Point of Sale – Developers Manual

Version
2/9/2020

Table of Contents

Class Index 2
Class Documentation 3
CartItem< T > 3
PointOfSale 7
Index 13

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

CartItem< T > (Implements the logic of a single item in the cart) 3

PointOfSale (Provides API for interacting with the Point of Sale system to compute the Pre Tax cost for shopping cart of items) 7

Class Documentation

CartItem< T > Class Template Reference

Implements the logic of a single item in the cart.
`#include <CartItem.h>`

Public Member Functions

- ReturnCode_t **setPrice** (double price)
  Allows for setting price for the item in the cart.

- ReturnCode_t **applyMarkdown** (double amount)
  Allows for setting a reduction in the price of the item.

- ReturnCode_t **applyGetXforPriceDiscount** (T buy_amount, double price)
  Allows for discounts where the customer is able to buy multiples of an item for a single lower price.

- ReturnCode_t **applyGetXforPriceDiscount** (T buy_amount, double price, T limit)
  Allows for discounts where the customer is able to buy multiples of an item for a single lower price.

- ReturnCode_t **applyBuyXGetYDiscount** (T buy_x, T get_y, double percent_off)
  Allows for discounts where the customer is able to get items at a discounted rate if other qualifying purchases are performed.

- ReturnCode_t **applyBuyXGetYDiscount** (T buy_x, T get_y, double percent_off, T limit)
  Allows for discounts where the customer is able to get items at a discounted rate if other qualifying purchases are performed.

- ReturnCode_t **addToCart** (T amount)
  Allows for adding items, or pounds of a good, to the shopping cart.

- ReturnCode_t **removeFromCart** (T amount)
  Allows for removing items, or pounds, from the shopping cart.

- ReturnCode_t **computePreTax** (double *pTaxAmount)
  Calculates the pre-tax cost of the item.

Detailed Description

template<class T>

class CartItem< T >

Implements the logic of a single item in the cart.

The **PointOfSale** class maintains a **CartItem** for each SKU within the cart. This class is utilized to handle both fixed price items and weight based items. The use of templates makes it possible to re-use all code associated with all items. The only difference in the logic is whether a whole number of items is maintained or a floating point weight.

Member Function Documentation

template<class T > ReturnCode_t CartItem< T >::addToCart (T *amount*)

Allows for adding items, or pounds of a good, to the shopping cart.

The **PointOfSale** system supports fixed price and weight based items. These items are then added to the cart to be purchased. This function provides the mechanism by which items are added to the cart to be purchased.

Parameters

amount	The number of item, or pounds, that should be added to the cart

template<class T > ReturnCode_t CartItem< T >::applyBuyXGetYDiscount (T *buy_x*, T *get_y*, double *percent_off*)

Allows for discounts where the customer is able to get items at a discounted rate if other qualifying purchases are performed.

This discount allows a customer to purchase a specified number of items at a discounted rate. To get this benefit, the customer must purchase the specified number of items at full price. For instance, a discount such as "Buy 5 Cookies at full price and get the next 2 at %50 off" is possible with this function.

Parameters

buy_x	The number of items, or pounds, that must be purchased at full price
get_y	The number of items, or pounds, that are able to be purchased at a discounted rate
percent_off	The amount of savings given to the customer, must be between 0 and 1

template<class T > ReturnCode_t CartItem< T >::applyBuyXGetYDiscount (T *buy_x*, T *get_y*, double *percent_off*, T *limit*)

Allows for discounts where the customer is able to get items at a discounted rate if other qualifying purchases are performed.

This discount allows a customer to purchase a specified number of items at a discounted rate. To get this benefit, the customer must purchase the specified number of items at full price. For instance, a discount such as "Buy 5 Cookies at full price and get the next 2 at %50 off" is possible with this function.

