

UltraCart Advanced JavaScript Checkout API Developers Guide

Written by UC Pro Services

Copyright© 2009-2010 BPS Info Solutions, Inc.

Last Updated 12/15/2010

Contents

UltraCart Advanced JavaScript Checkout API Developers Guide	1
Background	6
Prerequisite skills for using the API	7
Support	7
Setting up the API on your page.....	7
Object Model.....	8
Cart	9
CartCoupon	11
CartItem	11
CartItemAttribute	12
CartItemMultimedia	12
CartItemMultimediaThumbnail.....	12
CartItemOption	13
CartItemOptionValue.....	13
CartKitComponentOption	14
CheckoutHandoffResult	14
CheckoutTerms.....	14
CustomerProfile.....	15
CustomerProfileAddress	16
Distance.....	16
GiftSettings.....	16
GiftWrap.....	17
Item.....	17
ItemAttribute.....	18
ItemMultimedia.....	18
ItemMultimediaThumbnail	18
ItemOption	19
ItemOptionValue	19
ShippingEstimate	20
Weight.....	20
API Methods.....	21

addItems	21
applyCoupon	21
applyGiftCertificate.....	22
backgroundUpdateCart.....	22
clearFinalizeAfter	23
clearItems.....	23
checkoutHandoff	24
checkoutHandoffOnCustomSSL.....	25
createCart.....	25
establishCustomerProfile.....	26
establishCustomerProfileImmediately	26
estimateShipping	27
getAdvertisingSources	27
getAllowedCountries	28
getCart	29
getCartItemInstance	29
getCartItemMultimediaThumbnail.....	30
getCheckoutTerms.....	30
getCustomerProfile.....	31
getGiftSettings.....	31
getItem.....	32
getItemMultimediaThumbnail	32
getItems	33
getParameterValues	33
getParameterValues	34
getRelatedItems	34
getReturnPolicy	35
getStateProvinces.....	35
getStateProvinceCodes	36
getTaxCounties	36
googleCheckoutHandoff	37
googleCheckoutHandoffOnCustomSSL.....	37

initializeCheckoutApi	38
logError	38
loginCustomerProfile	39
logoutCustomerProfile.....	39
paypalHandoff	40
paypalHandoffOnCustomSSL.....	40
removeCoupon	41
removeGiftCertificate	41
removeItem.....	41
removeItems	42
resetCustomerProfilePassword	42
setFinalizeAfter.....	43
subscribeToAutoResponder	43
updateCart	44
updateCustomerProfile.....	44
updateItems	45
validate.....	45
validateAll.....	46
validateGiftCertificate.....	46
Making an Asynchronous Call	47
Global Variables.....	48
Cookie Issues	48
Building a Custom Upsell After Sequence.....	48
XHR Issues and the Relay Script	49
Automatic Updating of the Global Cart Variable	50
Call Logging and Javascript Error Handling	50
Implementing buySAFE on a Custom Checkout	50
Handling Reloads with Error Messages	53
Performance Considerations.....	53
Using JQuery with the Mootools Framework	53
Working with JSON Dates	54

Background

Through the many years that UltraCart has been around we have found the customers have always wanted a wide variety of features in their checkout. UltraCart has been built to accommodate almost all the requests people have come upon with, but they're represents in a standard way. The regular checkout is at least six pages long and the single page checkout is three pages long including the receipt and in progress page. This is necessary to accommodate all the different business rules associated with the large numbers of features.

We found that most merchants would do a nice job branding up their checkout and choose either the single page checkout or regular checkout based upon their business scenario. We then started noticing that a new group of merchants started forming. These merchants either tried to heavily manipulate the screens on the checkout using CSS and JavaScript or created their own custom forms on their website that posted the information into the checkout process. There are two main drawbacks to this style of customization. For the merchants that used CSS and JavaScript to customize the checkout there always were inherent limitations as to how far they could take it. For merchants that built custom forms on their website they always dumped the customer into the normal checkout if there were any errors associated with the form.

There had to be a better way to provide the merchant near complete flexibility to build whatever checkout best suits their business. Certain custom checkouts increase sales 15-20%. When that means six or seven figure increases in sales it's not good business sense to not pursue building one. The ROI on the development time required to build one can quickly be recorded. Our solution is the JavaScript Checkout API.

The JavaScript Checkout API is a lightweight API that allows the web page to control the UltraCart shopping cart process. Merchants can develop almost any kind of checkout using the API. The API is built around popular cross-browser compatible frameworks like MooTools and JQuery to make sure it operates robustly and efficiently.

Prerequisite skills for using the API

In order to program against this API you need to be very familiar with javascript programming. This is not for people that have never coded AJAX style javascript before. You also need to be familiar with Javascript Object Notation (JSON).

Support

Due to the complex nature of implanting custom checkouts, all support related to building or troubleshooting a custom checkout is considered premium support which has an hourly cost associated with it. All questions related to using this API are beyond the scope of regular free email or phone support. You should factor this in when considering whether to build a custom checkout.

Setting up the API on your page

The first thing you need to do in order to use the API is include the proper version of MooTools and the API itself. Next you need to initial the merchant ID that is going to be used with the API. Below is a sample snippet of code that you would place in the page. This should happen in the <HEAD> tag of your page.

```
<script type="text/javascript" src="https://secure.ultracart.com/js/mootools-1.2-core-yc.js"></script>
<script type="text/javascript" src="https://secure.ultracart.com/checkoutapi/checkoutapi.js"></script>
<script type="text/javascript">
  initializeCheckoutAPI("MID");
</script>
```

Object Model

There are lots of simple objects that comprise the API. Below we will list out the names of each object and describe what it is. Then on subsequent pages we will detail all the fields on the object.

