CIQ Platform Specification and API

Last Revised 12/22/2011

Commerce IQ Platform Specification & API

In the advent of e-commerce technologies and smartphones equipped with cameras, GPS and Internet connectivity, the act of "shopping" as we once knew it has changed dramatically.

> Now "shopping commandos" repel in to their favorite retailer. Using their smartphone, they scan the manufacturer code of a TV or home appliance, compare prices and specs online, then purchase at the store, or if the discounts are worthwhile, head back to home "base" to purchase online.
>
> Rather than browsing, perhaps to pick up a few items that catch their eye, these shopping commandos are making surgical strikes; visiting a retailer to buy only that one targeted item, and then being evac'd from behind enemy lines as fast as they came in.

To many, this sounds like something that may occur in the distant future. But it is happening right now. In fact, during the 2011 holiday season, 70% of those using smartphones for holiday shopping used them in-store. "eMarketer 2011" says that smartphones now reach 38% of mobile users and that number is expected to increase nearly 50% over the next year.

Consider for example the reasons that consumers shop online. They like the convenience of shopping at home and can easily compare product quality, specifications, support and prices on many popular websites. Online consumers also have the benefit of seeing what others' product experiences have been and can even record their own experiences online using the Community aspects of the Internet.

Because of this, the retail landscape has started to shift. Electronics retailers are selling less; losing market share to e-commerce "e-tailers" like Amazon; both to high-ticket items, and higher-margin accessory items. In fact, Amazon's market share for consumer electronics sales has been increasing by about 4% each year since 2008 and currently represents about 24% of all consumer electronics business in the United States.

If there are so many benefits to online shopping, why isn't everyone doing it, and doing it for every purchase? As it turns out, there are benefits to shopping at brick and mortar retailers too. Why, for instance do consumers still shop at Best Buy for a TV when they can buy online?

The answer is that consumers like to experience products before making a purchase decisions. Don't you like to look at the walls and walls of TVs at electronics retailers to see which actually gives YOU the best viewing experience? Or to touch the washer and dryer you're considering purchasing; turn the knobs, open and close the door, look inside to "feel" the dryer's capacity.

Another benefit to shopping at brick and mortar stores is that you can usually make a purchase and take your prize home to enjoy that same day. No matter how competitive online retailers have become with shipping costs and fast delivery times, consumers don't get that immediate gratification that brick and mortars have to offer.

The Commerce IQ platform works in conjunction with smartphone shopping applications to combine the benefits of both e-commerce and brick and mortar shopping. Now, using their favorite smartphone shopping application, a consumer can go into Best Buy, Walmart or other retailer, snap an HDTV tag to obtain information about the product, do product comparisons, get reviews, etc. and at the same time receive manufacturer discounts for that same or similar item. What's more, the discounts can be redeemed at the retailer using the same smartphone technology. The benefits to the consumer are staggering. They have access to up-to-the-minute discount offers direct from the manufacturers and retailers of big-ticket items, at the precise time that they are making a decision to purchase the item.

Consumers aren't the only ones that benefit from the Commerce IQ Platform. Consider that manufacturers and retailers spend tens of billions of dollars per year in promotions and advertising in order to generate demand and to obtain market intelligence for price sensitivity, competition and

demographics. The effectiveness of their marketing spend is almost always difficult or impossible to measure. In addition, there will always be latency between the occurrence of an advertising impression, and the impact of that impression on a purchase.

Conversely, the Commerce IQ platform enables manufacturers and retailers to reach consumers at exactly the point in time that they are ready to purchase a product. Consider the example of a consumer at Best Buy using their smartphone app to snap an HDTV tag in order to do some online comparison shopping. The Consumer IQ Platform receives this information from the app and through a proprietary matching process, pushes pertinent discounts back to the consumer's smartphone.

These advertising discounts are very targeted in that they apply to the exact product that was scanned, or to a competing "like" product. They are also based on the day, time and location at which the tag was snapped. This relevancy of focus and context combined with the fact that the consumer is precisely at the point of making a purchase decision makes Commerce IQ an extremely powerful advertising medium for manufacturers and retailers.

