UltraCart Legacy API

Written By: UltraCart Support

Version 3.0 - Last Update 6/1/2011

Table of Contents

What this Document Covers

The primary focus of this document is to teach developers how to access information provided by UltraCart via an API. This interface is useful for ad-hoc extraction and manipulation of data within UltraCart. This document is targeted towards actual software developers that are going to understand the terminology associated with this technology.

Why is this API Legacy?

This API is called the Legacy API because it predates more modern application independent API languages such as SOAP, XML-RPC, AJAX/JSON, etc. While this API is still active and supported, it is limited to existing API functions described in this document. Newer APIs will be published using SOAP, etc.

Access Rights

Calls to the API happen through one of the user accounts configured for a merchant. We recommend that a separate API user be configured for the merchant account with only the API permission set. This will minimize the chance that someone changes the password for the user and breaks the program using the API. Configuration of users and permissions occurs under:

Configuration Menu -> Users

Basics of an API Call

The API allows programmers to access functions provided by UltraCart by making an HTTPS GET request with a set of parameters and receiving an XML response. The minimum set of parameters required by all function calls is:

- merchantId
- login
- password
- function

Some functions may require additional parameters. The additional parameter requirements and their format will be specified in the descriptions below. The response is a simple XML document that contains three fields:

- ResultCode
- ResultMessage
- ResultData

The result code is a numeric return code for the function call. Programmers should always look at this return code before manipulating the result data to see if the call was successful. A value of zero indicates success. All other values indicate some sort of failure. The ResultMessage field will contain additional details about the result code if available. The following table below shows the meaning of each numeric result code that can be returned by the API.

ResultCode	ResultMessage
0	Success
1	Unknown Error
2	Missing Parameter
3	Invalid Login
4	Inadequate Permissions
5	Invalid Parameter Value
6	Internal Error
7	HTTPS Required

The ResultData is always a CDATA object of the function result. The contents of this can vary from function call to function call. For instance some functions may return a simple string, XML document, BASE64 encoded image, etc.

All API calls are made to the same URL end-point with different parameters. These calls must be secured over HTTPS (SSL) in order to insure that the data provided is secured at all times. The URL to make HTTPS GET requests of is:

https://secure.ultracart.com/cgi-bin/UCApi

Attempts to make these calls over regular HTTP will result in an error code being returned.

Respecting System Resources

Please respect our system and only call the APIs as often as is reasonable. Make every effort to minimize the load placed upon the UltraCart system. For instance if you want to do analytic reporting on the order database then you should export the information on a daily basis in to a local database and report off it there. Please don't abuse the system by making excessive queries, too frequent of a query, etc. All access via APIs is logged in extreme detail. Merchants that abuse the system will have their right to access it via the API terminated.

Function Reference

Functions are divided into different functional groups: general, catalog management, item management, and order management methods. Each function will document the function name, a description of what the API does, as well as the additional method parameters required to make the call.

General Functions

CheckLogin

This function allows you to verify your credentials before calling other methods. It is a good idea for your application to validate a successful login before calling order methods and report meaningful errors to the user if the login fails. The most common caused for a failed login is a change in the password or IP authentication failing.

Additional Required Parameters

None

Result Data Format

No ResultData is returned with this API call. If the login is successful then the result code of 0 will be returned. See the ResultCode table under Basics of an API Call for other possible result codes.

Catalog Management Functions

AddCatalogGroup

This function call will create a new catalog group.

Additional Required Parameters

- code
- description
- groupTemplateOID
- hostOID
- imageURL
- itemTemplateOID
- parentOID
- title

code – each catalog group has a code associated with it. The code is appended to the path of the parent to create the new path for this group. You should not use spaces or other characters that make difficult urls in your code values.

description – The description of the group. This is commonly displayed as a paragraph of text on the catalog.

groupTemplateOID – the OID of the template to use when a group page is rendered. This should be obtained from a call to QueryCatalogTemplateList.

hostOID – the OID of the host that you are creating the catalog group in. This should be obtained from a call to QueryCatalogHostList.

imageURL – (deprecated) the URL to an image that should be associated with the catalog group. This should not be used. Instead merchants should use the integrated multimedia functionality available within the catalog group management.

itemTemplateOID – the OID of the template to use when an individual item page is rendered within this group. This should be obtained from a call to QueryCatalogTemplateList.

parentOID – The OID of the parent group.

title – the title of the group. This is typically the use one or more words like “Products” or “Off-road Cars”. The code for the group is typically the title with punctuation removed and spaces replaced with underscores.