Class	Description
Cart	Represents a shopping cart.
CartChangeResult	Object returned after a cart change. Contains the cart plus any errors.
CartCoupon	Represents a coupon that has been applied to a shopping cart
CartItem	An item in the shopping cart.
CartItemAttribute	The attributes configure on the item that is in the shopping cart.
CartItemOption	The option that is associated with an item.
CartItemOptionValue	The values in a drop down list or radio button type option.
CartKitComponentOption	If an item is a kit and the component has an option then it's represented with this object.
CheckoutHandoffResult	The result object returned when you hand off the cart into the UltraCart checkout.
CustomerProfile	A customer profile that can be associated with a shopping cart.
CustomerProfileAddress	An address in the customer profile's billing or shipping address book.
Distance	A object to represent a distance. Used on the item dimensions.
GiftSettings	All the settings associated with gift giving on the store.
GiftWrap	A wrapping papers configuration details.
Item	Represents an item configure on the store.
ItemAttribute	The attributes configured on the item.
ItemMultimedia	The multimedia configured on the item.
ItemMultimediaThumbnail	The thumbnail of a multimedia object.
ShippingEstimate	A shipping estimate returned from UltraCart for a shopping cart
Weight	An object to represent a weight. Used on the item weight.

Cart

This object represents a complete shopping cart session of the customer.

Field	Description
String cartId	The unique ID for the cart. This needs to be stored as a cookie on the customer's browser so that the cart can be retrieved.
String paymentMethod	Type of payment method. See the PAYMENT_METHOD_ constants in the checkoutapi.js file for valid values.
String creditCardType	Type of credit card. See the CREDIT_CARD_TYPE_ constants in the checkoutapi.js file for valid values.
String creditCardNumber	Credit card number
integer creditCardExpirationMonth	Credit card expiration month 1 = January 12 = December
integer creditCardExpirationYear	Credit card expiration year. Must be a four digit year.
String creditCardVerificationNumber	Credit card verification number 4 digits for American Express and 3 digits for all other types of credit cards.
boolean collectCreditCardVerificationNumber	True if the CVV2 should be collected for this merchant.
String purchaseOrderNumber	Purchase order number
String billToFirstName	Bill to first name
String billToLastName	Bill to last name
String billToTitle	Bill to title
String billToCompany	Bill to company
String billToAddress1	Bill to address line 1
String billToAddress2	Bill to address line 2
String billToCity	Bill to city
String billToState	Bill to state
String billToPostalCode	Bill to postal code
String billToCountry	Bill to country. Must be a valid country name from the getAllowedCountries() API call.
String billToDayPhone	Bill to day phone
String billToEveningPhone	Bill to evening phone
String email	Email address
boolean mailingListOptIn	Whether the customers wants to receive news and special offers via email.
String shipToFirstName	Ship to first name
String shipToLastName	Ship to last name
String shipToTitle	Ship to title
String shipToCompany	Ship to company
String shipToAddress1	Ship to address line 1
String shipToAddress2	Ship to address line 2
String shipToCity	Ship to city
String shipToState	Ship to state
String shipToPostalCode	Ship to postal code
String shipToCountry	Ship to country. Must be a valid country name from the getAllowedCountries() API call.
String shipToPhone	Ship to phone

String shippingMethod	Shipping method
boolean needShipping	true if the cart needs shipping calculated.
Date shipOnDate	Ship on date (optional)
Date deliveryDate	Delivery date (optional)
boolean shipToResidential	True if the address is residential.
String specialInstructions	Special instructions for delivery.
CartItem[] items	All the items in the cart.
number subtotal	Subtotal
number subtotalDiscount	Subtotal discount (because of coupon)
number subtotalWithDiscount	Subtotal after discounts have been applied
number taxExempt	True if the customer is tax exempt
number taxRate	Tax rate
number tax	Tax
number taxableSubtotal	Taxable subtotal
number taxableSubtotalDiscount	Taxable subtotal discount (because of coupons)
number taxableSubtotalWithDiscount	Taxable subtotal after discounts.
number buysafeBondCost	Cost of the buySAFE bond (for buySAFE merchants only)
number shippingHandling	Shipping and handling cost
number shippingHandlingDiscount	Shipping and handling discount (because of coupon)
number shippingHandlingWithDiscount	Shipping and handling cost after discount applied.
number giftCharge	Gift charge
number giftCertificateAmount	Gift certificate amount
number giftWrapCost	Gift wrap cost
String giftMessage	Gift message
number surcharge	Credit card surcharge amount
number total	Total
boolean buysafeBondAvailable	buySAFE bond availability
number buysafeBondFree	True if the buySAFE bond is free to the customer
boolean buysafeBondWanted	True if the customer has opted in to a buySAFE bond
String buysafeBondingSignal	The HTML code for the buySAFE flash control
String buysafeBondingSignalJavascript	The Javascript code for the buySAFE flash control
String buysafeCartDisplayText	The sales text to display by the buySAFE control
String buysafeCartDisplayUrl	The URL to link the sales text to so the customer can learn more about buySAFE.
String ipAddress	IP Address of the customer. Used for geo-location of shipping estimates.
String screenBrandingThemeCode	The screen branding theme associated with the cart.
String advertisingSource	Advertising source the customer selected or entered.
CartCoupon[] coupons	Coupons that have been applied to the cart.
boolean hasPayPal	True if the merchant has PayPal enabled.
boolean paypalCompatible	True if this cart is compatible with PayPal
String paypalButtonUrl	URL of the PayPal express checkout image
String paypalButtonAltText	Alt text to use on the PayPal express checkout image
boolean hasGoogleCheckout	True if the merchant has Google Checkout enabled.
boolean googleCheckoutCompatible	True if this cart is compatible with Google Checkout
String googleCheckoutButtonUrl	URL of the Google Checkout express checkout image
String googleCheckoutButtonAltText	Alt text to use on the Google Checkout image.