Equally compelling is that the Commerce IQ Platform provides accurate, up-to-the-second analytics that until now could never be achieved through an online or print advertising, rebate, or promotional campaign. Every time a product is scanned, or a discount is sent to a smartphone, Commerce IQ tracks the information which includes the product code, geo-location, date and time, retailer, and other contextual information. In doing so a wealth of analytical information is available to Commerce IQ clients. Consider a few of the possibilities:

1) A geographic map (national, state, local, regional) overlaid with markers that represent the numbers and types of products scanned in various locations. The user can filter by retailer, product category, manufacturer and other interesting characteristics.
2) As a manufacturer I want to track how many scans were made of my product, but where I lost the purchase to a competitor. Or the inverse; I want to track how many scans were made of a competing product, but where I won the purchase. In either case, I want to know the discount amount that won the deal.
3) As a manufacturer, I want to know which size of our HDTVs gets the most, or least scans and consider a marketing campaign to drive more interest to a particular product model. I can use Commerce IQ's data analytics to monitor and verify the success of the campaign.
4) As a retailer, I want to know the number of scans that occur for my kitchen appliance products as compared to those of my competitors.

The value proposition and business model of Commerce-IQ parallels that of Google Ad Words. Consider that with Google Ad Words, marketers bid for "advertising space" based on search keywords that will trigger impressions of their ads. The more "common" the keyword, the more expensive the cost-per-click charged to the advertiser. Conversely, the more "specialized" the keyword, the less expensive the cost-per-click.

When a Google user performs a search using a word or phrase, Google begins to work its magic to find and display the most-relevant advertisements based on the search terms that were entered. This "magic" is based on the bids that were placed by marketers that relate to the search words entered by the user.

The Commerce IQ Platform works in a similar way to Google Ad Words but gives marketers a much richer set of characteristics on which to target their customers. Consider that every type of product, TV's, cameras, appliances; have their own sets of descriptive properties. For a TV, there's the screen size

and resolution whereas a camera has megapixels, and Capacity is a property of a refrigerator. Marketers can bid for "advertising space" based on specific products, or the properties of products, that are snapped by the consumer at a retail store. In addition to bidding on product properties, the Commerce IQ Platform also let's marketers target consumers based on the tag geographic location, retailer, day of the week, time of the day, etc.

This level of targeted advertising is the perfect fit for the "new world order" of shopping experience; taking advantage of existing technologies that are rapidly becoming commonplace and incorporating ALL of the resources that make it possible; the Internet, e-commerce, smartphones and our favorite retail stores.

Affiliate Services

Affiliate Services Data

Scan Activity

When a smartphone user scans a product code and sends it to CIQ, the information is captured in the ScanActivity table. The table has the following format:

Field Name	Type	Description
DateTime	datetime	The time at which a product scan was captured.
ScanData	String	The actual code or data that was scanned.
Longitude	Double	The GPS longitude (location) of the scan occurrence.
Latitude	Double	The GPS latitude (location) of the scan occurrence.

An example of the data follows:

DateTime	ScanData	Longitude	Latitude
2011-12-10 20:22:47.163	HTTP://1A.TT/1HBLKEZPJ	-79.7689599990845	40.4383935928345
2011-12-10 20:40:32.820	123456789	-79.7683353424072	40.4378995895386
2011-12-10 21:15:42.370	http://bit.ly/epa5ZX?r=qr	-79.768495615646	40.437850845084
2011-12-10 21:16:12.650	http://bit.ly/epa5ZX?r=qr	-79.7688238509742	40.4380306788165

Affiliate Services API

CProductAds **GetAdList** (Guid affiliateID, string scannedCode, double lon, double lat,
int adsToReturn);

Description:

This function accepts a barcode and GPS location then using the CIQ proprietary search engine, returns a list of manufacturer and/or retail store discount coupons that are most pertinent to the scanned product.

Arguments

Guid affiliateID	This is an ID specifically assigned to each Affiliate.
string scannedCode	This is a barcode that has been scanned from a smartphone. I'm not sure if this is the right type.
double lon, double lat	Represents the GPS location from a smartphone. Again, I'm not sure if this is the right type.
int adsToReturn	The maximum number of ads that should be returned in the list. There could possibly be tens or hundreds of ads. This lets the phone app specify the max number it will use.

Return:

CProductAds	Returned is a List object that contains zero or more CProductAd objects. Each CProductAd object is made up of the following elements (or properties):

string AdTitle	Typically would be displayed by the smartphone app in a list of Ads.
double DiscountAmount	The dollar amount of the discount being offered.
Guid adGuid	This Guid is an identifier for the actual Ad. In order to obtain the actual Ad, the caller would use this adGuid and call GetAdDetails (the service call which is also documented herein).

