Price Spectre

1.0

Generated by Doxygen 1.9.1

Chapter 1

Price Spectre Custom Algorithms

This document describes a javascript language interface that allows users to create their own custom algorithms.

1.1 Introduction

Price Spectre is a dynamic pricing agent for eBay listings. Every user has access to several pre-defined algorithms (pricing scripts) but some users may need more flexibility or control with their pricing.

Price Spectre provides a javascript interface which allows users to write their own scripts to control their pricing. Each user-defined script is run once per pricing cycle per managed listing.

It is assumed the reader already has a beginner level understanding of javascript and programming concepts. An introduction to the javascript language can be found at https://developer.mozilla.org/en/A↩_re-introduction_to_JavaScript . Since pricing scripts are run server side and not in a browser all browser related javascript functions are disabled.

Chapter 2

Algorithm Editor

The algorithm editor consists of three components.

2.1 Script

This is where the pricing script is input. The script must be valid javascript code.

2.2 Algorithm Options

2.2.1 Name

Allows the user to specify a name of this algorithm.

2.2.2 Require Search Parameters

Allows the user to indicate whether search parameters (keywords or product code) are required.

2.2.3 Perform Search Automatically

Allows the user to specify whether a search is performed automatically at the start of the script without requiring a call to PS::performSearch() or PS::performSafeSearch().

If this is set, it counts towards the total search count for premium billing purposes.

If this is set, Require Search Parameters Require Search Parameters must be also checked.

2.2.4 X

Allows the user to define whether the variable x is defined in the algorithm along with its datatype.

2.2.5 Y

Allows the user to define whether the variable y is defined in the algorithm along with its datatype.

2.2.6 Description

Allows the user to specify a description for the algorithm.

2.2.7 Number of Searches

The calculated number of searches performed by the algorithm. This is calculated every time the algorithm is saved. If the number of searches exceeds 1 then premium points may be charged for using the algorithm. See your "Premium Points" page for details on billing.

2.3 Algorithm Simulator

The embedded search console allows the user to test their new algorithm against a live listing. The simulator uses the script as it is currently defined in the script section. It is not necessary to save the script between changes for them to take effect in the simulator.

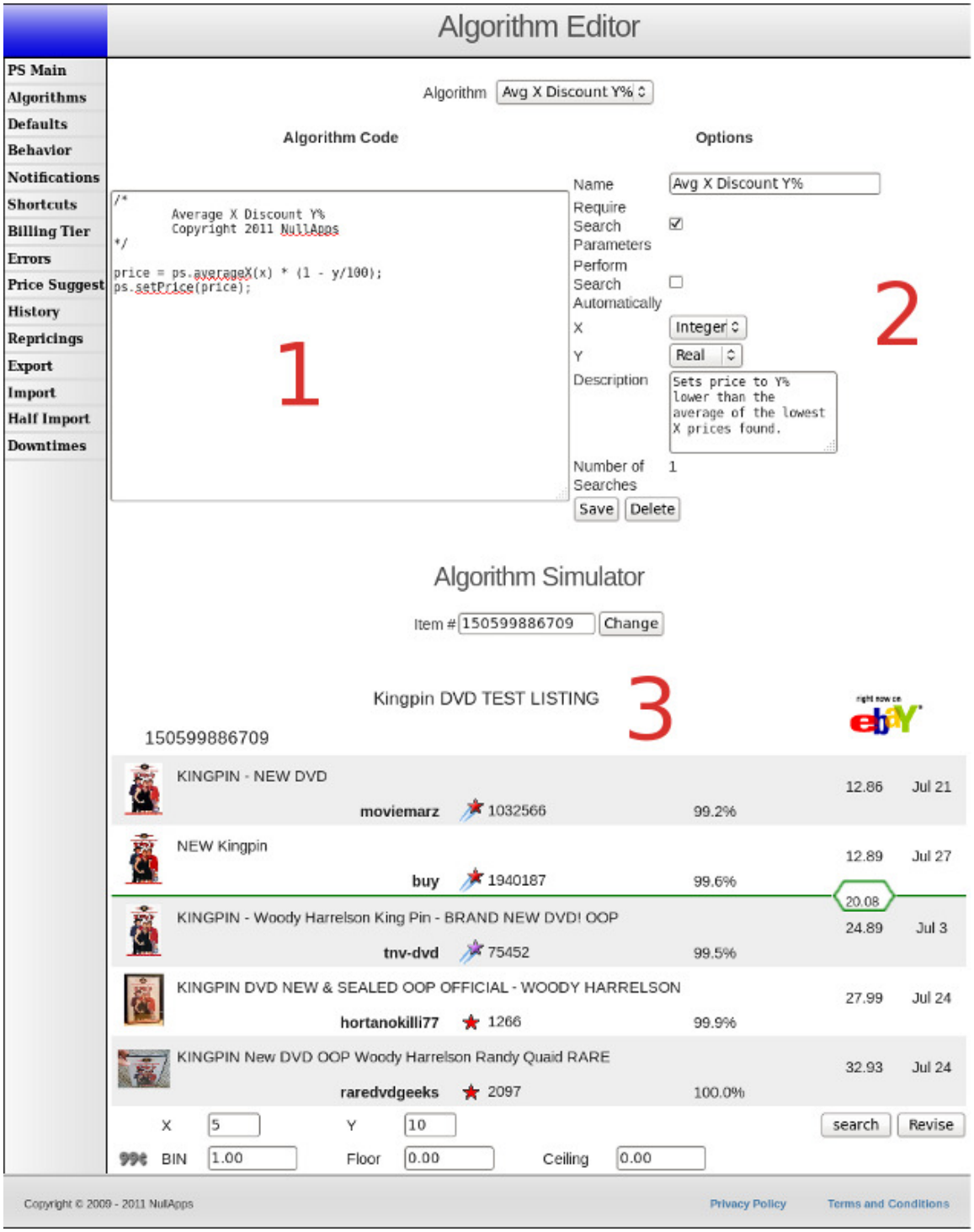

Figure 2.1 Algorithm Simulator

Chapter 3

Pricing Logic

The following steps occur once per pricing cycle per managed listing:

1) Initial search is performed if the option "Perform Search Automatically" was selected.

2) The PS object "ps" is created. This is the object the user-defined script interfaces with. Variables x and y are also set if specified.

3) User-defined script is executed. In order to set a new total price PS::setPrice() must be executed.

4) If an exception was thrown, then execution stops and an error message is sent to the user.

5) "No competition found" check is performed.

5a) If no competitors were found and PS::overrideNoCompetitionFound() was not called, then the new total price is changed according to the behavior "No competition found".

5b) Otherwise, the new total price is passed through the Bat'a pricing filter if Bat'a pricing is enabled.

6) The new total price is then passed through the floor and ceiling price filter to ensure it meets these constraints.

7) Finally the total price of the listing is set to the new total price.