Parameters

buy_x	The number of items, or pounds, that must be purchased at full price
get_y	The number of items, or pounds, that are able to be purchased at a discounted rate
percent_off	The amount of savings given to the customer, must be between 0 and 1
limit	Allows for placing a limit on the number of items that an be purchased with this discount

template<class T > ReturnCode_t CartItem< T >::applyGetXforPriceDiscount (T *buy_amount*, double *price*)

Allows for discounts where the customer is able to buy multiples of an item for a single lower price.

This discount allows a customer to buy multiple items for a single given price. For instance, this function would allow for the creation of a discount such as "Buy 5 Chips for the only 3.99" event though the chips price may be 1.00 for each.

Parameters

buy_amount	The number of items, or pounds, that the customer is allowed for the specified price
price	The cost for acquiring all the items, or pounds

template<class T > ReturnCode_t CartItem< T >::applyGetXforPriceDiscount (T *buy_amount*, double *price*, T *limit*)

Allows for discounts where the customer is able to buy multiples of an item for a single lower price.

This discount allows a customer to buy multiple items for a single given price. For instance, this function would allow for the creation of a discount such as "Buy 5 Chips for the only 3.99" event though the chips price may be 1.00 for each. It's possible to limit the number of items that a customer is allowed to purchase using the discount. This function allows for the specification of a limit that the customer is not allowed to go beyond.

Parameters

buy_amount	The number of items, or pounds, that the customer is allowed for the specified price
price	The cost for acquiring all the items, or pounds
limit	The maximum number of items, or pounds, that the customer is able to buy using the discount

template<class T > ReturnCode_t CartItem< T >::applyMarkdown (double *amount*)

Allows for setting a reduction in the price of the item.

Each item in the cart can be configured with a marked down price to support specials. The marked down value is subtracted from the configured price before all computations are performed on the cost. In the event of a discount being applied, the marked down price for an item will be utilized rather than the original price.

Parameters

amount	Amount to take off the normal price for the special

template<class T > ReturnCode_t CartItem< T >::computePreTax (double * *pTaxAmount*)

Calculates the pre-tax cost of the item.

This function will compute the pre-tax cost of the item for the customer. This calcution will take into affect any markdowns and/or discounts that have been applied.

Parameters

pTaxAmount	Location that the computed pre-tax figure should be stored

template<class T > ReturnCode_t CartItem< T >::removeFromCart (T *amount*)

Allows for removing items, or pounds, from the shopping cart.

Tellers sometimes make mistakes when checking out a customer. When this happens, it becomes necessary to remove items from the shopping cart. This function provides the ability to remove those items, or weight, from the cart.

Parameters

amount	The number of items, or pounds, that should be removed

template<class T > ReturnCode_t CartItem< T >::setPrice (double *price*)

Allows for setting price for the item in the cart.

Each item in the cart has a price that is defined. This price will either apply to a single item or to a pound of the item in the cart.

Parameters

price	Cost per item/pound of the item

The documentation for this class was generated from the following files:

- E:/Nextcloud/Documents/repos/point-of-sale/src/CartItem.h
- E:/Nextcloud/Documents/repos/point-of-sale/src/CartItem.cpp

PointOfSale Class Reference

Provides API for interacting with the Point of Sale system to compute the Pre Tax cost for shopping cart of items.

```
#include <PointOfSale.h>
```

Public Member Functions

- ReturnCode_t **setMarkdown** (std::string sku, double price)
  Provides ability to enable a marked down price on an item.

- double **getPreTaxTotal** ()
  Calculates the pre-tax total for all items within cart.

- ReturnCode_t **setItemPrice** (std::string sku, double price)
  Provides ability to setup a fixed price for a SKU.

- ReturnCode_t **setPerPoundPrice** (std::string sku, double price)
  Provides ability to setup a per pound based price for a SKU.

- ReturnCode_t **addToCart** (std::string sku, int count)
  Adds fixed price items to the cart.

- ReturnCode_t **addToCart** (std::string sku, double weight)
  Adds weight to an item in the cart.

- ReturnCode_t **removeFromCart** (std::string sku, int count)
  Removes fixed price items from cart.

- ReturnCode_t **removeFromCart** (std::string sku, double weight)
  Removes portion of weight based item from shopping cart.