Current Disposition:

The input arguments afilliateID, scannedCode, lon, lat, adsToReturn are ignored for now. They are acting as placeholders. I'm not sure that the argument types are correct, so they may be changed in the future to accommodate smartphone scans and GPS.

The CIQ proprietary search engine is not yet implemented. I'm hardcoding results for now.

CProductAds **GetAdDetails** (Guid affiliateID, string scannedCode, double lon, double lat, Guid adGuid);

Description:

This function accepts a barcode, GPS location and adGuid and returns the Ad's associated CProductAd object.

Arguments

Guid affiliateID	This is an ID specifically assigned to each Affiliate.
string scannedCode	This is a barcode that has been scanned from a smartphone. I'm not sure

	if this is the right type.
double lon, double lat	Represents the GPS location from a smartphone. Again, I'm not sure if this is the right type.
Guid adGuid	A Guid which identifies the specific Ad whose information will be returned. The adGuid parameter would typically be obtained from a call to GetAdList where an adGuid property exists for each CProductAd list item.

Return:

CProductAd — Returned is a CProductAd object which has the following elements (or properties):

string AdTitle	Typically would be displayed by the smartphone app in a list of Ads.
double DiscountAmount	The dollar amount of the discount being offered.
string ProductName	The product that the ad applies to. This may be very specific, or broad, depending on the purpose of the coupon.
string Manufacturer	The manufacturer or retailer offering the coupon. Eg. Sony, Best Buy, etc.
string imageUrl	A url to a representative image. Check for empty string.

Current Disposition:

The input arguments afilliateID, scannedCode, lon, lat are ignored for now. They are acting as placeholders. I'm not sure that the argument types are correct, so they may be changed in the future to accommodate smartphone scans and GPS.

CProductAds **GetAdCoupon** (Guid affiliateID, string scannedCode, double lon, double lat, Guid redeemGuid);

Description:

This function accepts a barcode, GPS location and redeemGuid and returns the associated CProductAd object that represents an actual, redeemable coupon.

Arguments

Guid affiliateID	This is an ID specifically assigned to each Affiliate.
string scannedCode	This is a barcode that has been scanned from a smartphone. I'm not sure if this is the right type.
double lon, double lat	Represents the GPS location from a smartphone. Again, I'm not sure if this is the right type.
Guid redeemGuid	A Guid which identifies the specific Coupon whose information will be returned. The redeemGuid parameter would typically be obtained from a call to GetAdDetails using the adGuid property of its returned CProductAd object.

Return:

CProductAd	Returned is a CProductAd object which has the following elements (or properties):	
	string AdTitle	Typically would be displayed by the smartphone app in a list of Ads.
	double DiscountAmount	The dollar amount of the discount being offered.
	string ProductName	The product that the ad applies to. This may be very specific, or broad, depending on the purpose of the coupon.
	string Manufacturer	The manufacturer or retailer offering the coupon. Eg. Sony, Best Buy, etc.
	string imageUrl	A url to a representative image. Check for empty string.
	string manufacturerCode	This is a unique identifier of the coupon for the purposes of redemption.

Current Disposition:

The input arguments afilliateID, scannedCode, lon, lat are ignored for now. They are acting as placeholders.

CProductProperties **GetProductInformation** (string scannedCode, double lon, double lat);

Description:

This is probably more of a debugging function than anything. It accepts a scan of a product bar/QR code and GPS location and will return a Dictionary of product properties. For example, for an HDTV, that may include size, resolution, HZ, etc.

Arguments

string scannedCode	This is a barcode that has been scanned from a smartphone. I'm not sure if this is the right type.
double lon, double lat	Represents the GPS location from a smartphone. Again, I'm not sure if this is the right type.

Return:

CProductProperties	Returned is a Dictionary object that contains zero or more name/value pairs which are of type string. Each of these pairs represents a property of a product. Note that different types of products will have different sets of properties. As an example, an HDTV may have name/value pairs such as {Product/HDTV, Class/LED, Size/32, Manufacturer/Sony} while a camera may have name/value pairs like {Product/Camera, Megapixels/10, Manufacturer/Nikon}

Current Disposition:

The input arguments scannedCode, lon, lat are ignored for now. They are acting as placeholders. I'm not sure that the argument types are correct, so they may be changed in the future to accommodate smartphone scans and GPS.

The return dictionary of properties, for now, contains the list {Retailer/Best Buy, Product/HDTV, Class/LED, Size/32}. This is to test the plumbing of the function.