Result Data Format

The ResultData will contain the OID of the new catalog group that has been created if the function is successful.

AssignItemsToCatalogGroup

Assigns item(s) to a specified catalog group.

Additional Required Parameters

- assign

assign – an XML document that contains all the assignment information. There are two methods of identifying the items to assign (OID or item Id). An OID is just an internal number that identifies the item within UltraCart. It is very common to see OIDs in the result sets from API calls. The format of the XML document is

```
<assign>
        <catalogGroup>catalog_group_oid_here</catalogGroup>
        <items>
                <item>item_oid_here</item>
                <item>item_oid_here</item>
                <itemId>item_id_here</itemId>
                <itemId>item_id_here</itemId>
        </items>
</assign>
```

You can utilize either item oids or item ids to identify the items to assign. You should not use both at the same time.

Result Data Format

No result data is returned. A successful call will have a resultCode of zero.

CreateCatalogGroupAttributes

Updates all the attributes associated

Additional Required Parameters

- CatalogGroup

CatalogGroup – XML document with the OID of the catalog group and all the attribute pairs. The attributes in the XML document will replace all existing attributes on the group. Below is an example of the XML document.

```
<catalog_group>
      <oid>catalog_group_oid_here</oid>
      <attributes>
            <attribute>
                  <name>author</name>
                  <value>HTML Coder</value>
            </attribute>
            <attribute>
                  <name>version</name>
                  <value>2.0</value>
            </attribute>
      </attributes>
</catalog_group>
```

Result Data Format

No result data is returned. A successful call will have a resultCode of zero.

DeleteCatalogGroup

This function deletes a catalog group and any descendants of the group.

Additional Required Parameters

- oid

oid - The identifier of the catalog group to delete. This should be obtained from a call to QueryCatalogGroups.

Result Data Format

No result data is returned. A successful call will have a resultCode of zero.

EditCatalogGroup

Edits the configuration associated with an individual catalog group.

Additional Required Parameters

- description
- groupTemplateOID
- hostOID
- imageURL
- itemTemplateOID
- oid
- parentOID
- title

code – each catalog group has a code associated with it. The code is appended to the path of the parent to create the new path for this group. You should not use spaces or other characters that make difficult urls in your code values.

description – The description of the group. This is commonly displayed as a paragraph of text on the catalog.

groupTemplateOID – the OID of the template to use when a group page is rendered. This should be obtained from a call to QueryCatalogTemplateList.

imageURL – (deprecated) the URL to an image that should be associated with the catalog group. This should not be used. Instead merchants should use the integrated multimedia functionality available within the catalog group management.

itemTemplateOID – the OID of the template to use when an individual item page is rendered within this group. This should be obtained from a call to QueryCatalogTemplateList.

oid – the OID of the catalog group to obtain. This is typically obtained from a call to QueryCatalogGroups.

parentOID – The OID of the parent group.

title – the title of the group. This is typically the use one or more words like “Products” or “Off-road Cars”. The code for the group is typically the title with punctuation removed and spaces replaced with underscores.

Result Data Format

No result data is returned. A successful call will have a resultCode of zero.

QueryCatalogGroups

Queries the configuration and hierarchy of all the catalog groups associated with a catalog.

Additional Required Parameters

- hostOID

hostOID – The OID of the catalog to query. This should be obtained from a call to QueryCatalogHostList.

Result Data Format

The ResultData is a Base-64 encoded XML document. The format of the XML document is simple. An example is shown below:

```
<groups>
  <group>
    <oid></oid>
    <parentOid></parentOid>
    <title></title>
    <description></description>
    <root></root>
    <imageURL></imageURL>
    <itemTemplateOid></itemTemplateOid>
    <groupTemplateOid></groupTemplateOid>
    <code></code>
  </group>
</groups>
```

The group element will repeat for as many groups as there are in the catalog. The single root element of the catalog is identified with a root = Y in the XML document.