- ReturnCode_t **applyGetXForYDiscount** (std::string sku, int buy_x, double amount)
  Applys a buy X items for the Z price.

- ReturnCode_t **applyGetXForYDiscount** (std::string sku, int buy_x, double amount, int limit)
  Applys a buy X items for the Z price.

- ReturnCode_t **applyBuyXGetYAtDiscount** (std::string sku, int buy_x, int get_y, double percent_off)
  Applies a discount that allows the customer to purchae items at discounted rate after purchasing a qualified number of items.

- ReturnCode_t **applyBuyXGetYAtDiscount** (std::string sku, int buy_x, int get_y, double percent_off, int limit)
  Applies a discount that allows the customer to purchae items at discounted rate after purchasing a qualified number of items.

- ReturnCode_t **applyBuyXGetYAtDiscount** (std::string sku, double buy_x, double get_y, double percent_off)
  Applies a discount that allows the customer to purchae items at discounted rate after purchasing a qualified number of items.

- ReturnCode_t **applyBuyXGetYAtDiscount** (std::string sku, double buy_x, double get_y, double percent_off, double limit)
  Applies a discount that allows the customer to purchae items at discounted rate after purchasing a qualified number of items.

Detailed Description

Provides API for interacting with the Point of Sale system to compute the Pre Tax cost for shopping cart of items.

The **PointOfSale** class serves as the external interface for the Point Of Sale system. Each instantiation of the **PointOfSale** class represents a single checkout instance with a given customer. All operations needing to be performed by the cashier are supported through exported API's from this class.

Items being added, or scanned, into this interface can be done based on a fixed price or by weight. The **PointOfSale** object will keep track of the number of items that have been scanned of each time and correctly compute the total pre-tax cost. The API also supports the ability to remove items that have been scanned in error as well. Once removed, the items will no longer be figured into the total pre-tax cost for the items in the cart.

The **PointOfSale** class provides methods for applying discounts to items that are added to the shopping cart. There are two variants of discounts that can be applied to items in the cart. The first discount is where the customer buys a specified amount of of an item and they get a discounted rate on a specified number of items. For sale, the customer may get 5.00 lbs of beef at 75% off if 10 pounds are purchased at full price. This discount can apply to both fixed price items as well as items sold by the pound. The second discount is a bundling where the customer gets X number of items for a set price. Each of the discounts also support a limit whereby the customer is now allowed to get the discount on more than the specified number of items, or pounds.

Member Function Documentation

ReturnCode_t PointOfSale::addToCart (std::string *sku*, double *weight*)

Adds weight to an item in the cart.

This function adds the specified amount of the SKu to the shopping cart. The provided SKU must have already been configured as a weight based item via the setPerPoundPrice function. Multiple calls can be made to this function for a given SKU. The amounts for all calls will be added when the total is calculated.

Parameters

sku	Represents the item that is being added
pound	Amount of the item that should be added to the cart, in pounds

ReturnCode_t PointOfSale::addToCart (std::string *sku*, int *count*)

Adds fixed price items to the cart.

This function adds the specified amount of the SKU to the shopping cart. The provided SKU must have already been configured with a fixed price before it can be added to the cart. Multiple copies that item can be added to the cart at once.

Parameters

sku	Represents the item that is being added
pound	Amount of the item that should be added to the cart

ReturnCode_t PointOfSale::applyBuyXGetYAtDiscount (std::string *sku*, double *buy_x*, double *get_y*, double *percent_off*)

Applies a discount that allows the customer to purchae items at discounted rate after purchasing a qualified number of items.

The function allows for registering a discount that allows the customer to purchase items at a discounted rate given that they have purchased a qualifying number of items. For instance, a discount of this form may say something like "Buy 5 Cookies and Get 2 more at 70% off". A limit can be applied to the number f items that can be purchased with this discount

Parameters

sku	Represents the item that is being added
buy_x	The number of pounds that must be purchased at full price to receive discount
buy_y	The number of pounds that customer is allowed to buy at discounted rate
percent_off	The percentage of discount on the y pounds. Must be between 0 and 1.0

ReturnCode_t PointOfSale::applyBuyXGetYAtDiscount (std::string *sku*, double *buy_x*, double *get_y*, double *percent_off*, double *limit*)

Applies a discount that allows the customer to purchae items at discounted rate after purchasing a qualified number of items.