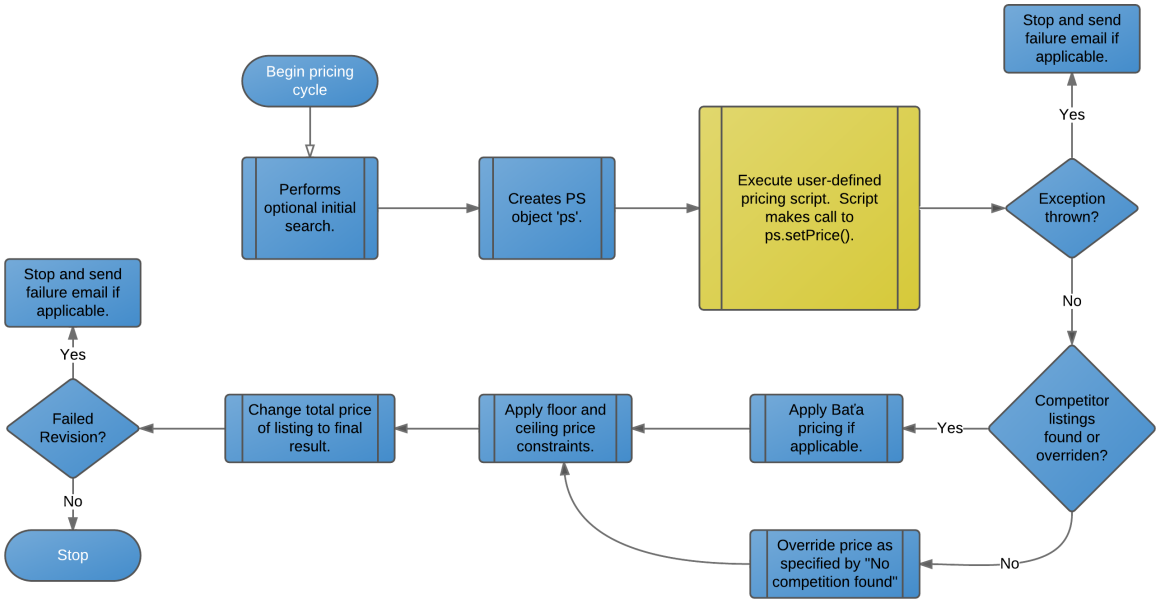

Figure 3.1 Pricing Logic

Chapter 4

Script Debugging

Price Spectre provides two ways of receiving output from custom algorithms.

4.1 throw

The javascript throw syntax is supported. This immediately terminates the script with the specified error message. Use this on conditions that are unrecoverable.

```
throw 'An error has occurred';
```

4.2 debugMessage

The PS class has a public function PS::debugMessage() that is similar to throw but does not immediately terminate the script. Each call to PS::debugMessage() appends the new message to the previous ones. This is useful in cases where control flow is being observed or where multiple different bugs or errors must be tracked.

Attention

Even though execution continues after a call to PS::debugMessage() the script will still terminate with an error once execution finishes.

```
if(x < 0)
   ps.debugMessage('x is less than 0 x = ' + x + '.');
if(y < 0)
   ps.debugMessage('y is less than 0 y = ' + y + '.');
```

Chapter 5

Sample Scripts

Price Spectre provides several sample scripts to get started. Several of these are also pre-defined algorithms that are available to all users.

All sample scripts are free to use in part or whole within Price Spectre. Users are welcome to create their own derivative works based on these samples for use within Price Spectre.

5.1 X Lowest By $Y

The X Lowest By $Y script sets a final price that is Y less than the Xth lowest price found.

```
// X Lowest By $Y
// Assumes "Perform Search Automatically" is selected.
// Copyright 2011 NullApps
if(x < 1)
   throw 'X must be positive.';
count = 0;
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
   count++;
   // if this is the listing to match against
   if(count >= x)
   {
      ps.setPrice(results[i].getPrice() - y);
      break;
   }
}
```

5.2 X Lowest By Y%

The X Lowest Y% script sets a final price that is Y less than the Xth lowest price found.

```
// X Lowest Y%
// Assumes "Perform Search Automatically" is selected.
// Copyright 2011 NullApps
if(x < 1)
   throw 'X must be positive.';
if(y > 50)
   throw 'Y must be 50% or less.';
count = 0;
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
   count++;
   // if this is the listing to match against
   if(count >= x)
   {
      ps.setPrice(results[i].getPrice() * (1 - y/100));
      break;
   }
}
```

5.3 Average X Y%

The Average X Y% script sets a final price that is Y% less than the average X lowest prices found.

```
// Average X Y%
// Assumes "Perform Search Automatically" is NOT selected.
// Copyright 2011 NullApps
if(x < 1)
   throw 'X must be positive.';
if(y > 50)
   throw 'Y must be 50% or less.';
// Grab the average of the X lowest priced listings.
averagePrice = ps.averageX(x);
// Sets price to y% lower than the average of the lowest X prices found.
ps.setPrice(averagePrice * (1 - y/100));
```

5.4 Discount X Days $Y

The Discount X Days $Y script sets a final price that is Y less than the current price if X days have passed since the last price change.

```
// Discount X Days $Y
// Copyright 2011 NullApps
if(x < 1)
   throw 'X must be positive.';
if(y == 0)
   throw 'Y cannot be 0.';
secondsPerDay = 86400;
secondsSincePriceChange = ps.getSecondsSinceLastPriceChange();
daysSinceChange = secondsSincePriceChange / secondsPerDay;
currentPrice = ps.getCurrentPrice();
if(daysSinceChange >= x)
   ps.setPrice(currentPrice - y);
```

5.5 Discount X Days Y%

The Discount X Days Y% script sets a final price that is Y% less than the current price if X days have passed since the last price change.

```
// Discount X Days Y%
// Copyright 2011 NullApps
if(x < 1)
   throw 'X must be positive.';
secondsPerDay = 86400;
secondsSincePriceChange = ps.getSecondsSinceLastPriceChange();
daysSinceChange = secondsSincePriceChange / secondsPerDay;
currentPrice = ps.getCurrentPrice();
if(daysSinceChange >= x)
   ps.setPrice(currentPrice * (1 - y/100));
```

5.6 Quota Elastic X Days Y%

The Quota Elastic script increases or decreases price Y% depending on whether the required quota of sales was achieved over the period of X days.

Attention

This is not the same as the pre-defined Elastic algorithm which uses the revenue generated in determining the final price.

```
// Quota Elastic X Days Y%
// Copyright 2011 NullApps
if(x < 1)
  throw 'X must be positive.';
if(y <= 0)
  throw 'Y must be positive.';
if(y > 50)
  throw 'Y must be 50% or less.';
quota = 3;
secondsPerDay = 86400;
secondsPerX = secondsPerDay * x;
secondsSincePriceChange = ps.getSecondsSinceLastPriceChange();
// Make sure enough time has elapsed.
if(secondsSincePriceChange >= secondsPerX)
{
   numSales = ps.getNumSalesSince(secondsPerX);
   currentPrice = ps.getCurrentPrice();
   // Change price by y% depending on if sales quota has been reached.
   if(numSales >= quota)
      ps.setPrice(currentPrice * (1 + y/100));
   else
      ps.setPrice(currentPrice * (1 - y/100));
}
```

5.7 Ebay Products Lowest

The Find Ebay Products Lowest script uses the WS class to call eBay's Finding API and set the price $0.01 lower than the lowest price found

Attention

This script is provided for informational purposes only and should not be used. Leave calling eBay's API to Price Spectre as we have years of experience doing this. However this script can be used as a starting point for calling similar external APIs.

```
// Find Ebay Products Lowest
// Copyright 2011 NullApps
// fetch UPC from keywords
var upc = ps.getSearchParameter('keywords');
// create request XML
var xmlStr = "<?xml version='1.0'?>"
   + '<findItemsByProductRequest xmlns="http://www.ebay.com/marketplace/search/v1/services">'
   + "<productId type=\"UPC\">" + upc + "</productId>"
   + "<sortOrder>PricePlusShippingLowest</sortOrder>"
   + "</findItemsByProductRequest>";
var request = new XMLDocument(xmlStr);
var headers = new Array(
   "X-EBAY-SOA-OPERATION-NAME: findItemsByProduct",
   "X-EBAY-SOA-SECURITY-APPNAME: <YOUR APP ID>",
   "X-EBAY-SOA-GLOBAL-ID: EBAY-US",
   "X-EBAY-SOA-SERVICE-VERSION: 1.11.0"
);
ws.setRequestBody(request);
ws.setHeaders(headers);
ws.setEndpoint("http://svcs.ebay.com/services/search/FindingService/v1");
var response = ws.post();
var responseXML = new XMLDocument(response);
var priceNode = responseXML.getElement('currentPrice');
var price = ps.getCurrentPrice();
if(priceNode)
{
   price = priceNode.getValue() - 0.01;
}
ps.setPrice(price);
```

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

Data Structure Documentation

7.1 Listing Class Reference

Public Member Functions

- getListingID ()
- getTitle ()
- getPrice ()
- getStartTime ()
- getStartDate ()
- getEndTime ()
- getEndDate ()
- getSeller ()
- getMerchantID ()
- getFeedbackScore ()
- getFeedbackPercent ()
- getFeedbackRating ()
- getFeedbackStar ()
- isTRS ()
- getReturnsAccepted ()
- getFreeShipping ()
- getOneDayShipping ()
- getConditionID ()
- getConditionName ()
- getSite ()
- getLocation ()
- getSecondsSinceStart ()
- getSecondsRemaining ()
- isDeleted ()

7.1.1 Detailed Description

A class that represents a competitor's listing.