Product Services

The Product Services handle all product modeling requirements necessary for the creation and management of bids, mapping of product specifications to model numbers, and in the process of determining the best ads to present to the user when a code is scanned from an affiliate smartphone app.

Data

Product Taxonomy table

This table defines the attributes and valid values for each Product Classification that is supported by the CIQ platform. Since this system defines a Category/Subcategory/Product hierarchy for products, attributes and valid values may also be specified for the super classes. A special "ROOT CLASS" is the penultimate class, the virtual parent class for all Categories.

Here's how the attribute system works. Given the product hierarchy below, attributes can be defined at each level, such that when a product is "instantiated", it contains all of the properties of itself and its ancestors:

ROOTCLASS (Attributes: Brand)
Electronics (Attributes: Type)
Televisions (Attributes: Size)
HDTV (Attributes: Resolution, Refresh Rate)

When a new HDTV product is instantiated by the system, it's formation is:

Product Class: HDTV (Attributes: Brand, Type, Size, Resolution, Refresh Rate)

The CIQ **ProductTaxonomy** table has the following format.

Field Name	Type	Description
ProductClass	String	The Product Classification (or Category, Subcategory, or "ROOT CLASS").
Property	String	The name of a Property associated with the class.
ValidValues	String	When a class is created (as opposed to saved/loaded), valid values for the attribute are also provided. This will enable applications to provide valid attribute values when a new bid is created, or when a model number is cataloged in the database. One or more valid values are represented by a semicolon—delimited string.

An example of the database follows:

ProductClass	Property	ValidValues
HDTV	Type	Plasma;LCD;LED;Projection
HDTV	Size	1 - 20;21 - 29;30 - 39;40 - 49;50 - 59;60 and Up
HDTV	Resolution	1080p;1080i;720p
HDTV	Refresh Rate	60Hz;120Hz;240Hz;600Hz
Digital Camera	Megapixels	9 or Less;10 - 11;12 - 13;14 - 15;16 - 18;19+
Refrigerator	Capacity	Under 6cf;6 - 15 cf;16 - 19 cf;20 - 22 cf;23 - 25cf;26 cf and up
Refrigerator	Width	20in and under;21 - 26in;27 - 28in;29 - 30in;31 - 34in;35;36 - 39;40in and up
Refrigerator	Height	30in and under;31 - 50in;51 - 62in;63 - 66in;67 - 68in
ROOTCLASS	Brand	NULL

Product Manager API

CProduct **CreateProductInstance (**string productClassification, bool allowNulls **)**

Description:

Given the name of a product class, this function returns a new object instance which includes a list of attributes specific to the class, along with valid values for each attribute.

Arguments

String productClassification	The class name of the product that is to be instantiated.
Bool allowNulls	If true, indicates that this object will be used for UI, and that valid values will also be returned for each property.

Returns:

CProduct	Represents the instantiated object which supports the following properties.

String ProductClassification	The class name of the object.
List<CProperty> ProductProperties	A list of CProperty objects, each of which contains the following properties.

String PropertyName	The name of the property.
String PropertyValue	The value of the property. Null or empty represents no value.
List<String> ValidValues	A list of values which are deemed valid for the property. This is useful for user applications to set up Bids, etc.

Current Disposition:

I'm considering removing allowNulls argument in the next version. Not necessary.

bool **Serialize (**CProduct aProduct, ref string flattenedObj**)**

Description:

Given an instantiated CProduct object, this method "flattens" or serializes the object into the given string.

Arguments

CProduct aProduct	A product object that is to be serialized.
Ref string flattenedObj	A string that will contain the serialized product object.

Returns:

True if the given product object is valid and the serialization is successful.

bool **Deserialize (**string flattenedObj, ref CProduct aProduct**)**

Description:

Given a string that contains a serialized product object, this method de-serializes the object into the given CProduct.

Arguments

string flattenedObj	A string that contains a serialized product object.
Ref CProduct aProduct	A CProduct that will be the de-serialized object.

Returns:

True if the given arguments are valid and the de-serialization is successful.

Bid Services

Overview of Bid Services

Manufacturers and retailers can create discount advertisements using the CIQ platform, and then bid on the placement and frequency in which the ads are presented to smartphone users when they scan a code. The cost per click (CPC) of each ad is based on the details of the bid. The more general or frequently-scanned properties will have a higher CPC, while the more-specific, thus less-scanned properties will be lower.