String customField1	A custom field to store up to 50 characters.
String customField2	A custom field to store up to 50 characters.
String customField3	A custom field to store up to 50 characters.
String customField4	A custom field to store up to 50 characters.
String customField5	A custom field to store up to 50 characters.
boolean insureShipAvailable	True if bond is available.
boolean insureShipSeparate	True if the bond would be separate from shipping.
number insureShipCost	Cost of the bond
Boolean insureShipWanted	True/false if the bond is wanted. This will be null if the customer has not made a choice yet.
String taxCounty	The tax county assigned to this customer. See method <code>getTaxCounties()</code> .

CartCoupon

This object represents a coupon that has been applied to a shopping cart.

Field	Description
String couponCode	The coupon code used by the customer.

CartItem

This object represents a an item currently in the shopping cart.

Field	Description
integer itemId	A unique item object identifier. This is provided to easy debugging with UltraCart premium support.
String itemId	Item ID
String description	Description
String extendedDescription	Extended description
number quantity	Quantity
number unitCost	Unit cost of item
number unitCostWithDiscount	Unit cost of item after discounts (because of a coupon or mix and match groups)
number arbitraryUnitCost	Arbitrary unit cost (not functional at this time)
Weight weight	Weight of the item
Distance length	Length of the item
Distance width	Width of the item
Distance height	Height of the item
boolean kit	True if this item is a kit
String autoOrderSchedule	Schedule to put this auto orderable item on (only applicable if the item is configured for customer selected auto order)
CartItemOption[] options	Options on the item that customer needs to provide
CartKitComponentOption[] kitComponentOptions	Kit component options that the customer needs to provide
CartItemAttribute[] attributes	Attributes configured on the item
Boolean upsell	True if this item was added to the cart as part of an upsell
number	The MSRP of the item.

manufacturerSuggestedRetailPrice	
CartItemMultimedia[] multimedias	Multimedia objects available for this item.
number minimumQuantity	The minimum quantity of this item the customer must purchase.
number maximumQuantity	The maximum quantity of this item a customer can purchase.

CartItemAttribute

An attribute that has been configured on an item that is in the shopping cart.

Field	Description
String name	Name of the attribute
String value	Value of the attribute
integer position	Position of the attribute in the attribute list
String type	Type of attribute

CartItemMultimedia

This object represents a multimedia file that is configured on an item.

Field	Description
String type	Type of multimedia file. See the ITEM_MULTIMEDIA_TYPE_ constants in the checkoutapi.js file for a list of valid values.
boolean isDefault	True if this is the default multimedia object for it's type. Please note there can be a default image, default video, default PDF file, etc. all on the same item.
integer imageWidth	Width if this is an image.
integer imageHeight	Height if this is an image.
String viewUrl	URL to view the multimedia file.
String description	Description
String code	Code
boolean excludeFromGallery	True if the file should be excluded from gallery displays
CartItemMultimediaThumbnail[] thumbnails	Thumbnails available for this image.

CartItemMultimediaThumbnail

This object represents a thumbnail of a multimedia image.

Field	Description
integer height	Height of the thumbnail in pixels
integer width	Width of the thumbnail in pixels
String httpUrl	URL to view the thumbnail
String httpsUrl	Secure URL to view the thumbnail

CartItemOption

This object represents an option on one of the items in the shopping cart.

Field	Description
integer optionOid	A unique option object identifier. This is provided to easy debugging with UltraCart premium support.
String name	Option name
String label	Option display label
boolean required	True if required
Boolean ignoreIfDefault	True if option will be ignored on the order if the default value is selected
String type	Type of option. See OPTION_TYPE_ constants in the checkoutapi.js for a list of valid values.
boolean oneTimeFee	True if the fee is only applied once (irregardless of quantity)
number costPerLine	Fee per line of text specified in a multiline option
number costPerLetter	Fee per letter specified in a single or multiline option
number costIfSpecified	Fee if specified
String selectedValue	Seleted value on the option
CartItemOptionValue[] values	Values that the customer can select from for radio or drop down style options

CartItemOptionValue

This object represents an option on one of the items in the shopping cart.

Field	Description
String value	Value of the option
number additionalCost	Additional cost the customer will pay if they select this option
Weight additionalWeight	Additioanl weight included in the order if they select this option
boolean defaultValue	True if this option value should be the default
integer displayOrder	Order in quick the options should be displayed. Lower display orders should come first.

CartKitComponentOption

This object represents an option on one of the kit components of a kit item in the shopping cart.

Field	Description
String itemId	Item id of the kit component item that this option belongs to.
integer itemId	Item oid of the kit component item that this option belongs to.
integer optionOid	A unique option object identifier. This is provided to easy debugging with UltraCart premium support.
String name	Option name
String label	Option display label
boolean required	True if required
Boolean ignoreIfDefault	True if option will be ignored on the order if the default value is selected
String type	Type of option. See OPTION_TYPE_ constants in the checkoutapi.js for a list of valid values.
boolean oneTimeFee	True if the fee is only applied once (irregardless of quantity)
number costPerLine	Fee per line of text specified in a multiline option
number costPerLetter	Fee per letter specified in a single or multiline option
number costIfSpecified	Fee if specified
String selectedValue	Selected value on the option
CartItemOptionValue[] values	Values that the customer can select from for radio or drop down style options

CheckoutHandoffResult

This object is returned by all handoff calls.

Field	Description
String redirectToUrl	The URL that the customer's browser should be redirected to for the handoff. This will be null if there are errors.
String[] errors	The errors that occurred while validating the handoff.

CheckoutTerms

This object is returned from the getCheckoutTerms method.

Field	Description
String text	The text of the checkout terms
Boolean html	True if the text is HTML, otherwise it's plain text that should be formatted accordingly.

CustomerProfile

This object represents a customer profile.