7.1.2 Member Function Documentation

7.1.2.1 getConditionID()

```
getConditionID ( )
```

Retrieves the numerical representation of the condition of the item.

Returns

If a condition is specified returns an integer as specified at http://developer.ebay.↩com/devzone/finding/callref/Enums/conditionIdList.html . Returns 0, otherwise.

```
// Delete all non-new listings
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
  // Delete if not new
  if(results[i].getConditionID() != 1000)
  {
     ps.delete(i);
  }
}
```

7.1.2.2 getConditionName()

```
getConditionName ( )
```

Retrieves the string representation of the condition of a listing, if available.

Returns

A string such as 'Brand New', 'Like New', 'New in Box', etc. This value is dependent on the category and site the item is listed to. See http://pages.ebay.com/sellerinformation/sellingresources/itemconditior html and http://developer.ebay.com/devzone/finding/callref/Enums/condition↩IdList.html for example return values.

7.1.2.3 getEndDate()

```
getEndDate ( )
```

Retrieves the end date of a competitor's listing.

Returns

The end date of the listing in format 'MMM DD' if listed on eBay. Returns an empty string, otherwise.

7.1.2.4 getEndTime()

```
getEndTime ( )
```

Retrieves the number of seconds after 1970-01-01T00:00:00Z that the listing will end.

Returns

The epoch time representing when the listing will end if the listing was posted to eBay. Returns 2147483647, otherwise.

```
results = ps.getResults();
numResults = ps.getNumResults();
now = Math.round(Date.now() / 1000);
secondsPerDay = 86400;
// Delete all results that will end in less than one day.
for(i = 0; i < numResults; ++i)
{
   end = results[i].getEndTime();
   timeleft = end - now;
   if(timeleft < secondsPerDay)
   {
      ps.delete(i);
   }
}
```

7.1.2.5 getFeedbackPercent()

```
getFeedbackPercent ( )
```

Retrieves the feedback percentage of the listing's seller.

Returns

For eBay listings, a real number between 0 and 100 representing the feedback percentage of the listing's seller. Returns 0 for Amazon listings.

```
// Delete all listings that have an eBay feedback percent below 98% or Amazon rating below 4.8.
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
  site = results[i].getSite();
  percent = results[i].getFeedbackPercent();
  rating = results[i].getFeedbackRating();
  if(site == 'eBay' && percent < 98)
  {
     ps.delete(i);
  }
  else if(site == 'Amazon' && rating < 4.8)
  {
     ps.delete(i);
  }
}
```

7.1.2.6 getFeedbackRating()

```
getFeedbackRating ( )
```

Retrieves the feedback rating of the listing's seller.

Returns

For eBay listings, returns 0. For Amazon sellers returns a real number between 0 and 5 representing the seller's feedback rating.

```
// Delete all listings that have an eBay feedback percentage below 98% or Amazon rating below 4.8.
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
  site = results[i].getSite();
  percent = results[i].getFeedbackPercent();
  rating = results[i].getFeedbackRating();
  if(site == 'eBay' && percent < 98)
  {
     ps.delete(i);
  }
  else if(site == 'Amazon' && rating < 4.8)
  {
     ps.delete(i);
  }
}
```

7.1.2.7 getFeedbackScore()

```
getFeedbackScore ( )
```

Retrieves the feedback score of the listing's seller.

Returns

An integer representing the net feedback score of the listing's seller.

```
// Delete all listings that have an Amazon feedback score below minFeedback or rating less than 4.8.
// By default the minFeedback filter only works on eBay listings.
results = ps.getResults();
numResults = ps.getNumResults();
minFeedback = ps.getSearchParameter('minFeedback');
for(i = 0; i < numResults; ++i)
{
  site = results[i].getSite();
  rating = results[i].getFeedbackRating();
  score = results[i].getFeedbackScore();
  if(site == 'Amazon' && (rating < 4.8 || score < minFeedback))
  {
     ps.delete(i);
  }
}
```

7.1.2.8 getFeedbackStar()

```
getFeedbackStar ( )
```

Retrieves a string representation of the feedback star of the seller.

Returns

If the listing is on eBay returns a string representation such as 'none', 'yellow', or 'silverShooting'. Returns 'none' for all Amazon listings.

7.1.2.9 getFreeShipping()

```
getFreeShipping ( )
```

Retrieves whether the listing offers free shipping.

Returns

If free shipping is offered return TRUE. Returns FALSE, otherwise.

```
// Delete all listings that don't offer free shipping
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
  if(!results[i].getFreeShipping())
  {
     ps.delete(i);
  }
}
```

7.1.2.10 getListingID()

```
getListingID ( )
```

Retrieves the item number or ASIN of a competitor's listing.

Returns

If the listing is found on eBay, returns the 12 digit item number. If the listing was located on Amazon then return the ten character ASIN.

```
// Sets price first listing found that isn't specified by X.
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
   if(results[i].getListingID() != x)
   {
      ps.setPrice(results[i].getPrice());
      break;
   }
}
```

7.1.2.11 getLocation()

```
getLocation ( )
```

Retrieves the country the item is located.

Returns

Two-letter ISO 3166 country code to indicate the country where the item is located (e.g., "US" for the United States or "GB" for the United Kingdom).

```
results = ps.getResults();
numResults = ps.getNumResults();
// Delete all results that are not located in the US or Canada.
for(i = 0; i < numResults; ++i)
{
   country = results[i].getLocation();
   if(country != 'US' && country != 'CA')
   {
      ps.delete(i);
   }
}
```

7.1.2.12 getMerchantID()

```
getMerchantID ( )
```

Retrieves the merchant ID of the listing's seller.

Returns

For Amazon listings returns the Amazon merchant ID of the listing's seller. Otherwise returns an empty string.

7.1.2.13 getOneDayShipping()

```
getOneDayShipping ( )
```

Retrieves whether the listing offers one day shipping.

Returns

If one day shipping is available return TRUE. Returns FALSE, otherwise.

```
// Delete all listings that don't offer one day shipping
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
  if(!results[i].getOneDayShipping())
  {
     ps.delete(i);
  }
}
```

7.1.2.14 getPrice()

```
getPrice ( )
```

Retrieves the total price of a competitor's listing.

Returns

The total price of the competitor's listing.

```
// Matches the price of the lowest found listing.
results = ps.getResults();
numResults = ps.getNumResults();
if(numResults > 0)
   ps.setPrice(results[0].getPrice());
```

7.1.2.15 getReturnsAccepted()

```
getReturnsAccepted ( )
```

Retrieves whether the listing allows for returns.