Bid Services are used specifically to create and manage bids that manufacturers or retailers use to govern the placement and frequency of their discount advertising. Each Bid is characterized by a set of common properties and product properties. While product properties are specific to the class of product used as part of a bid, common bid properties include:

- Product Category: This is the highest, most-general level of product categorization. Examples include “Electronics”, “Cameras and Optics”, and “Kitchen-Dining”.

- Product Subcategory: This is the next level of detail in product categorization. Based on the product category, some examples of product subcategory are “Televisions”, “Cameras” and “Appliances”.

- Product Classification: This is the actual type of product that is used in a bid. Based on the Product Subcategory, some examples of a Product Classification are “HDTV”, “Digital Camera” and “Refrigerator”.

Depending on the Product Classification that is used in a bid, other properties that are specific to that classification are also available to bid on. These product-specific properties are managed by Product Services the details of which are provided in another section of this document.

Here is an example of how a bid is created. Both Bid Services and Product Services are required for this process. The user (Manufacturer or Retailer marketer) selects values from each of the below properties:

Product Category : “Any”, “Electronics”, “Cameras and Optics”, “Kitchen-Dining”, etc.

- If “Any” is selected, that means the placement of an advertisement will apply to ANY product that is scanned. The CPC will potentially be very high since it is based on the total number of product scans that have occurred in the past month. There will be no further selections for the below properties.
- If a product category, such as Electronics, is selected, then product subcategories (below) related to Electronics are presented to the user.

Product Subcategory : “Any”, “Audio”, “Communications”, “Computers”, “Televisions”, etc.

- If “Any” is selected, that means the placement of an advertisement will apply to ANY product within the selected Category, such as Electronics. Again, the CPC will potentially be high, since it is based on the total number of product scans that have occurred in the past month, for products in the selected Category. There will be no further selections available for the below properties.
- If a product subcategory, such as “Televisions” is selected, then Product Classifications related to Televisions are next presented to the user.

Product Classification : “Any”, “CRT”, “HDTV”, “Portable”, “Projection”, etc.

- If “Any” is selected, that means the placement of an advertisement will apply to ANY product of the selected subcategory, such as “Televisions”. The CPC will be potentially high since it’s based on the total number of product scans of products that fall within the selected subcategory. There will be no further selections available for the below properties.
- If a Product Classification such as “HDTV” is selected, properties specific to that classification are then presented, along with a selection of valid values for each. This is where Product Services are used to manage product-specific properties for a bid.

The following properties are managed by Product Services, but are included in this section because they are critical to the bid process.

Region: “Any”, “Northeast”, “Southeast”, “Midwest”, “Northwest”, “Southwest”, etc.

- In this case, “Any” will most-likely be the common choice, although bidders could place ads regionally by setting this property.
- This is a Product property whose values will be based on the “Product Category”. Consider CIQ may not be supporting certain product categories in all regions of the US (not sure about this yet).

Retailer: “Any”, “Best Buy”, “HH Gregg”, “Sears”, “Walmart”, etc.

- In this case, “Any” will most-likely be the common choice, although bidders could place ads for specific retailers.
- This is a Product property whose values are based on the Product Category, since certain retailers may not carry all categories of product.

Product Classification-specific properties: Based on the selected Product Classification (such as "HDTV"), other properties (and their valid value selections) may also be presented to the user. For example, for an HDTV these properties may include:

Type : Any, Plasma, LDC, LED
Size: Any, 1-20, 21 – 29, 30 – 39, etc.
Resolution: Any, 1080p, 1080i, 720p, etc.
Refresh Rate: Any, 60Hz, 120Hz, 240Hz, 600Hz, etc.

- Note that "Any" is a valid option for each property.
- The bidder can specialize any or none of the product properties.

Calculating CPC

At any time the bidder can request the software to calculate the CPC based on the current property selections. In general, the algorithm will evaluate each property selection against the number of product scans, from the most recent 30 day period, that match the property selection. If a property has a value of "Any", then more of the scans will apply, thus the CPC will be higher.

As an example, assume that a bid is configured for Product Classification of "HDTV", where the Size property is "30 – 39", and all other product properties are "Any". To calculate the CPC, the total number of scans for "30 – 39 inch HDTVs", within the last 30 days, is counted.

As another example, assume that a bid is configured where Product Category = "Electronics" and Product Subcategory = "Any". To calculate the CPC, the total number of scans for ALL Electronics products, within the last 30 days, is counted.