QueryCatalogHostList

Queries a list of all the catalog hosts configured on the account.

Additional Required Parameters

- <none>

Result Data Format

The ResultData is a base-64 encoded XML document. An example is shown below:

```
<hosts>
  <host>
    <oid></oid>
    <name></name>
  </host>
  <host>
    <oid></oid>
    <name></name>
  </host>
</hosts>
```

The oid element is a unique identifier for the catalog host. This will be used in calls to add/edit catalog groups, etc. There can be as many host records in the XML document as there are catalog hosts configured on an individual UltraCart account.

QueryCatalogTemplateList

Queries all the templates associated with a given catalog host.

Additional Required Parameters

- hostOID

hostOID – the OID of the catalog host to retrieve templates for.

Result Data Format

The ResultData is a base-64 encoded XML document. An example of the document is shown below:

```
<templates>
  <template>
    <oid></oid>
    <name></name>
  </template>
</templates>
```

The template elements will repeat for as many configured templates on the host. The oids will be used in calls to AddCatalogGroup and EditCatalogGroup.

QueryItemsInCatalogGroup

Queries all the items assigned to a single catalog group.

Additional Required Parameters

- parentCatalogGroupOid

parentCatalogGroupOid – the OID of the catalog group to query the items on. This should be retrieved from a call to QueryCatalogGroups.

Result Data Format

The result is a base-64 encoded XML document. An example is provided below:

```
<items>
  <item>
    <oid></oid>
    <itemId></itemId>
    <description></description>
    <manufacturerName></manufacturerName>
    <manufacturerSku></manufacturerSku>
  </item>
</items>
```

The item element will repeat as many times as there are items assigned to the group. The oid is used in calls to RemoveItemsFromCatalogGroup. The itemId, description, and manufacturer information are returned for convenience to make building user interfaces easier.

RemoveItemsFromCatalogGroup

This function removes one or more items from a catalog group.

Additional Required Parameters

- remove

An XML document that describes the catalog group to remove items from and which items to remove. All the data within the document references OIDs of catalog groups and items. Below is an example document for removes item Oids 101, 102, and 103 from catalog group 10. The actual oids to make a call to this function will be retrieved from calls to QueryCatalogGroups and QueryItemsInCatalogGroup.

```
<remove>
  <catalogGroup>10</catalogGroup>
  <items>
    <item>101</item>
    <item>102</item>
    <item>103</item>
  </items>
</remove>
```

Result Data Format

No result data is returned. A successful call will have a resultCode of zero.

Item Management Functions

CreateItem

This function creates a new item, or updates an existing item, within the item management section using the details provided in an XML document.

Additional Required Parameters

- Item

Item – XML document based upon the Schema file located at https://secure.ultracart.com/xml/apiCreateItem.xsd The easiest way to understand the XML is to query an item using the QueryItem function.

Result Data Format

No result data is returned. A successful call will have a resultCode of zero.

DeleteItem

This function deletes one item from the account.

Additional Required Parameters

- ItemId

ItemId – The item ID of the item to delete from the account. This function uses the item ID and not the OID.

Result Data Format

No result data is returned. A successful call will have a resultCode of zero.

ItemMultimediaUpload

This function allows the programmatic upload of files to an item's multimedia section. This function is not suitable for large files over 10MB. For files that large merchants should use the virtual FTP server.

Additional Required Parameters

- data
- filename
- ItemId

data – base 64 encoded file data.

filename – the name of the file that is being uploaded.

ItemId – the item to associate this multimedia file with.

Result Data Format

No result data is returned. A successful call will have a resultCode of zero.

QueryItem

This function queries a single item from the account.

Additional Required Parameters

- ItemId

Item Id – item id of the item to query.