Returns

If returns are accepted return TRUE. Returns FALSE, otherwise.

```
// Delete all listings that allow returns.
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
  if(results[i].getReturnsAccepted())
  {
     ps.delete(i);
  }
}
```

7.1.2.16 getSecondsRemaining()

```
getSecondsRemaining ( )
```

Retrieves the number of seconds before the listing will end.

Returns

The number of seconds that before the listing will end if it was listed on eBay. Returns 2147483647, otherwise.

```
results = ps.getResults();
numResults = ps.getNumResults();
secondsPerDay = 86400;
// Delete all results that will end in less than one day.
for(i = 0; i < numResults; ++i)
{
   timeleft = results[i].getSecondsRemaining();
   if(timeleft < secondsPerDay)
   {
      ps.delete(i);
   }
}
```

7.1.2.17 getSecondsSinceStart()

```
getSecondsSinceStart ( )
```

Retrieves the number of seconds since the listing has started.

Returns

The number of seconds that have elapsed since the listing started if it was listed on eBay. Otherwise, returns 0.

```
results = ps.getResults();
numResults = ps.getNumResults();
secondsPerDay = 86400;
// Delete all results that have been listed less than one day.
for(i = 0; i < numResults; ++i)
{
   age = results[i].getSecondsSinceStart();
   if(age < secondsPerDay)
   {
      ps.delete(i);
   }
}
```

7.1.2.18 getSeller()

```
getSeller ( )
```

Retrieves the seller ID of the listing's seller.

Returns

A string containing the userid of the listing's seller.

7.1.2.19 getSite()

```
getSite ( )
```

Retrieves the site the listing is from.

Returns

The string 'eBay' if the item is listed on eBay. The string 'Amazon' if the item is listed on Amazon.

7.1.2.20 getStartDate()

```
getStartDate ( )
```

Retrieves the start date of a competitor's listing.

Returns

The start date of the listing in format 'MMM DD' if listed on eBay. Returns an empty string, otherwise.

7.1.2.21 getStartTime()

```
getStartTime ( )
```

Retrieves the number of seconds after 1970-01-01T00:00:00Z that the listing was listed.

Returns

The epoch time representing when the listing was first listed if the listing was posted to eBay. Returns 0, otherwise.

```
results = ps.getResults();
numResults = ps.getNumResults();
now = Math.round(Date.now() / 1000);
secondsPerDay = 86400;
// Delete all results that have been listed less than one day.
for(i = 0; i < numResults; ++i)
{
   start = results[i].getStartTime();
   age = now - start;
   if(age < secondsPerDay)
   {
      ps.delete(i);
   }
}
```

7.1.2.22 getTitle()

```
getTitle ( )
```

Retrieves the title of a competitor's listing.

Returns

The title of listing.

```
results = ps.getResults();
numResults = ps.getNumResults();
// Delete all results containing the word 'parts'
// Note: this should be done using the keyword "-parts" instead.
for(i = 0; i < numResults; ++i)
{
   title = results[i].getTitle();
   if(title.indexOf('parts') != -1)
   {
      ps.delete(i);
   }
}
```

7.1.2.23 isDeleted()

```
isDeleted ( )
```

Retrieves whether the listing has been deleted or not.

Returns

TRUE if the listing has been deleted. Otherwise, returns FALSE.

```
// Delete all non-new listings
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
  // Delete if not new
  if(results[i].getConditionID() != 1000)
  {
     ps.delete(i);
  }
}
// Perform X lowest by $Y on remaining listings.
count = 0;
for(i = 0; i < numResults; ++i)
{
   // skip if this is a deleted listing.
   if(results[i].isDeleted())
      continue;
   // if this is the listing to match against
   count++;
   if(count >= x)
   {
      ps.setPrice(results[i].getPrice() - y);
      break;
   }
}
```

7.1.2.24 isTRS()

```
isTRS ( )
```

Retrieves whether or not the listing is from a Top Rated Seller.

Returns

If the listing is on eBay and the seller is a Top Rated Seller then return TRUE. Returns FALSE, otherwise.

7.2 PS Class Reference

Public Member Functions

- performSearch ()
- getErrorMessage ()
- performSafeSearch ()
- performSelfSearch ()
- performSelfSafeSearch ()
- xLowestY ($x, $y, $reuseSearchResults=false)
- xLowestYPercent ($x, $y, $reuseSearchResults=false)
- averageX ($x, $reuseSearchResults=false)
- elastic ($x)
- getMyAmazonPrice ()
- getCeilingPrice ()
- getFloorPrice ()
- getShippingHandling ()
- getSearchParameter ($key)
- setSearchParameter ($key, $value)
- getListingID ()
- getSiteCountry ()
- getResults ()
- getRealResults ()
- getNumResults ()
- getRealNumResults ()
- getRank ($price)
- getRealRank ($price)
- getNoCompetitionFound ()
- overrideNoCompetitionFound ()
- delete ($index)
- setPrice ($price)
- getCurrentPrice ()
- getLastPriceChangePrice ()
- getLastPriceChangeTime ()
- getSecondsSinceLastPriceChange ()
- getLastSaleTime ()
- getNumSales ($start=0, $end=2147483647)
- getNumSalesSince ($seconds)
- getData ()
- writeData ($str)
- debugMessage ($message)
- getSecondsSinceStart ()
- getSecondsRemaining ()
- getQuantity ()
- getGoldBid ($currency, $unit)
- getGoldAsk ($currency, $unit)
- getSilverBid ($currency, $unit)
- getSilverAsk ($currency, $unit)
- getPlatinumBid ($currency, $unit)
- getPlatinumAsk ($currency, $unit)
- getPalladiumBid ($currency, $unit)
- getPalladiumAsk ($currency, $unit)

7.2.1 Detailed Description

A publicly available class that exposes Price Spectre functionality.

7.2.2 Member Function Documentation

7.2.2.1 averageX()

```
averageX (
              $x,
              $reuseSearchResults = false )
```

Performs pre-defined x Average algorithm.

Performs a search based on the current search parameters and then performs the "x Average" algorithm. Returns the total price based on "x Average".

Warning

This generates a new search and discards the results of any prior searches generated from PS::performSearch(), PS::xLowestY(), etc. or the one provided automatically by Price Spectre if reuse↩ SearchResults is not true. This may result in usage of premium points.

Parameters

x	Specifies the number of search results to average.
reuseSearchResults	Optional. When specified and true reuse the search most recently performed When not specified or false generate a new search

Returns

The total price as computed by "x Average". Execution terminates on failure.

```
// Sets our price to y% lower than the average of the lowest X prices found.
price = ps.averageX(x) * (1 - y/100);
ps.setPrice(price);
```

7.2.2.2 debugMessage()

```
debugMessage (
              $message )
```

Adds debug message. Append message to any previously debug messages.

If this is called at anytime during execution this will become an error message and may be reported the user.

Attention

This does not interrupt execution of the script but any price changes will be ignored. To immediately terminate execution of script with an error message use throw instead.

Parameters

message	Debug message that will be returned as an error once script ends.

```
// Perform X lowest by $Y on listings.
// if x is less than or equal to 0 generate an error but continue running.
if(x <= 0)
{
  ps.debugMessage('Warning:  Invalid value for x ' + x + '.');
}
count = 0;
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
   // if this is the listing to match against
   count++;
   if(count >= x)
   {
      price = results[i].getPrice() - y;
      // if we are trying to set a price of $0 or less generate warning but continue.
      if(price <= 0)
      {
         item = results[i].getListingID();
         ps.debugMessage('Warning:  Price attempted set to ' + price + ' item = ' + item + '.');
         continue;
      }
      ps.setPrice(price);
      break;
   }
}
```

7.2.2.3 delete()

```
delete (
               $index )
```

Deletes a listing from the results set.