Field	Description
integer customerProfileOid	The customer profile object id. This is provided to make debugging easier by UltraCart premium support.
String email	Email address associated with the customer profile
boolean taxExempt	True if the customer is tax exempt
boolean allowPurchaseOrder	True if the customer is allowed to use a purchase order
boolean autoApprovePurchaseOrder	True if the purchase order will automaticall be approved
boolean allowCod	True if the customer is allowed to use a COD
boolean autoApproveCod	True if the COD will automatically be approved
boolean freeShipping	True if the customer receives free shipping
Number freeShippingMinimum	Amount required to qualify for free shipping. Value will be null if there is no threshold specified
String firstName	First name
String lastName	Last Name
String title	Title
String company	Company
String address1	Address line 1
String address2	Address line 2
String city	City
String postalCode	Postal Codew
String country	Country
String dayPhone	Day phone
String eveningPhone	Evening phone
String fax	Fax
String taxId	Tax ID
Number minimumSubtotal	Minimum subtotal the customer is required to purchase. Will be null if no threshold is specified
Integer minimumItemCount	Minimum item count the customer is required to purchase. Will be null if no threshold is specified.
boolean noRealtimeCharge	True if the customer credit card will not be charged real-time during the checkout process.
boolean exemptShippingHandlingCharge	True if the customer is exempt from shipping and handling charges
boolean noFreeShipping	True if the customer is prevented from standard retail customer free shipping
boolean allow3rdPartyBilling	True if the customer is allowed to bill the shipping to their 3 rd party account number.
boolean noCoupons	True if the customer is not allowed to use coupons
String upsAccountNumber	3 rd party account number to bill the UPS shipping to
String fedexAccountNumber	3 rd party account number to bill the FedEx shipping to
String dhlAccountNumber	3 rd party account number to bill the DHL shipping to
String[] pricingTiers	Array of all the pricing tiers this customer has been granted
CustomerProfileAddress[] billingAddresses	Addresses in this customer's billing address book
CustomerProfileAddress[]	Addresses in this customer's shipping address book

shippingAddresses	
String password	New password for the customer profile. Can only be set before a call to updateCustomerProfile.

CustomerProfileAddress

This object represents an address in the customer profile's address book.

Field	Description
String firstName	First name
String lastName	Last name
String company	Company
String address1	Address line 1
String address2	Address line 2
String city	City
String state	State
String postalCode	Postal code
String country	Country
String dayPhone	Day Phone
String fax	Fax
String eveningPhone	Evening phone
String title	Title
String taxCounty	Tax county

Distance

An object that represents distance in a specific unit of measure.

Field	Description
String uom	The unit of measure. See DISTANCE_UOM_ constants in the checkoutapi.js file for a list of valid values.
number value	The value of the distance in the specified unit of measure.

GiftSettings

All the settings about gift giving that are available during the checkout to the customer.

Field	Description
boolean allowGifts	True if the customer is allowed to give a gift
number giftCharge	Charge for sending the order as a gift
integer maxMessageLength	Maximum length of the gift message
GiftWrap[] giftWraps	Wrapping papers that are available to the customer

GiftWrap

Represents a gift wrap that is available for the customer to select

Field	Description
String title	Title of the gift wrap
number cost	Cost to the customer if they select this gift wrap
String url	URL to the image of the gift wrap

Item

This object represents an item. This is a different object than the CartItem which represents the item after it's been added to the cart.

Field	Description
integer itemId	Unique item object identifier. Provided to make debugging easier by UltraCart premium support
String itemId	Item ID
String description	Description
String extendedDescription	Extended description
number cost	Cost
Weight weight	Weight
Distance length	Length
Distance width	Width
Distance height	Height
ItemAttribute[] attributes	Attributes on the item
ItemMultimedia[] multimedias	Multimedia on the item
int availableQuantity	The quantity of this item in stock and available for immediate shipment
boolean inStock	True if this item is instock
boolean allowBackorder	True if this item can be back ordered
boolean preorder	True if this item is on pre-order
Boolean inventoryTracked	True if this item is tracking inventory levels
ItemOption[] options	The options that are available on this item.
number manufacturerSuggestedRetailPrice	The MSRP of the item.
number minimumQuantity	The minimum quantity the customer must purchase.
number maximumQuantity	The maximum quantity the customer can purchase.

ItemAttribute

An object representing an attribute configured on the item.

Field	Description
String name	Name of the attribute
String value	Value of the attribute
String type	Type of the attribute

ItemMultimedia

This object represents a multimedia file that is configured on an item.

Field	Description
String type	Type of multimedia file. See the ITEM_MULTIMEDIA_TYPE_ constants in the checkoutapi.js file for a list of valid values.
boolean isDefault	True if this is the default multimedia object for it's type. Please note there can be a default image, default video, default PDF file, etc. all on the same item.
integer imageWidth	Width if this is an image.
integer imageHeight	Height if this is an image.
String viewUrl	URL to view the multimedia file.
String description	Description
String code	Code
boolean excludeFromGallery	True if the file should be excluded from gallery displays
ItemMultimediaThumbnail[] thumbnails	Thumbnails available for this image.

ItemMultimediaThumbnail

This object represents a thumbnail of a multimedia image.

Field	Description
integer height	Height of the thumbnail in pixels
integer width	Width of the thumbnail in pixels
String httpUrl	URL to view the thumbnail
String httpsUrl	Secure URL to view the thumbnail

ItemOption

This object represents an option on an item.

Field	Description
integer optionOid	A unique option object identifier. This is provided to easy debugging with UltraCart premium support.
String name	Option name
String label	Option display label
boolean required	True if required
Boolean ignoreIfDefault	True if option will be ignored on the order if the default value is selected
String type	Type of option. See OPTION_TYPE_ constants in the checkoutapi.js for a list of valid values.
boolean oneTimeFee	True if the fee is only applied once (irregardless of quantity)
number costPerLine	Fee per line of text specified in a multiline option
number costPerLetter	Fee per letter specified in a single or multiline option
number costIfSpecified	Fee if specified
String selectedValue	Selected value on the option
ItemOptionValue[] values	Values that the customer can select from for radio or drop down style options

ItemOptionValue

This object represents an option on one of the items in the shopping cart.