Result Data Format

If the ResultCode is zero then the function call was successful and the ResultData will contain a base-64 encoded XML document of the item's details. The XML will conform to:

https://secure.ultracart.com/xml/apiCreateItem.xsd

The returned XML can be modified and then used in an additional call to CreateItem in order

QueryItemFolders

This function queries the item folder structure within the UltraCart account under Item Management -> Items. This folder structure can be used to place items in particular folders in calls to CreateItem or useful in displaying the hierarchy in a catalog item assignment tool.

Additional Required Parameters

- hostOid

hostOid – the hostOid is the oid of a catalog host. The folders will be returned with a count of the number of unassigned items there are in the folder.

Result Data Format

The ResultData of the function call is a base-64 encoded XML document of the folder structure. Below is an example of the XML document format.

```
<folders>
  <folder>
    <oid></oid>
    <parentOid></parentOid>
    <description></description>
    <unassignedItems></unassignedItems>
  </folder>
</folders>
```

The folder element will repeat within the document as many times as there are folders configured on the system. The parentOid and oid information can be used to display the hierarchy within as item assignment user interface. The unassignedItems element will contain the number of items in the folder that are not assigned to any category within the catalog host. Displaying only folders with unassigned items within an item assignment tool can be it easier on the user.

QueryItemInventory

This function queries the available inventory for a given item.

Additional Required Parameters

- ItemId

ItemId – the item id to query inventory on.

Result Data Format

If the function is successful then the ResultCode will be zero and the ResultData will contain a base-64 encoded XML document. An example of the XML document is displayed below:

```
<inventory>
  <merchant_item_id></merchant_item_id>
  <is_inventory_tracked></is_inventory_tracked>
  <quantity></quantity>
  <allocated_to_placed_orders></allocated_to_placed_orders>
  <allocated_to_shopping_carts></allocated_to_shopping_carts>
  <available_to_allocate></available_to_allocate>
</inventory>
```

QueryItemInventoryList

This function queries the inventory level associated with all items on an UltraCart account.

Additional Required Parameters

- <none>

No additional parameters are required on this call.

Result Data Format

If the function is successful then the ResultCode will be zero and the ResultData will contain a base-64 encoded XML document. Below is an example of the XML:

```
<inventories>
  <inventory>
    <merchant_item_id></merchant_item_id>
    <quantity></quantity>
    <allocated_to_placed_orders></allocated_to_placed_orders>
    <allocated_to_shopping_carts></allocated_to_shopping_carts>
    <available_to_allocate></available_to_allocate>
    <dc code="DFLT">
      <quantity></quantity>
      <allocated_to_placed_orders></allocated_to_placed_orders>
      <allocated_to_shopping_carts></allocated_to_shopping_carts>
      <available_to_allocate>0</available_to_allocate>
    </dc>
  </inventory>
  <inventory>
    <merchant_item_id></merchant_item_id>
    <quantity></quantity>
    <allocated_to_placed_orders></allocated_to_placed_orders>
    <allocated_to_shopping_carts></allocated_to_shopping_carts>
    <available_to_allocate></available_to_allocate>
    <dc code="DFLT">
      <quantity></quantity>
      <allocated_to_placed_orders></allocated_to_placed_orders>
      <allocated_to_shopping_carts></allocated_to_shopping_carts>
      <available_to_allocate>0</available_to_allocate>
    </dc>
  </inventory>
</inventories>
```

The inventory element will repeat for every item configured on the UltraCart account. There will also be a “dc” element within each “inventory” element for each distribution center configured on the UltraCart account. This allows your program to understand the inventory status at each distribution center as well as the overall inventory.

QueryItemList

This function queries a list of all the item ids configured on an UltraCart account.

Additional Required Parameters

- <none>

No additional parameters are required on this function.

Result Data Format

If the function is successful then the ResultCode will contain a zero and the ResultData will contain a base-64 encoded XML document. Below is an example of the XML document:

```
<merchant_item_ids>
  <merchant_item_id></merchant_item_id>
  <merchant_item_id></merchant_item_id>
  <merchant_item_id></merchant_item_id>
  <merchant_item_id></merchant_item_id>
</merchant_item_ids>
```

The merchant_item_id element will be repeated for every single item configured on an account.