This function does not actually delete the listing but prevents Price Spectre from considering it in the final results (e.g. reporting, reprice history, etc.). Calling this function will affect the values returned by PS::getRealResults() and PS::getRealNumResults(). Calling this function also makes Listing::isDeleted() return TRUE.

This may be useful in situations where a search parameter is not sufficient to filter out a particular type of listing.

Warning

> Since Price Spectre is limited to retrieving the first 100 matching results from eBay, search parameters should always be used as a filter instead whenever possible.
>
> If all results are deleted Price Spectre will observe that no competitor listings were found. Deletion of a listing is not required to ignore it but may make the results easier to understand. Deletions are not carried over to subsequent searches.

Parameters

index	Specifies which result to remove from the results set.

```
// Delete all non-new listings
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
  // Delete if not new
```

```
  if(results[i].getConditionID() != 1000)
  {
     ps.delete(i);
  }
}
// Perform X lowest by $Y on remaining listings.
count = 0;
results = ps.getRealResults();
numResults = ps.getRealNumResults();
for(i = 0; i < numResults; ++i)
{
   // if this is the listing to match against
   count++;
   if(count >= x)
   {
      ps.setPrice(results[i].getPrice() - y);
      break;
   }
}
```

7.2.2.4 elastic()

```
elastic (
              $x )
```

Performs pre-defined Elastic algorithm.

Warning

This doesn't generate a search so should normally be combined with PS::overrideNoCompetitionFound().

Parameters

x	Specifies the minimum number of days to look back in evaluating the algorithm.

Returns

The total price as computed by "Elastic".

```
// Sets our price to the results of the Elastic algorithm.
price = ps.elastic(x);
ps.setPrice(price);
// ensures the price is set since no search is used
ps.overrideNoCompetitionFound();
```

7.2.2.5 getCeilingPrice()

```
getCeilingPrice ( )
```

Retrieves the total ceiling price of the seller's listing. This amount is inclusive of shipping charges unless shipping and handling charges are being ignored.

Returns

The total ceiling price.

7.2.2.6 getCurrentPrice()

```
getCurrentPrice ( )
```

Returns the current price of the listing.

Returns

The current price of the listing.

```
// Increases current price by 1%.
currentPrice = ps.getCurrentPrice();
ps.setPrice(currentPrice * 1.01);
```

7.2.2.7 getData()

```
getData ( )
```

Retrieves user data for the listing.

A string of up to 255 characters can be stored containing any user-defined data or state information.

Examples of possible data could be information regarding the current or past state of the listing or additional variables.

Returns

A string containing user data that has been saved for the listing. If no data has been stored then returns an empty string.

```
// Example of pulling 3 variables from data.
// Previously a string of three comma-separated
// values was stored in data.
// Example: 1,2,3
var data = ps.getData();    // fetches data
var arr = data.split(",");  // create array
// assign each variable x, y, and z.
x = arr[0];
y = arr[1];
z = arr[2];
if(x < 1)
   throw 'x must be positive';
if(y < 1)
   throw 'y must be positive';
if(z < 1)
   throw 'z must be positive';
```

7.2.2.8 getErrorMessage()

```
getErrorMessage ( )
```

Returns error message for last call to PS::performSearch().

Returns

Error message string if error exists. If no error message exists returns empty string.

```
// Performs search and terminates with error message on failure.
if(!ps.performSearch())
   throw ps.getErrorMessage();
```

7.2.2.9 getFloorPrice()

```
getFloorPrice ( )
```

Retrieves the total floor price of the seller's listing. This amount is inclusive of shipping charges unless shipping and handling charges are being ignored.

Returns

The total floor price.

```
// Perform X lowest by $Y on listings.
floorPrice = ps.getFloorPrice();
count = 0;
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
   // ignore this listing if it is below our floor price.
   if(results[i].getPrice() < floorPrice)
   {
      continue;
   }
   // if this is the listing to match against
   count++;
   if(count >= x)
   {
      ps.setPrice(results[i].getPrice() - y);
      break;
   }
}
```

7.2.2.10 getGoldAsk()

```
getGoldAsk (
               $currency,
               $unit )
```

Retrieves the current ask price for gold bullion.

Parameters

currency	Specifies the currency to retrieve price in. Possible values: 'USD', 'GBP', 'EUR', 'AUD', 'CAD'
unit	Specifies the mass unit. Possible values: 'kg', 'g', 'gr', 'ozt', 'troy'

Returns

The current ask price for gold bullion per the unit mass specified.

```
// Sets our price to 5% higher than gold.
// X specifies the weight in grains we are selling.
price = ps.getGoldAsk('USD', 'gr') * x * 1.05;
ps.setPrice(price);
```

7.2.2.11 getGoldBid()

```
getGoldBid (
               $currency,
               $unit )
```

Retrieves the current bid price for gold bullion.

Parameters

currency	Specifies the currency to retrieve price in. Possible values: 'USD', 'GBP', 'EUR', 'AUD', 'CAD'
unit	Specifies the mass unit. Possible values: 'kg', 'g', 'gr', 'ozt', 'troy'

Returns

The current bid price for gold bullion per the unit mass specified.

```
// Sets our price to 5% higher than gold.
// X specifies the weight in troy ounces we are selling.
price = ps.getGoldBid('USD', 'ozt') * x * 1.05;
ps.setPrice(price);
```

7.2.2.12 getLastPriceChangePrice()

```
getLastPriceChangePrice ( )
```

Retrieves the last price that the listing was set to. This should be equal to PS::getCurrentPrice();

Returns

The last price that the listing was set to.

7.2.2.13 getLastPriceChangeTime()

```
getLastPriceChangeTime ( )
```

Retrieves the epoch time of the last price change for the listing.

Returns

The epoch time of the last price change. If no price changes then returns the timestamp when Price Spectre first became aware of the listing.

```
// Reduce price by y% every x days.
now = Math.round(Date.now() / 1000);
secondsPerDay = 86400;
lastPriceChange = ps.getLastPriceChangeTime();
daysSinceChange = (now - lastPriceChange) / secondsPerDay;
currentPrice = ps.getCurrentPrice();
if(daysSinceChange >= x)
   ps.setPrice(currentPrice * (1 - y/100));
```

7.2.2.14 getLastSaleTime()

```
getLastSaleTime ( )
```

Retrieves the epoch time of the last time a sale was made for the listing.

Returns

The epoch time of the last time a sale was made for the listing. If no sales are recorded then returns 0.

```
// Reduce price by y% if sale hasn't been made in x days and price
// hasn't changed in 7 days.
now = Math.round(Date.now() / 1000);
secondsPerDay = 86400;
secondsPerXDay = secondsPerDay * x;
secondsSincePriceChange = ps.getSecondsSinceLastPriceChange();
daysSinceChange = secondsSincePriceChange / secondsPerDay;
lastSaleTime = ps.getLastSaleTime();
daysSinceLastSale = (now - lastSaleTime) / secondsPerDay;
currentPrice = ps.getCurrentPrice();
if(daysSinceLastSale >= x && daysSinceChange >= secondsPerXDay)
   ps.setPrice(currentPrice * (1 - y/100));
```

7.2.2.15 getListingID()

```
getListingID ( )
```

Retrieves the item number of the listing.

Returns

Returns the 12 digit item number of the listing.

```
// sets up WS object to communicate with another website.
var APIendpoint = 'http://test.com/v1/getItem';
ws.setEndpoint(APIendpoint);
var str = ps.getListingID();
ws.setRequestBody(str);
var response = ws.post();
ps.setPrice(response);
```

7.2.2.16 getMyAmazonPrice()

```
getMyAmazonPrice ( )
```

Checks Amazon for the seller's own Amazon price based on ASIN.