Field	Description
String value	Value of the option
number additionalCost	Additional cost the customer will pay if they select this option
Weight additionalWeight	Additional weight included in the order if they select this option
boolean defaultValue	True if this option value should be the default
integer displayOrder	Order in which the options should be displayed. Lower display orders should come first.

ShippingEstimate

This object represents the estimate for a shipping method on a cart.

Field	Description
String name	Name of the shipping method. If this method is selected by the customer than this is the method name that should be set on the cart.shippingMethod field.
String displayName	Name of this method that should be displayed to the customer.
String comment	Comment associated with the shipping method.
String estimatedDelivery	Estimated delivery time. Should be displayed beside the method if it's available.
number cost	Cost of the shipping method
boolean discounted	True if this method has been discounted because of a coupon.

Weight

An object that represents weight in a specific unit of measure.

Field	Description
String uom	The unit of measure. See WEIGHT_UOM_ constants in the checkoutapi.js file for a list of valid values.
number value	The value of the weight in the specified unit of measure.

API Methods

addItem

Method Signature

String[] addItem(CartItem[] items);

Description

Adds the specified item(s) to the cart. The minimum amount of information that must be specified on the CartItem object is itemId and quantity. If you want to pass in options then you'll also need to populate those fields. The global cart variable is automatically included in the call to the server and updated after the call is complete.

Parameters

CartItem[] items – The array of items to add to the cart.

Result

String[] –All the errors that occurred while trying to add the items to the cart. If this array is empty then the addition was successful. If this array contains any values then you'll want to display the errors to the customer. This could include things like out of stock conditions, invalid item ids, etc. Even if the errors array contained something the global cart variable is updated.

applyCoupon

Method Signature

String[] applyCoupon(String couponCode);

Description

Applies a coupon to the cart. The global cart variable is automatically submitted with this call.

Parameters

String couponCode – the coupon code to use.

Result

String[] –The errors associated with applying the coupon code. These could range from invalid coupon codes to coupon conflicts. If this array contains any values then you should display them to the customer. Even if the errors array contained something the global cart variable is updated automatically.

applyGiftCertificate

Method Signature

Cart applyGiftCertificate(String giftCertificateCode);

Description

Applies a gift certificate to the cart. A cart can only have one gift certificate on it. You should call the validateGiftCertificate method before calling this method. The global cart variable is automatically submitted with this call.

Parameters

String giftCertificateCode – the gift certificate code to use.

Result

Cart – the updated cart that reflects the applied gift certificate.

backgroundUpdateCart

Method Signature

void backgroundUpdateCart()

Description

Updates all the fields on a cart, except for the item information. The cart does not have to be passed to this method because the method knows to submit the global cart variable. This method is used to update the cart inside of UltraCart without using the return result to modify the global cart object. This method should not be called directly. Instead use triggerBackgroundUpdateCart() on your on* handlers. This helper method will set a 2.5 second timeout and then fire the backgroundUpdate call automatically. If they continue typing, etc. then the timer is automatically reset. This drastically minimizes the amount of traffic back and forth to the server while still effectively capturing the customer's information for abandon marketing.

Parameters

None

Result

None

clearFinalizeAfter

Method Signature

`boolean clearFinalizeAfter ()`

Description

This method will clear the finalize timer that you might have set with the `setFinalizeAfter` method. This would typically only be called if the customer choose some type of navigation where they were going to continue to browse your entire store. This is not a common method to need to call.

Parameters

None

Result

Boolean – returns true if the timer was successfully cleared.

clearItems

Method Signature

`String[] clearItems();`

Description

Removes all the items from the cart. The global cart variable is automatically submitted to the server.

Parameters

None

Result

`String[]` –All the errors that occurred while trying to remove all the items from the cart. This should rarely ever contain anything, but it still should be checked and handled properly by the client. Even if the errors array contained something the global cart variable is updated.

checkoutHandoff

Method Signature

CheckoutHandoffResult checkoutHandoff(String returnUrl, String errorMessageParameterName);

Description

Hands off the customer's cart into the UltraCart checkout. If the merchant has upsells configured then the customer will be shown those upsells. Then the customer will see the inprogress screen and if the payment is successful the receipt. If there are any errors then the customer is going to be redirect back to the page specific as a parameter and the errors will be appended as query string parameters.

Parameters

String returnUrl – the URL to return the browser to if there are any errors.

String errorMessageParameterName – the query string parameter name to put the errors into. If there is more than one error then there will be multiple parameters on the query string with the same name. Be careful to read and display these errors to the customer properly.

Result

CheckoutHandoffResult – Inspect the errors array on the handoff object. If there are any errors you'll need to display them to the customer first and have them correct them before they can continue. If there are no errors then there will be a redirectToUrl populated. You will need to change the browser's window location to this URL to successfully hand off the browser.

checkoutHandoffOnCustomSSL

Method Signature

CheckoutHandoffResult checkoutHandoffOnCustomSSL(String secureHostName, String returnUrl, String errorMessageParameterName);

Description

Hands off the customer's cart into the UltraCart checkout. If the merchant has upsells configured then the customer will be shown those upsells. Then the customer will see the inprogress screen and if the payment is successful the receipt. If there are any errors then the customer is going to be redirect back to the page specific as a parameter and the errors will be appended as query string parameters. The checkoutHandoff method above should be used if the merchant does not have a custom SSL on their account.

Parameters

String secureHostName – the custom SSL host name that you want the customer to finish their order on.

String returnUrl – the URL to return the browser to if there are any errors.

String errorMessageParameterName – the query string parameter name to put the errors into. If there is more than one error then there will be multiple parameters on the query string with the same name. Be careful to read and display these errors to the customer properly.

