

In the above example, we keep a note of the total orders processed. When this value is greater than or equal to the RecordsAffected property in the Pagination object, we know that we have received all orders specified by the date range in the request. If the total orders processed is less than the RecordsAffected value then we increment the page number passed in the next request. Notice also how the IsError flag is used to handle processing issues.

4.1.4. Calculating Date Ranges

We recommend that you call GetCustomerOrders and GetCustomerOrdersExtended every 15 to 30 minutes. The fromDate and toDate parameters should be calculated as follows:

Call	fromDate	toDate
1st	The earliest time at which an order could have been downloaded into Seller Dynamics. For example, if your account was enabled on the March 1st 2020 then set the first fromDate to 01/03/2020 00:00:00.	The current date and time (e.g. 05/03/2020 14:30:00).
2nd (15 minutes later)	The previous toDate (e.g. 05/03/2020 14:30:00).	The current date and time (e.g. 05/03/2020 14:45:00)

4.2. Updating orders

The previous section explained how to retrieve the orders from Seller Dynamics. In this section we discuss updating the status of orders. Pending orders can be updated to indicate that they have either been shipped or refunded. Both updates use the methods UpdateOrders and UpdateOrdersWithTracking.

4.2.1. The UpdateOrders Method

The UpdateOrders method takes four parameters, detailed below:

Parameter name	Parameter type	Description
encryptedLogin	String	This is the encrypted login, provided by your account manager.
retailerId	Guid	This is the account identifier, provided by your account manager.
orderIds	List<OrderUpdateItem>	This is the orders lines that you want to update
updateType	Constants.OrderUpdateType	The type of update you want to perform. This can be either SHIPPED or REFUNDED

The encryptedLogin and retailerId are the same as described previously. The updateType is used to apply the relevant update to all OrderUpdateItem objects contained in orderIds. Each OrderUpdateItem object supplied by the orderIds parameter may contain the following properties:

Parameter name	Parameter type	Description
GoodsPurchasedId	Guid	This is the unqiue identifier of the order line to be updated. This value is returned in the CustomerOrder and CustomerOrderExtended objects provided by GetCustomerOrders and GetCustomerOrdersExtended respectively.
RecordVersion	Byte[]	(not used)

4.2.2. The UpdateOrdersWithTracking Method

The UpdateOrdersWithTracking method takes four parameters, detailed below:

Parameter name	Parameter type	Description
encryptedLogin	String	This is the encrypted login, provided by your account manager.
retailerId	Guid	This is the account identifier, provided by your account manager.
orderIds	List<OrderUpdateItemWithTracking>	This is the orders lines that you want to update
updateType	Constants.OrderUpdateType	The type of update you want to perform. This can be either SHIPPED or REFUNDED

The encryptedLogin and retailerId are the same as described previously. The updateType is used to apply the relevant update to all OrderUpdateItem objects contained in orderIds. Each OrderUpdateItemWithTracking object supplied by the orderIds parameter contain the following properties:

Parameter name	Parameter type	Description
GoodsPurchasedId	Guid	This is the unqiue identifier of the order line to be updated. This value is returned in the CustomerOrder and CustomerOrderExtended objects provided by GetCustomerOrders and GetCustomerOrdersExtended respectively.
Courier	Enumerated value	This enumerated value allows you to specify one of several pre-defined couriers. If the required courier is unavailable, then specify 'Other' and use the 'OtherCourier' parameter to set the name of the courier. If a courier has not been used to dispatch the order, or the updateType parameter is REFUNDED then specify the value 'NotSet'.
OtherCourier	String	This value is used to specify the name of a third-party courier and will only be considered if the value 'Other' has been specified using the 'Courier' enumerated parameter. If the 'Courier' enumerated parameter specifies anything other than 'Other' or if the updateType parameter is REFUNDED then specify an empty string.
ShippingMethod	String	This value allows you to specify the shipping method used to dispatch the item. For example, if the Courier property was **Royal Mail**¸ then this value might be **Royal Mail Tracked 24**. Although this value is optional within Seller Dynamics, it is recommended that it is supplied where possible as it is mandatory on some marketplaces including Amazon.
TrackingNumber	String	This optional value is used to specify a courier tracking number. If a courier has not been used to dispatch the order, or if the courier's shipping service does not provided a tracking number, or if the updateType parameter is REFUNDED then specify an empty string.
RecordVersion	Byte[]	(not used)

4.2.3. Response to UpdateOrders and UpdateOrdersWithTracking

An UpdateOrdersResponse object is returned from both methods with the following values:

Parameter name	Parameter type	Description
IsError	Boolean	Used to check for errors encountered during the updating of orders
ErrorMessage	String	This stores the error message encountered during the updating of orders

4.2.4. General Guidelines

Under normal circumstances you should only call this method once for a give order line.

4.3. Placing Pending Orders "On Hold"

There may be instances when you want to place a pending order line on hold. A status of On Hold indicates that the order line should not be immediately processed. Within the Seller Dynamics User Interface, such an order would appear on the On Hold tab on the orders screen.
To place an order on hold, use the PlaceOrdersOnHold API call. This method takes the following parameters:

Parameter name	Parameter type	Description
encryptedLogin	String	This is the encrypted login, provided by your account manager.
retailerId	Guid	This is the account identifier, provided by your account manager.
ordersToPlaceOnHold	List<OrdersToPlaceOnHold>	A list of the orders to be placed on hold including optional notes.

The encryptedLogin and retailerId are the same as described previously. Each OrdersToPlaceOnHold object supplied by the ordersToPlaceOnHold parameter contain the following properties:

Parameter name	Parameter type	Description
GoodsPurchasedId	Guid	This is the unqiue identifier of the order line to be placed on hold. This value is returned in the CustomerOrder and CustomerOrderExtended objects provided by GetCustomerOrders and GetCustomerOrdersExtended respectively.
Notes	String	This optional value can include comments to be appended to the order line's notes section.