The function allows for registering a discount that allows the customer to purchase items at a discounted rate given that they have purchased a qualifying number of items. For instance, a discount of this form may say something like "Buy 5 Cookies and Get 2 more at 70% off". A limit can be applied to the number f items that can be purchased with this discount

Parameters

sku	Represents the item that is being added
buy_x	The number of pounds that must be purchased at full price to receive discount
buy_y	The number of pounds that customer is allowed to buy at discounted rate
percent_off	The percentage of discount on the y pounds. Must be between 0 and 1.0
limit	The number of pounds that are allowed to be purchased with this discount

ReturnCode_t PointOfSale::applyBuyXGetYAtDiscount (std::string *sku*, int *buy_x*, int *get_y*, double *percent_off*)

Applies a discount that allows the customer to purchae items at discounted rate after purchasing a qualified number of items.

The function allows for registering a discount that allows the customer to purchase items at a discounted rate given that they have purchased a qualifying number of items. For instance, a discount of this form may say something like "Buy 5 Cookies and Get 2 more at 70% off".

Parameters

sku	Represents the item that is being added
buy_x	The number of items that must be purchased at full price to receive discount
buy_y	The number of items that customer is allowed to buy at discounted rate
percent_off	The percentage of discount on the y items. Must be between 0 and 1.0

ReturnCode_t PointOfSale::applyBuyXGetYAtDiscount (std::string *sku*, int *buy_x*, int *get_y*, double *percent_off*, int *limit*)

Applies a discount that allows the customer to purchae items at discounted rate after purchasing a qualified number of items.

The function allows for registering a discount that allows the customer to purchase items at a discounted rate given that they have purchased a qualifying number of items. For instance, a discount of this form may say something like "Buy 5 Cookies and Get 2 more at 70% off". A limit can be applied to the number f items that can be purchased with this discount

Parameters

sku	Represents the item that is being added
buy_x	The number of items that must be purchased at full price to receive discount
buy_y	The number of items that customer is allowed to buy at discounted rate
percent_off	The percentage of discount on the y items. Must be between 0 and 1.0
limit	The number of items that are allowed to be purchased with this discount

ReturnCode_t PointOfSale::applyGetXForYDiscount (std::string *sku*, int *buy_x*, double *amount*)

Applys a buy X items for the Z price.

The following function allows for the application of a discount in which the customer is allowed to buy x items for a bundled price. For instance, the customer may be allowed to buy 4 bags of chips for 5.00.

Parameters

sku	Represents the item that is being added
buy_x	Number of items that must be purchased for discount to apply
amount	Cost for purchasing the number of specified items

ReturnCode_t PointOfSale::applyGetXForYDiscount (std::string *sku*, int *buy_x*, double *amount*, int *limit*)

Applys a buy X items for the Z price.

The following function allows for the application of a discount in which the customer is allowed to buy x items for a bundled price. For instance, the customer may be allowed to buy 4 bags of chips for 5.00. This function allows for placing a limit on the number of items in which this discount may apply.

Parameters

sku	Represents the item that is being added
buy_x	Number of items that must be purchased for discount to apply
amount	Cost for purchasing the number of specified items

double PointOfSale::getPreTaxTotal ()

Calculates the pre-tax total for all items within cart.

The **PointOfSale** system keep track of the items in the shopping cart. The items in the cart and any assoicated discounts and/or markdowns are taken into account when calculating the total cost of the cart.

ReturnCode_t PointOfSale::removeFromCart (std::string *sku*, double *weight*)

Removes portion of weight based item from shopping cart.