Result

CheckoutHandoffResult – Inspect the errors array on the handoff object. If there are any errors you'll need to display them to the customer first and have them correct them before they can continue. If there are no errors then there will be a redirectToUrl populated. You will need to change the browser's window location to this URL to successfully hand off the browser.

createCart

Method Signature

Cart createCart();

Description

Creates a new shopping cart and returns the Cart object. The cart.cartId parameter should be stored in a cookie so that subsequent requests to the page can use the cartId in a call to getCart.

Parameters

None

Result

Cart – the newly created cart.

establishCustomerProfile

Method Signature

Cart establishCustomerProfile(String email, String password);

Description

This function will store the email address on the cart and the password for the new customer profile that will be created when the customer finalizes the checkout.

Parameters

String email – the email address of the customer.

String password – the password to be used on the customer profile.

Result

Cart – the updated cart. The global cart variable is automatically updated after this call.

establishCustomerProfileImmediately

Method Signature

Cart establishCustomerProfileImmediately(String email, String password);

Description

This function will create a new customer profile immediately and associate it with the cart. If the customer does not complete their order, the customer profile will still be created.

Parameters

String email – the email address of the customer.

String password – the password to be used on the customer profile.

Result

Cart – the updated cart. The global cart variable is automatically updated after this call.

estimateShipping

Method Signature

ShippingEstimate[] estimateShipping();

Description

Estimates the shipping for a given cart. This can be used to display all the shipping options that are available to the customer. Typically this is done as a set of radio buttons. Some merchants may choose to set the shipping cost at the lowest one that is available and not give the customer the option of selecting shipping.

Parameters

None – the global cart variable is automatically submitted with this call.

Result

ShippingEstimate[] – An array of shipping estimates for the given cart. The estimates will be sorted lowest to highest in cost. If the cart contains items that require shipping and this array is empty then an error condition is present.

getAdvertisingSources

Method Signature

String[] getAdvertisingSources();

Description

Returns a list of configured advertising sources for a given cart. This can be used to display a nice drop down to the customer. Since advertising sources can be configured on a per screen branding level it is important to make sure the screenBrandingThemeCode is set on the cart object to the proper theme. By default a new Cart will have the default screen branding theme so that is the list that will be returned when this function is called.

Parameters

None – the global cart variable is automatically submitted for this call.

Result

String[] – an array of advertising sources. This will be an empty array if you have not configured any advertising sources.

getAllowedCountries

Method Signature

`String[] getAllowedCountries();`

Description

Returns a list of all the countries that this merchant has configured to allow the customer to purchase from. This list should be used to display a drop down list to the customer. It is very important that the country values on your Cart object match something in this list.

Parameters

None

Result

`String[]` – a list of all the countries that can be purchased from in the checkout.

getCart

Method Signature

Cart getCart(String cartId);

Description

Fetches the cart associated with the specific cartId.

Parameters

String cartId – The cartId associated with the cart. This should come from the result of a createCart call that was stored in a cookie and then read back from the client.

Result

Cart – The cart associated with the cartId. This can be null if the cart has expired and been purged. You should be prepared to check for this condition and make a new call to createCart to setup a new cart.

getCartInstance

Method Signature

Cart getCartInstance();

Description

This method retrieves the cart associated with this customer. It handles all the cookie setup, new cart creation, existing cart retrieval, etc. If it is called more than once on a page then it will return the global cart variable that has already been initialized.

Parameters

None

Result

Cart – the customer's shopping cart.

getCartItemMultimediaThumbnail

Method Signature

CartItemMultimediaThumbnail getCartItemMultimediaThumbnail(CartItem cartItem, CartItemMultimedia cartItemMultimedia, int width, int height);

Description

Creates a thumbnail of the specified width and height for the given multimedia object.

Parameters

CartItem – The item that the multimedia object is associated with.

CartItemMultimedia – The multimedia object that you want a thumbnail of.

int width – the width of the thumbnail

int height – the height of the thumbnail

Result

If the system is able to create the thumbnail in real-time it will return a CartItemMultimediaThumbnail object. This method can return null if the system is unable to create the thumbnail in real-time or the thumbnail is not ready from a background creation.

getCheckoutTerms

Method Signature

CheckoutTerms getCheckoutTerms();

Description

Retrieves the checkout terms associated with a cart.

Parameters

None – the global cart variable is automatically submitted with this call.

Result

CheckoutTerms – The checkout terms appropriate for this cart. This will be null if the cart has not already been created.

getCustomerProfile

Method Signature

CustomerProfile getCustomerProfile();

Description

Retrieves the customer profile associated with a cart. This can be performed after a successful login to retrieve information about the customer profile.

Parameters

None – the global cart variable is automatically submitted with this call.

Result

CustomerProfile – the customer profile associated with the cart. This will be null if the cart is not logged in to a specific customer profile.

getGiftSettings

Method Signature

GiftSettings getGiftSettings();

Description

Retrieves all the gift giving settings related to a specific cart. This includes whether gift giving is available, the fees associated with gift giving, the maximum message size, the wrapping papers available, etc.

Parameters

None – the global cart variable is automatically submitted with this call.

Result

GiftSettings – the gift giving settings for the specific cart.

getItem

Method Signature

Item getItem(String itemId);

Description

Retrieves an item with the specified itemId.

Parameters

String –item id to retrieve.

Result

Item– the item object. If you specified an invalid item id in the parameter then the item will be null.

getItemMultimediaThumbnail

Method Signature

ItemMultimediaThumbnail getItemMultimediaThumbnail(Item item, ItemMultimedia itemMultimedia, int width, int height);

Description

Creates a thumbnail of the specified width and height for the given multimedia object.

Parameters

Item – The item that the multimedia object is associated with.

ItemMultimedia – The multimedia object that you want a thumbnail of.

int width – the width of the thumbnail

int height – the height of the thumbnail

Result

If the system is able to create the thumbnail in real-time it will return a ItemMultimediaThumbnail object. This method can return null if the system is unable to create the thumbnail in real-time or the thumbnail is not ready from a background creation.

getItems

Method Signature

Item[] getItems(String[] itemIds);

Description

Retrieves an array of items with the specified itemIds.