Data

Bids Table

Bids are stored in the Bids Table which has the following format:

Field Name	Type	Description
ProdCategory	String	The top level product category value for a bid. Valid values include "Any".
ProdSubcategory	String	The subcategory value for a bid. Valid values include "Any" and empty string.
ProdClass	String	The product classification used for a bid. Valid values include "Any" and empty string.
ProdBlob	String	Contains a serialized representation of a product that includes all of the product-specific properties and values specified for a bid.

An example of the data table is:

ProdCategory	ProdSubcategory	ProdClass	ProdBlob

Electronics	Televisions	HDTV	1;HDTV;5;Type;LCD;Size;;Resolution;1080i;Refresh Rate;;Brand;;
Cameras & Optics	Cameras	Digital Camera	1;Digital Camera;2;Megapixels;12 - 13;Brand;;
Kitchen-Dining	Appliances	Refrigerator	1;Refrigerator;4;Capacity;;Width;29 - 30in;Height;51 - 62in;Brand;;

Product Classes table

Each product classification, such as an HDTV, Refrigerator and Camera will fall under certain categories and subcategories. This categorization can be thought of as a class hierarchy in an object-oriented programming language. It is useful in organizing the potentially vast number of product classifications in the system, and inheritance of properties may become a useful, if not necessary, as complex products are modeled.

An example of how this class hierarchy can be structured can be seen as follows, which is a take-off of the product taxonomy that is provided online by Google.

Category	**Subcategory**	**Product Classes**
Electronics	Televisions	HDTV, Projection TV, Portable TV
	Video Players	Blu-Ray, HD-DVD, DVD Recorders, VCR
Kitchen & Dining	Appliances	Refrigerators, Ranges, Steamers

The CIQ **ProductClasses** table has the following format.

Field Name	**Type**	**Description**
Category	String	The major category for a product classification. It may have one or more subcategories.
Subcategory	String	The subcategory for a product classification. It may have one or more product classifications.
ProductClass	String	The product classification.

An example of the data table is:

Category	**Subcategory**	**ProductClass**
Electronics	Televisions	HDTV
Cameras & Optics	Cameras	Digital Camera
Kitchen-Dining	Appliances	Refrigerator

Using this table the following is determined:

1) All of the available Categories
2) All of the Subcategories for a given Category
3) All of the given Product Classifications for a given Category & Subcategory.

4) Given a Product Classification, the parent Subcategory and Category.

Bid Services API

CBid **CreateBidInstance ( )**

Description:

This function returns a new Bid object which can be subsequently used to get and set property values and to save bids to the database.

Returns:

CBid	Represents the instantiated object which supports the following properties.	
	String Product Category	The product category for this bid which is used in the determination of the placement/frequency of an Ad. May also have the value “Any”.
	String Product Subcategory	The product subcategory for this bid which is used in the determination of the placement/frequency of an Ad. May also have the value “Any” or may be null (if Product Category is “Any”).
	String ProductClass	The product class for this bid which is used in the determination of the placement/frequency of an Ad. May also have the value “Any” or may be null (if Product Subcategory is “Any”). If this value is not “Any” and not null, there should be an associated Product (see Product Services) which serves to provide product-specific bid attributes.

bool **SaveBid(**CBid aBid, string flattenedProduct**)**

Description:

This function saves a bid to the CIQ platform.

Arguments

CBid aBid	The Bid object to save.
String flattenedProduct	This is the result of serializing a product object that has been instantiated and configured as part of the associated bid.

Returns:

Returned true if the arguments are valid and the "save" was successful.

List<String> **GetProductCategories( )**

Description:

This function returns a list of all available product categories of the CIQ platform.

Returns:

Returned is a list of string elements such that each element names a product category.

List<String> **GetProductSubcategories (**string sCategory**)**

Description:

Given a category, this function returns a list of all available product subcategories represented in the CIQ platform.

Arguments

String sCategory	The category name for which its subcategories will be retrieved.

Returns:

Returned is a list of string elements such that each element names a product subcategory.

List<String> **GetProductClassifications (**string sCategory, string sSubcategory**)**

Description:

Given a category and subcategory, this function returns a list of all available product classes represented in the CIQ platform.

Arguments

String sCategory	The category name for which product class names will be retrieved.
sSubcategory	The subcategory for which product class names will be retrieved.

Returns:

Returned is a list of string elements such that each element names a product classification.