QueryItemsInFolder

This function queries all the items that are configured within a particular folder in the item management hierarchy.

Additional Required Parameters

- parentFolderOid

parentFolderOid – the Oid of the folder to query. This can be obtained with a call to QueryItemFolders.

Result Data Format

If the function is successful then the ResultCode will contain a zero and the ResultData will contain a base-64 encoded XML document. Below is an example of the XML document:

```
<items>
  <item>
    <oid></oid>
    <itemId></itemId>
    <description></description>
    <manufacturerName></manufacturerName>
    <manufacturerSku></manufacturerSku>
    <unassignedItem></unassignedItem>
    <inactive></inactive>
  </item>
</items>
```

The item element will be repeated for each item within the folder.

UpdateInventory

This function updates the inventory associated with multiple items. The inventory is specified at the individual distribution center level.

Additional Required Parameters

- Inventory

Inventory – An XML document that contains the item ids, distribution center identifier, and inventory levels to update. An example of the XML document is below.

```
<update_inventory>
  <distribution_center_code></distribution_center_code>
  <items>
    <item>
      <item_id></item_id>
      <quantity></quantity>
    </item>
    <item>
      <item_id></item_id>
      <quantity></quantity>
    </item>
    <item>
      <item_id></item_id>
      <quantity></quantity>
    </item>
  </items>
</update_inventory>
```

The XML document must conform to the Schema file located at:
https://secure.ultracart.com/xml/apiUpdateInventory.xsd

One API call per distribution center should be used. The quantity should reflect the inventory level before allocations to carts or placed orders.

Result Data Format

No result data is returned. A successful call will have a resultCode of zero.

Order Management Functions

DoesOrderExist

This function will check whether a given set of order Ids exists on an account.

Additional Required Parameters

- orders

orders – XML document containing all the order Ids to check. Below is an example:

```
<orders>
  <order>
    <order_id></order_id>
  </order>
  <order>
    <order_id></order_id>
  </order>
  <order>
    <order_id></order_id>
  </order>
</orders>
```

The order element can be repeated for as many order Ids as you want to check.

Result Data Format

If the function call is successful then the resultCode will be zero and the resultData will contain a base-64 encoded XML document. An example of the XML document is provided below.

```
<orders>
  <order>
    <order_id></order_id>
    <exists></exists>
  </order>
  <order>
    <order_id></order_id>
    <exists></exists>
  </order>
  <order>
    <order_id></order_id>
    <exists></exists>
  </order>
</orders>
```

The additional exists element will contain the value of “true” or “false” depending upon whether or not the order Id exists on the account. The function will also return the order Ids listed in the exact same order that they were provided in the request.

ExportOrders

This method allows programs to retrieve information associated with orders in XML format. This includes all the address, item, shipping, etc. information. The information is useful for integration with marketing, accounting, and fulfillment systems. Merchants that are using QuickBooks and want to do integration should simply use the UltraBooks software instead of attempting to write their own.

Additional Required Parameters

- StartDTS

- StopDTS
- DateType
- CurrentStage

StartDTS and StopDTS specify the time range to look for orders between. The format for each parameter is YYYY/MM/DD HH:MM:SS. All times should be specified in EST.

The DateType parameter specifies which date to consider in the search. The three options for this parameter are: Creation, Payment, and Shipment.

The CurrentStage specifies which stage the order must be in to match the search. The valid options are: AR, SD, CO, and REJ. These abbreviations stand for:

- PO – Pre-Orders
- PC – Pending Clearance
- AR – Accounts Receivable
- SD – Shipping Department
- CO – Completed Orders
- REJ – Rejected Orders

Result Data Format

The result data is an XML document similar to the way order information can be exported from the web interface. The schema definition for the XML document is located at:

http://secure.ultracart.com/xml/ultracart.xsd

Example Scenario

The following is an example scenario. The goal is to export all the orders that are completed yesterday. For this example we're going to use the following parameter values:

- merchantId = TEST
- login = mylogin
- password = mypassword
- function = ExportOrders
- StartDTS = 2004/01/01 00:00:00
- StopDTS = 2004/01/01 23:59:59
- CurrentStage = CO
- DateType = Shipment

https://secure.ultracart.com/cgi-bin/UCApi?merchantId=TEST&login=mylogin&password=mypassword&function=ExportOrders

&StartDTS=2004/01/01+00:00:00&StopDTS=2004/01/01+23:59:59&CurrentStage=CO&DateTyp
e=Shipment

ExportToAccounting

This function exports up to 100 pending orders in the specified format. The orders included in this result will be the first 100 orders that have the exported to QuickBooks flag set to false on them. This function is useful for building custom exports to an accounting package.

Additional Required Parameters

- format

format – valid values are "XML", "CSV", or "BizTracker". Typically the XML format is utilizes as it is the easiest format to represent the entire order data.

Result Data Format

If the function is successful the resultCode will be zero and the resultData will contain a base 64 encoded XML document. An example document is shown below:

```
<export>
  <orderIds>
    <orderId></orderId>
    <orderId></orderId>
    <orderId></orderId>
    <orderId></orderId>
  </orderIds>
  <document>base 64 encoded document</document>
</export>
```

The orderId element will repeat once for each order Id contained in the export up to 100. The contents of the document element are a base 64 encoded version of the export file. Are you process an order contained in this export you should call the MarkExportedToAccounting method for each order Id. This will set the flag on the order so that subsequent calls to this function will return the next batch of orders.

MarkExportedToAccounting

This function marks one or more orders as exported to accounting.

Additional Required Parameters

- orderIds

orderIds – XML document of all the orderIds to mark as exported to accounting. An example is shown below:

```
<orderIds>
  <orderId></orderId>
  <orderId></orderId>
  <orderId></orderId>
  <orderId></orderId>
  <orderId></orderId>
</orderIds>
```

Result Data Format

If the function is successful the resultCode is zero. There is no resultData associated with this function.

MarkOrderAsShipped

This function marks an order as shipped for the default distribution center.

Additional Required Parameters

- orderId
- skipCustomerNotification
- trackingNumber

orderId – the order id to mark as shipped.

skipCustomerNotification – prevents a shipment email from being sent to the customer

trackingNumber – the package tracking number. This parameter must be specified on the function call even if no tracking number is provided. Just provide an empty string if that is the case.

Result Data Format

If the function is successful the resultCode is zero. There is no resultData associated with this function.

ProcessedPayment

This function marks an order is having a successful payment processed on it. This function does not cause a card to be charged. It is only to facility offline payment processing and reporting back to UltraCart. The order will move from the Accounts Receivable to the Shipping Department or Completed Orders depending upon whether shipping is required.

Additional Required Parameters

- orderId

orderId – the order id to mark the payment as processed on.

Result Data Format

If the function is successful the resultCode is zero. There is no resultData associated with this function.

PurgeOrderPaymentInformation

This function will purge credit card information from an order.   This will clear out the card number and set the expiration to January, 1990.

Additional Required Parameters

- orderId

orderId – the order id to purge credit card information on.

Result Data Format

If the function is successful the resultCode is zero. There is no resultData associated with this function.

QueryOrder

This function will export a single order in XML Schema format for the given order Id.

Additional Required Parameters

- OrderId

OrderId – the order Id to export.

Result Data Format
If the function is successful the resultCode will be zero and the resultData will contain a base 64 encoded XML document for the order. The XML document will conform to the following Schema definition:

https://secure.ultracart.com/xml/ultracart.xsd

SearchOrders
This function searches the order database and returns the matching orders in an XML Schema export format. The result can contain up to 100 matching order records.

Additional Required Parameters
- firstName
- lastName
- email

firstName – first name to search on

lastName – last name to search on

email – email to search on

At least one of the three parameters is required. A % can be used as a wildcard in the search fields. All the parameters must be specified, but two of the three can be left blank.

Result Data Format
If the function is successful then the resultCode will be zero and the resultData will contain a base 64 encoded XML document. The XML document will conform to the following Schema definition:

https://secure.ultracart.com/xml/ultracart.xsd