Parameters

String[] – an array of item ids to retrieve.

Result

Item[]– the array of item objects. If you specified an invalid item id in the parameter then the item will not be present in the result.

getParameterValues

Method Signature

String getParameterValue(String parameterName);

Description

Reads the value for the parameter specified.

Parameters

String parameterName – The name of the parameter that should be retrieved.

Result

String – the value of the parameter. If the parameter was not specified then null will be returned.

getParameterValues

Method Signature

String[] getParameterValues (String parameterName);

Description

Reads the values for the parameter specified. Since a parameter can be specified multiple times on a query string this function returns an array.

Parameters

String parameterName – The name of the parameter that should be retrieved.

Result

String[] – array of values read from the query string. This array will be empty if there are no occurrences of the parameter.

getRelatedItems

Method Signature

Item[] getRelatedItems();

Description

Returns an array of all the items that are related to the items in the cart. This can be used to display additional items to the customer for purchase.

Parameters

None – the global cart variable is automatically submitted with this call.

Result

Item[] – an array of related items.

getReturnPolicy

Method Signature

String getReturnPolicy();

Description

Returns the configured return policy for a given cart. This should be displayed to the customer somewhere during the checkout process. Since the return policy can be configured on a per screen branding level it is important to make sure the screenBrandingThemeCode is set on the cart object to the proper theme. By default a new Cart will have the default screen branding theme so that is the return policy that will be returned when this function is called.

Parameters

None – the global cart variable is automatically submitted with this call.

Result

String – the return policy for the given cart.

getStateProvinces

Method Signature

String[] getStateProvinces(String country);

Description

Retrieves a list of valid states/provinces for the specified country. This can be used to display this field as a nice drop down list for certain countries. This call uses a dataset contained in the checkoutapi.js file so the call does not go remotely to UltraCart. This makes it extremely fast.

Parameters

String country – The country to retrieve the states for.

Result

String[] – The list of valid states. If this list is empty then UltraCart does not have a built in state list in which case the client should display the state field as a regular text field.

getStateProvinceCodes

Method Signature

String[] getStateProvinceCodes(String country);

Description

Retrieves a list of valid states/provinces for the specified country. This can be used to display this field as a nice drop down list for certain countries. This call uses a dataset contained in the checkoutapi.js file so the call does not go remotely to UltraCart. This makes it extremely fast.

Parameters

String country – The country to retrieve the states for.

Result

String[] – The list of valid state/province codes. If this list is empty then UltraCart does not have a built in state list in which case the client should display the state field as a regular text field.

getTaxCounties

Method Signature

String[] getTaxCounties();

Description

Retrieves the tax counties for a given Cart. A customer will need to select from a tax county if the information contained in their address is not enough to select a tax rate. This method call only needs to be used if the merchant has configured taxes beyond the state level, but has not gone all the way down to the zip code level. Some cities may belong in more than one county so it's necessary to ask the customer which county they're in.

Parameters

None – the global cart variable is automatically submitted with this call.

Result

String[] – an array of tax counties that the customer needs to select from. If this array is empty then there is no need to prompt the customer to select a tax county.

googleCheckoutHandoff

Method Signature

CheckoutHandoffResult googleCheckoutHandoff(String returnUrl, String errorMessageParameterName);

Description

Hands off the customer's cart to Google Checkout.

Parameters

String returnUrl – the URL to return the browser to if there are any errors.

String errorMessageParameterName – the query string parameter name to put the errors into. If there is more than one error then there will be multiple parameters on the query string with the same name. Be careful to read and display these errors to the customer properly.

Result

CheckoutHandoffResult – Inspect the errors array on the handoff object. If there are any errors you'll need to display them to the customer first and have them correct them before they can continue. If there are no errors then there will be a redirectToUrl populated. You will need to change the browser's window location to this URL to successfully hand off the browser to Google Checkout.

googleCheckoutHandoffOnCustomSSL

Method Signature

CheckoutHandoffResult googleCheckoutHandoffOnCustomSSL(String secureHostName, String returnUrl, String errorMessageParameterName);

Description

Hands off the customer's cart to Google Checkout. The googleCheckoutHandoff method above should be used if the merchant does not have a custom SSL on their account.

Parameters

String secureHostName – The custom SSL host name to use during the hand off.

String returnUrl – the URL to return the browser to if there are any errors.

String errorMessageParameterName – the query string parameter name to put the errors into. If there is more than one error then there will be multiple parameters on the query string with the same name. Be careful to read and display these errors to the customer properly.

Result

CheckoutHandoffResult – Inspect the errors array on the handoff object. If there are any errors you'll need to display them to the customer first and have them correct them before they can continue. If there are no errors then there will be a redirectToUrl populated. You will need to

change the browser's window location to this URL to successfully hand off the browser to Google Checkout.

initializeCheckoutApi

Method Signature

```
void initializeCheckoutApi(String merchantId, String secureHostName, String callbackUrl);
```

Description

This method initializes all the settings of the checkout API. This method should be called before any other methods are called.

Parameters

String merchantId – The UltraCart merchant ID to create this cart against.

String secureHostName (optional) – If you're using a custom SSL with UltraCart then this should be your custom SSL domain name.

String callbackUrl (optional) – If you're creating a custom checkout on our own website instead of in an UltraCart dynamic catalog then you'll need to specify the URL of the relay script. This URL must be an HTTPS URL to make sure all the communication with our server is secure.

Result

None

logError

Method Signature

```
void logError(String message);
```

Description

Records an error in the UltraCart checkout API call history. You should use try/catch blocks around your javascript and log the errors so that you can monitor the performance of your custom checkout and know that the customer's are not experiencing any problems. The message will automatically include the browser's name, version, and the platform it's running on. This will make it easier to know if a specific browser is having issues with your javascript code.