Performs an ASIN search against the seller's own Amazon account. If a product is found return its price. Otherwise, throw an exception. The ASIN is retrieved from the Product Code or Keywords search parameter. Useful for maintaining price parity.

Returns

The total price of product from the Amazon seller account. Exception thrown on failure.

Warning

Amazon does not permit searching against a specific Amazon competitor. In order for this function to work the seller must have provided Amazon MWS access to Price Spectre.

```
// Sync our eBay price to our Amazon price.
// If no Amazon listing found perform xLowestY instead
try {
   price = ps.getMyAmazonPrice();
} catch (err) {
   price = ps.xLowestY(x,y);
}
ps.setPrice(price);
```

7.2.2.17 getNoCompetitionFound()

```
getNoCompetitionFound ( )
```

Checks whether no competitors were found during last search performed.

Returns

Returns TRUE if no competitor listings were found. Returns FALSE if competitor listings were found.

```
// Set price to X lowest Y with x = 1, y = 0.01.
// But terminates with an error if no competitors were found.
price = ps.xLowestY(1, 0.01);
if(ps.getNoCompetitionFound())
{
   throw 'No competitors found';
}
```

7.2.2.18 getNumResults()

```
getNumResults ( )
```

Retrieves the number of competitor listings found. Deleted listings are included in the count.

Returns

The count of competitor listings found.

```
// Perform X lowest by $Y on listings.
count = 0;
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
   // if this is the listing to match against
   count++;
   if(count >= x)
   {
      ps.setPrice(results[i].getPrice() - y);
      break;
   }
}
```

7.2.2.19 getNumSales()

```
getNumSales (
               $start = 0,
               $end = 2147483647 )
```

Retrieves the quantity sold between two epoch times.

Parameters

start	Specifies the epoch time to begin counting sales. If not specified a default of 0 is used.
end	Specifies the epoch time to stop counting sales. If not specified a default of 2147483647 is used.

Returns

The quantity sold during the specified period of time.

```
// Reduce price by y% if x sales haven't been made in 7 days and price hasn't
// changed in 7 days.
now = Math.round(Date.now() / 1000);
secondsPerDay = 86400;
secondsPerWeek = secondsPerDay * 7;
secondsSincePriceChange = ps.getSecondsSinceLastPriceChange();
daysSinceChange = secondsSincePriceChange / secondsPerDay;
numSales = ps.getNumSales(now - secondsPerWeek);
currentPrice = ps.getCurrentPrice();
if(numSales < x && daysSinceChange >= 7)
   ps.setPrice(currentPrice * (1 - y/100));
```

7.2.2.20 getNumSalesSince()

```
getNumSalesSince (
              $seconds )
```

Retrieves the quantity sold during the past number of seconds.

Parameters

seconds	Specifies the number of seconds in the past to begin counting sales.

Returns

The quantity sold during the past number of seconds.

```
// Reduce price by y% if x sales haven't been made in 7 days and price
// hasn't changed in 7 days.
now = Math.round(Date.now() / 1000);
secondsPerDay = 86400;
secondsPerWeek = secondsPerDay * 7;
secondsSincePriceChange = ps.getSecondsSinceLastPriceChange();
daysSinceChange = secondsSincePriceChange / secondsPerDay;
numSales = ps.getNumSalesSince(secondsPerWeek);
currentPrice = ps.getCurrentPrice();
if(numSales < x && daysSinceChange >= 7)
   ps.setPrice(currentPrice * (1 - y/100));
```

7.2.2.21 getPalladiumAsk()

```
getPalladiumAsk (
              $currency,
              $unit )
```

Retrieves the current ask price for palladium bullion.

Parameters

currency	Specifies the currency to retrieve price in. Possible values: 'USD', 'GBP', 'EUR', 'AUD', 'CAD'
unit	Specifies the mass unit. Possible values: 'kg', 'g', 'gr', 'ozt', 'troy'

Returns

The current ask price for palladium bullion per the unit mass specified.

```
// Sets our price to 5% higher than palladium.
// X specifies the weight in grains we are selling.
price = ps.getPalladiumAsk('USD', 'gr') * x * 1.05;
ps.setPrice(price);
```

7.2.2.22 getPalladiumBid()

```
getPalladiumBid (
                $currency,
                $unit )
```

Retrieves the current bid price for palladium bullion.

Parameters

currency	Specifies the currency to retrieve price in. Possible values: 'USD', 'GBP', 'EUR', 'AUD', 'CAD'
unit	Specifies the mass unit. Possible values: 'kg', 'g', 'gr', 'ozt', 'troy'

Returns

The current bid price for palladium bullion per the unit mass specified.

```
// Sets our price to 5% higher than palladium.
// X specifies the weight in troy ounces we are selling.
price = ps.getPalladiumBid('USD', 'ozt') * x * 1.05;
ps.setPrice(price);
```

7.2.2.23 getPlatinumAsk()

```
getPlatinumAsk (
                $currency,
                $unit )
```

Retrieves the current ask price for platinum bullion.

Parameters

currency	Specifies the currency to retrieve price in. Possible values: 'USD', 'GBP', 'EUR', 'AUD', 'CAD'
unit	Specifies the mass unit. Possible values: 'kg', 'g', 'gr', 'ozt', 'troy'

Returns

The current ask price for platinum bullion per the unit mass specified.

```
// Sets our price to 5% higher than platinum.
// X specifies the weight in grains we are selling.
price = ps.getPlatinumAsk('USD', 'gr') * x * 1.05;
ps.setPrice(price);
```

7.2.2.24 getPlatinumBid()

```
getPlatinumBid (
               $currency,
               $unit )
```

Retrieves the current bid price for platinum bullion.

Parameters

currency	Specifies the currency to retrieve price in. Possible values: 'USD', 'GBP', 'EUR', 'AUD', 'CAD'
unit	Specifies the mass unit. Possible values: 'kg', 'g', 'gr', 'ozt', 'troy'

Returns

The current bid price for platinum bullion per the unit mass specified.

```
// Sets our price to 5% higher than platinum.
// X specifies the weight in troy ounces we are selling.
price = ps.getPlatinumBid('USD', 'ozt') * x * 1.05;
ps.setPrice(price);
```

7.2.2.25 getQuantity()

```
getQuantity ( )
```

Retrieves the quantity available of the listing.

Returns

The quantity available of the listing.

```
// Perform X Lowest By $Y algorithm
// But increases price 10x when only one remains.
quantity = ps.getQuantity();
if(quantity <= 1)
{
   price = ps.getSearchParameter('binPrice') * 10;
}
else
{
   price = ps.xLowestY(x, y);
}
ps.setPrice(price);
```

7.2.2.26 getRank()

```
getRank (
               $price )
```

Retrieves the rank achieved at a given price based on the competitors found.

Parameters

price	Specifies the price to check rank against.

Returns

The rank at a given price. Returns 0 if no competitors were found.

```
// Sets our price 0.05 lower than the lowest price.
price = ps.xLowestY(1, 0.05);
rank = ps.getRank(price);
if(rank > 1)
   throw 'FAILED:  Could only achieve a ranking of ' + rank;
ps.setPrice(price);
```

7.2.2.27 getRealNumResults()

```
getRealNumResults ( )
```

Retrieves the actual number of competitor listings found. Deleted listings are ignored in the count.

Returns

The count of competitor listings found excluding previously deleted listings.

```
// Delete all non-new listings
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
  // delete if not new
  if(results[i].getConditionID() != 1000)
  {
     ps.delete(i);
  }
}
// Perform X lowest by $Y on remaining listings.
count = 0;
results = ps.getRealResults();
numResults = ps.getRealNumResults();
for(i = 0; i < numResults; ++i)
{
   // if this is the listing to match against
   count++;
   if(count >= x)
   {
      ps.setPrice(results[i].getPrice() - y);
      break;
   }
}
```

7.2.2.28 getRealRank()

```
getRealRank (
              $price )
```

Retrieves the rank achieved at a given price based on the competitors found. Deleted listings are ignored in the count.