To account for possible mistakes that can take place at the register, the **PointOfSale** system supports the ability to remove items that may have been scanned in error. This function handles the process of removing a portion of a weight based item that was scanned in error. Once removed, the weight will no longer be factored into the calculation of the pre-tax price.

Parameters

sku	Represents the item that is being added
weight	Amount of the item that should be removed from the cart, in pounds

ReturnCode_t PointOfSale::removeFromCart (std::string *sku*, int *count*)

Removes fixed price items from cart.

To account for possible mistakes that can take place at the register, the **PointOfSale** system supports the ability to remove items that may have been scanned in error. This function handles the process of removing the specified number of items from the cart. The items must have already been added to the cart or this function will return an error.

Parameters

sku	Represents the item that is being added
count	The number of items that need to removed from the cart

ReturnCode_t PointOfSale::setItemPrice (std::string *sku*, double *price*)

Provides ability to setup a fixed price for a SKU.

The **PointOfSale** class supports fixed price and weight based items being added to the cart. The fixed price items are a set price for each item. This function provides the ability to configure an item to be considered a fixed price item. In addition, the price for each unit of the item is configured via this API. The price of the item can be updated up until a item of that type has been added to the cart. No further updates will be allowed at that point. Once an item has been marked as a fixed price item then it can no longer be marked as a weight based item.

Parameters

sku	Represents the item that is being added
price	The price per unit of the item within the cart

ReturnCode_t PointOfSale::setMarkdown (std::string *sku*, double *price*)

Provides ability to enable a marked down price on an item.

The **PointOfSale** system supports the ability to register an item with a marked down price. In this situation, the price of a fixed price item will be reduced by the amount. For a weight based item, the markdown price will be applied to each pound of the iem that is sold.

Parameters

sku	Represents the itemt hat is being added
price	Amount of discount to apply to the item

ReturnCode_t PointOfSale::setPerPoundPrice (std::string *sku*, double *price*)

Provides ability to setup a per pound based price for a SKU.

The **PointOfSale** class supports fixed price and weight based items being added to the cart. The weight based, or per pound, are a set price per pound of the item. As such, the more of an item that is purchased the more it will cost. The price per pound of a given SKU is configured via this API. All configuration of prices should be performed before items are scanned into the system. The price of the item can be updated up until a item of that type has been added to the cart. No further updates will be allowed at that point. Once an item has been marked as a fixed price item then it can no longer be marked as a weight based item, or vice versa.

Parameters

sku	Represents the item that is being added
price	The price per pound of the unit being added to the cart

---

The documentation for this class was generated from the following files:

- E:/Nextcloud/Documents/repos/point-of-sale/src/PointOfSale.h
- E:/Nextcloud/Documents/repos/point-of-sale/src/PointOfSale.cpp

Index

addToCart
  CartItem< T >, 4
  PointOfSale, 8
applyBuyXGetYAtDiscount
  PointOfSale, 9, 10
applyBuyXGetYDiscount
  CartItem< T >, 4
applyGetXforPriceDiscount
  CartItem< T >, 5
applyGetXForYDiscount
  PointOfSale, 10
applyMarkdown
  CartItem< T >, 5
CartItem< T >, 3
  addToCart, 4
  applyBuyXGetYDiscount, 4
  applyGetXforPriceDiscount, 5
  applyMarkdown, 5
  computePreTax, 5
  removeFromCart, 6
  setPrice, 6
computePreTax
  CartItem< T >, 5
getPreTaxTotal
  PointOfSale, 10
PointOfSale, 7
  addToCart, 8
  applyBuyXGetYAtDiscount, 9, 10
  applyGetXForYDiscount, 10
  getPreTaxTotal, 10
  removeFromCart, 11
  setItemPrice, 11
  setMarkdown, 11
  setPerPoundPrice, 12
removeFromCart
  CartItem< T >, 6
  PointOfSale, 11
setItemPrice
  PointOfSale, 11
setMarkdown
  PointOfSale, 11
setPerPoundPrice
  PointOfSale, 12
setPrice
  CartItem< T >, 6