Parameters

price	Specifies the price to check rank against.

Returns

The rank at a given price. Returns 0 if no competitors were found.

```
// Sets our price 0.05 lower than the lowest price.
price = ps.xLowestY(1, 0.05);
rank = ps.getRealRank(price);
if(rank > 1)
   throw 'FAILED:  Could only achieve a ranking of ' + rank;
ps.setPrice(price);
```

7.2.2.29 getRealResults()

```
getRealResults ( )
```

Retrieves an array of class Listing representing the actual competitor listings found. Deleted listings are ignored.

Returns

Array of Listing objects.

```
// Delete all non-new listings
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
  // Delete if not new
  if(results[i].getConditionID() != 1000)
  {
     ps.delete(i);
  }
}
// Perform X lowest by $Y on remaining listings.
count = 0;
results = ps.getRealResults();
numResults = ps.getRealNumResults();
for(i = 0; i < numResults; ++i)
{
   // if this is the listing to match against
   count++;
   if(count >= x)
   {
      ps.setPrice(results[i].getPrice() - y);
      break;
   }
}
```

7.2.2.30 getResults()

```
getResults ( )
```

Retrieves an array of class Listing representing the actual competitor listings found.

Returns

Array of Listing objects.

```
// Perform X lowest by $Y on listings.
count = 0;
results = ps.getResults();
numResults = ps.getNumResults();
for(i = 0; i < numResults; ++i)
{
   // if this is the listing to match against
   count++;
   if(count >= x)
   {
      ps.setPrice(results[i].getPrice() - y);
      break;
   }
}
```

7.2.2.31 getSearchParameter()

```
getSearchParameter (
               $key )
```

Retrieves a specified search parameter.

Parameters

key	Specifies the search parameter to retrieve. Possible values: 'keywords', 'productCode', 'minFeedback', 'maxFeedback', 'minPrice', 'maxPrice', 'minQty', 'maxQty', 'maxHandlingTime', 'includeSellers', 'excludeSellers', 'exemptSellers', 'ebayFixedprice', 'ebayAuctions', 'findAmazon', 'domestic', 'hideDuplicates', 'trs', 'returnsAccepted', 'restrictCategory', 'searchCategory', 'searchCondition', 'binPrice', 'floorPrice', 'ceilPrice', 'postalCodeUS', 'destination', 'valueBox'

Returns

The current value of the search parameter.

```
// Retrieves the keywords specified for the listing.
keywords = ps.getSearchParameter('keywords');
```

7.2.2.32 getSecondsRemaining()

```
getSecondsRemaining ( )
```

Retrieves the number of seconds before the listing will end.

Returns

The number of seconds that before the listing will end if it was listed on eBay. Returns 2147483647, otherwise.

```
results = ps.getResults();
numResults = ps.getNumResults();
secondsPerDay = 86400;
// Delete all results that will end in less than one day.
for(i = 0; i < numResults; ++i)
{
   timeleft = results[i].getSecondsRemaining();
   if(timeleft < secondsPerDay)
   {
      ps.delete(i);
   }
}
```

7.2.2.33 getSecondsSinceLastPriceChange()

```
getSecondsSinceLastPriceChange ( )
```

Retrieves the number of seconds since the last price change.

Returns

The number of seconds since the last price change.

```
// Reduce price by y% every x days.
secondsPerDay = 86400;
secondsSincePriceChange = ps.getSecondsSinceLastPriceChange();
daysSinceChange = secondsSincePriceChange / secondsPerDay;
if(daysSinceChange >= x)
{
   currentPrice = ps.getCurrentPrice();
   ps.setPrice(currentPrice * (1 - y/100));
}
```

7.2.2.34 getSecondsSinceStart()

```
getSecondsSinceStart ( )
```

Retrieves the number of seconds since the listing has started.

Returns

The number of seconds that have elapsed since the listing started if it was listed on eBay.

```
secondsPerDay = 86400;
daysBeforeChange = 30;
secondsSinceStart = ps.getSecondsSinceStart();
// Reduce the price 0.01 every reprice cycle after listed for 30 days.
if(secondsSinceStart > (secondsPerDay * daysBeforeChange))
{
   currentPrice = ps.getCurrentPrice();
   ps.setPrice(currentPrice - 0.01);
}
```

7.2.2.35 getShippingHandling()

```
getShippingHandling ( )
```

Retrieves the shipping and handling charges for the seller's listing.

Returns

The shipping and handling charges. If shipping and handling charges are being ignored then returns 0.

7.2.2.36 getSilverAsk()

```
getSilverAsk (
              $currency,
              $unit )
```

Retrieves the current ask price for silver bullion.

Parameters

currency	Specifies the currency to retrieve price in. Possible values: 'USD', 'GBP', 'EUR', 'AUD', 'CAD'
unit	Specifies the mass unit. Possible values: 'kg', 'g', 'gr', 'ozt', 'troy'

Returns

The current ask price for silver bullion per the unit mass specified.

```
// Sets our price to 5% higher than silver.
// X specifies the weight in kilograms we are selling.
price = ps.getSilverAsk('USD', 'kg') * x * 1.05;
ps.setPrice(price);
```

7.2.2.37 getSilverBid()

```
getSilverBid (
               $currency,
               $unit )
```

Retrieves the current bid price for silver bullion.

Parameters

currency	Specifies the currency to retrieve price in. Possible values: 'USD', 'GBP', 'EUR', 'AUD', 'CAD'
unit	Specifies the mass unit. Possible values: 'kg', 'g', 'gr', 'ozt', 'troy'

Returns

The current bid price for silver bullion per the unit mass specified.

```
// Sets our price to 5% higher than silver.
// X specifies the weight in grams we are selling.
price = ps.getSilverBid('USD', 'g') * x * 1.05;
ps.setPrice(price);
```

7.2.2.38 getSiteCountry()

```
getSiteCountry ( )
```

Retrieves the country associated with the eBay site the item was listed to.

Returns

Two-letter ISO 3166 country code to indicate the country of the eBay site where the item is listed (e.g., "US" for the eBay.com or "GB" for the eBay.co.uk).

```
results = ps.getResults();
numResults = ps.getNumResults();
myCountry = ps.getSiteCountry();
// Delete all international results.
for(i = 0; i < numResults; ++i)
{
   theirCountry = results[i].getLocation();
   if(myCountry != theirCountry)
   {
      ps.delete(i);
   }
}
```

7.2.2.39 overrideNoCompetitionFound()

```
overrideNoCompetitionFound ( )
```

Tells Price Spectre to not apply its own logic during the "No competition found" situation.

By default Price Spectre will override any prices set if the last search performed results in no competitor listings being found. The price set can be configured via the "No competition found" behavior.

Make a call to this function if you wish to provide your own logic for handling this situation and prevent Price Spectre from applying its own. This can be called at any time.

```
// Set price to X lowest Y with x = 1, y = 0.01.
// But decrease price 5% if no results.
price = ps.xLowestY(1, 0.01);
if(ps.getNoCompetitionFound())
{
   price = ps.getCurrentPrice() * 0.95;
}
ps.setPrice(price);
ps.overrideNoCompetitionFound();
```

7.2.2.40 performSafeSearch()

```
performSafeSearch ( )
```

Performs a search based on the given search parameters.

This function differs from PS::performSearch() in that if a failure occurs execution of algorithm will immediately terminate with the appropriate error message.

After the search is performed make a call to PS::getResults() and PS::getNumResults() to retrieve the search results and number of results found.

Warning

This generates a new search and discards the results of any prior searches generated from PS::performSearch(), PS::xLowestY(), etc. or the one provided automatically by Price Spectre. This may result in usage of premium points.

Returns

TRUE on success. Execution terminates on failure.

7.2.2.41 performSearch()

```
performSearch ( )
```

Performs a search based on the given search parameters.

After the search is performed make a call to PS::getResults() and PS::getNumResults() to retrieve the search results and number of results found.

The user must make their own accommodations in the case that the search fails. See PS::performSafeSearch() for an identical function that doesn't have this requirement.

Warning

This generates a new search and discards the results of any prior searches generated from PS::performSearch(), PS::xLowestY(), etc. or the one provided automatically by Price Spectre. This may result in usage of premium points.

Returns

TRUE on success or FALSE on search failure. Make call to PS::getErrorMessage() to retrieve error messages.

```
// Performs search and terminates on failure.
if(!ps.performSearch())
   throw ps.getErrorMessage();
```

7.2.2.42 performSelfSafeSearch()

```
performSelfSafeSearch ( )
```

Performs a self search on Amazon.

This function differs from PS::performSelfSearch() in that if a failure occurs execution of algorithm will immediately terminate with the appropriate error message.

After the search is performed make a call to PS::getResults() and PS::getNumResults() to retrieve the search results and number of results found.

Warning

This generates a new search and discards the results of any prior searches generated from PS::performSearch(), PS::xLowestY(), etc. or the one provided automatically by Price Spectre. This may result in usage of premium points.

Returns

TRUE on success. Execution terminates on failure.

```
// Sync our eBay price to our Amazon price.
ps.performSelfSafeSearch();
results = ps.getResults();
numResults = ps.getNumResults();
if(numResults >= 1)
{
   price = results[0].getPrice();
   ps.setPrice(price);
}
```

7.2.2.43 performSelfSearch()

```
performSelfSearch ( )
```

Performs a self search on Amazon.

After the search is performed make a call to PS::getResults() and PS::getNumResults() to retrieve the search results and number of results found.

The user must make their own accommodations in the case that the search fails. See PS::performSelfSafeSearch() for an identical function that doesn't have this requirement.

Warning

This generates a new search and discards the results of any prior searches generated from PS::performSearch(), PS::xLowestY(), etc. or the one provided automatically by Price Spectre. This may result in usage of premium points.

Returns

TRUE on success or FALSE on search failure. Make call to PS::getErrorMessage() to retrieve error messages.

```
// Sync our eBay price to our Amazon price.
// Performs search and terminates on failure.
if(!ps.performSelfSearch())
   throw ps.getErrorMessage();
results = ps.getResults();
numResults = ps.getNumResults();
if(numResults >= 1)
{
   price = results[0].getPrice();
   ps.setPrice(price);
}
```

7.2.2.44 setPrice()

```
setPrice (
              $price )
```

Sets the total price of the listing.

If this function is not called within the algorithm the final price will remain at its current level.

Warning

The listing's final price may or may not be equal to the price specified. Once your algorithm finishes Price Spectre may perform additional changes to ensure the final price fits within the constraints of your floor and ceiling prices along with performing Bat'a pricing.

By default if no competitor listings are found with the last search performed then the final price would also be overridden based on "No competition found" behavior that was set. This can be avoided by making a call to PS::overrideNoCompetitionFound().

Parameters

price	Specifies the total price to set the listing to.

Returns

The current price of the listing.

```
// Increases current price by 1%.
currentPrice = ps.getCurrentPrice();
ps.setPrice(currentPrice * 1.01);
```

7.2.2.45 setSearchParameter()

```
setSearchParameter (
              $key,
              $value )
```

Sets a specified search parameter.

Parameters

key	Specifies the search parameter to set. Possible values: 'keywords', 'productCode', 'minFeedback', 'maxFeedback', 'minPrice', 'maxPrice', 'minQty', 'maxQty', 'maxHandlingTime', 'includeSellers', 'excludeSellers', 'exemptSellers', 'ebayFixedprice', 'ebayAuctions', 'findAmazon', 'domestic', 'hideDuplicates', 'trs', 'returnsAccepted', 'restrictCategory', 'searchCategory', 'searchCondition', 'binPrice', 'floorPrice', 'ceilPrice', 'postalCodeUS', 'destination', 'valueBox'
value	Specifies the new value for the search parameter.

Returns

FALSE on failure. TRUE on success.

```
// Disables TRS filter and retries search if no competition found.
price = ps.xLowestY(x,y);
// if no competitor listings found
if(ps.getNoCompetitionFound())
{
  trs = ps.getSearchParameter('trs');
  // if we had TRS set unset and make us 5th lowest price.
  if(trs)
  {
     ps.setSearchParameter('trs', false);
     price = ps.xLowestY(5, 0.01);
  }
}
ps.setPrice(price);
```

7.2.2.46 writeData()

```
writeData (
              $str )
```

Writes user data to the database for the listing.

A string of up to 255 characters can be stored containing any user-defined data or state information.

Examples of possible data could be information regarding the current or past state of the listing or additional variables.

Parameters

str	Specifies a string to store as data for the listing.

Returns

TRUE on success or FALSE on failure.

```
// Example of writing an array to the database.
var arr = [1, 0.01, 3];    // our array which represents three variables.
var str = arr.toString();  // a string representation of that array.
ps.writeData(str);         // write the data to the DB
```

7.2.2.47 xLowestY()

```
xLowestY (
              $x,
              $y,
              $reuseSearchResults = false )
```

Performs pre-defined "x lowest by $y" algorithm

Performs a search based on the current search parameters and then performs the "x lowest by $y" algorithm. Returns the total price based on "x lowest by $y".

Warning

This generates a new search and discards the results of any prior searches generated from PS::performSearch(), PS::xLowestY(), etc. or the one provided automatically by Price Spectre if reuse↩ SearchResults is not true. This may result in usage of premium points.

Parameters

x	Specifies which search result to match against. ex: 1 matches against the lowest price, 2 matches against the second lowest price, etc.
y	Specifies the amount to discount the Xth lowest price. Positive values of Y result in pricing below the Xth lowest price. A zero value of Y results in matching the Xth lowest price. Negative values of Y result in pricing above the Xth lowest price.
reuseSearchResults	Optional. When specified and true reuse the search most recently performed When not specified or false generate a new search

Returns

The total price as computed by "x lowest by $y". Execution terminates on failure.

```
// Sets our price 0.05 lower than the lowest price.
price = ps.xLowestY(1, 0.05);
ps.setPrice(price);
```

7.2.2.48 xLowestYPercent()

```
xLowestYPercent (
              $x,
              $y,
              $reuseSearchResults = false )
```

Performs pre-defined "x lowest y%" algorithm.

Performs a search based on the current search parameters and then performs the "x lowest y%" algorithm. Returns the total price based on "x lowest y%".

Warning

This generates a new search and discards the results of any prior searches generated from PS::performSearch(), PS::xLowestY(), etc. or the one provided automatically by Price Spectre. This may result in usage of premium points.

Parameters

x	Specifies which search result to match against. ex: 1 matches against the lowest price, 2 matches against the second lowest price, etc.
y	Specifies the percentage to discount the Xth lowest price. Positive values of Y result in pricing below the Xth lowest price. A zero value of Y results in matching the Xth lowest price. Negative values of Y result in pricing above the Xth lowest price.

Returns

The total price as computed by "x lowest y%". Execution terminates on failure.

```
// Sets our price 5% lower than the lowest price.
price = ps.xLowestYPercent(1, 5);
ps.setPrice(price);
```

Index