Parameters

String message – The error message to record. This can be up to 2,000 characters of data. If you submit more than 2,000 characters the message will be truncated and recorded so there is no need to do client side checks on the length.

Result

None

loginCustomerProfile

Method Signature

CustomerProfile loginCustomerProfile(String email, String password);

Description

Logs in a customer profile for the given cart using the email address and password specified. This will allow the cart to reflected the discounted prices that are available to them as well as utilize their address book. The global cart variable is automatically submitted with this call.

Parameters

String email – email address of the customer profile.

String password – password of the customer profile.

Result

CustomerProfile – If the login is unsuccessful then the result of this call will be null. This does not mean that the cart has been lost. It's just a way of indicating an error condition so be careful to just check the result and not assign it to your cart variable in the page.

logoutCustomerProfile

Method Signature

Cart logoutCustomerProfile();

Description

Logs out a customer profile from a cart.

Parameters

None – the global cart variable is automatically submitted with this call.

Result

Cart – the updated cart reflecting the customer profile logout. The prices will return to retail levels. The global cart variable is automatically updated after this call.

paypalHandoff

Method Signature

CheckoutHandoffResult paypalHandoff(String returnUrl, String errorMessageParameterName);

Description

Hands off the customer's cart to PayPal

Parameters

String returnUrl – the URL to return the browser to if there are any errors.

String errorMessageParameterName – the query string parameter name to put the errors into. If there is more than one error then there will be multiple parameters on the query string with the same name. Be careful to read and display these errors to the customer properly.

Result

CheckoutHandoffResult – Inspect the errors array on the handoff object. If there are any errors you'll need to display them to the customer first and have them correct them before they can continue. If there are no errors then there will be a redirectToUrl populated. You will need to change the browser's window location to this URL to successfully hand off the browser to PayPal.

paypalHandoffOnCustomSSL

Method Signature

CheckoutHandoffResult paypalHandoffOnCustomSSL(String secureHostName, String returnUrl, String errorMessageParameterName);

Description

Hands off the customer's cart to PayPal. The paypalHandoff method above should be used if the merchant does not have a custom SSL on their account.

Parameters

String secureHostName – The custom SSL host name to use during the hand off.

String returnUrl – the URL to return the browser to if there are any errors.

String errorMessageParameterName – the query string parameter name to put the errors into. If there is more than one error then there will be multiple parameters on the query string with the same name. Be careful to read and display these errors to the customer properly.

Result

CheckoutHandoffResult – Inspect the errors array on the handoff object. If there are any errors you'll need to display them to the customer first and have them correct them before they can continue. If there are no errors then there will be a redirectToUrl populated. You will need to

change the browser's window location to this URL to successfully hand off the browser to PayPal.

removeCoupon

Method Signature

`Cart removeCoupon(String couponCode);`

Description

Removes a coupon from the cart. The global cart variable is automatically submitted with this call.

Parameters

String couponCode – the coupon code to remove.

Result

Cart – the updated cart the reflects the removal of the coupon. The global cart variable is automatically updated after this call.

removeGiftCertificate

Method Signature

`String[] removeGiftCertificate();`

Description

Removes the gift certificate from the cart. The global cart variable is automatically submitted with this call.

Parameters

None.

Result

String[] – All the errors that occurred while trying to remove the gift certificate from the cart. If this array is empty then the removal was successful. If this array contains any values then you'll want to display the errors to the customer. Even if the errors array contained something the global cart variable is automatically updated.

removeItem

Method Signature

`String[] removeItem(String itemId);`

Description

Removes the specified item ids from the cart.

Parameters

String itemId – An item Id to remove from the cart.

Result

String[] – All the errors that occurred while trying to remove the item to the cart. If this array is empty then the removal was successful. If this array contains any values then you'll want to display the errors to the customer. Even if the errors array contained something the global cart variable is automatically updated.

removeItems**Method Signature**

```
String[] removeItems(String[] itemIds);
```

Description

Removes the specified item ids from the cart.

Parameters

String[] itemIds – An array of item Ids to remove from the cart.

Result

String[] –All the errors that occurred while trying to remove items to the cart. If this array is empty then the removal was successful. If this array contains any values then you'll want to display the errors to the customer. Even if the errors array contained something the global cart variable is automatically updated.

resetCustomerProfilePassword**Method Signature**

```
String resetCustomerProfilePassword(String email);
```

Description

Resets the password associated with the email address and sends out an email with it immediately.

Parameters

String email – Email address of a customer profile.

Result

A string representing the result. "Success" will be returned if the reset succeeded otherwise the string will contain an error message.

setFinalizeAfter

Method Signature

boolean setFinalizeAfter(int minutes);

Description

Sets the finalize order after timer for the specified minutes. This should be called whenever you load a custom upsell after page. This timer expiring is what turns a shopping cart into an order if they abandon during your custom upsell after sequence.

Parameters

int minutes – the number of minutes without activity before the cart is turned into an order.

Result

Returns true if the timer was successfully set.

subscribeToAutoResponder

Method Signature

String[] subscribeToAutoResponder (String autoResponderName, String[] listIds);

Description

Subscribes the customer to the specified auto responder name. This works like using the parameters on the buy link. See the AUTO_RESPONDER_ constants in the checkoutapi.js for the proper names to pass in. Make sure that you've set at least the customer's email address on the cart object before calling this method. The global cart object is automatically submitted with this method and updated after the fact. Most auto responder's store information about the list's they're subscribing the customers to in the custom fields of the cart.

Parameters

String autoResponderName – name of the auto responder service.

String[] listIds – The name or IDs of the lists to subscribe them to. This works just like the values you would specify on the buy link parameters as documented for your auto responder.

Result

String[] – array of errors that occurred while trying to subscribe the customer.

updateCart

Method Signature

Cart updateCart()

Description

Updates all the fields on a cart, except for the item information. You should call these after updating information like billing, shipping, payment, etc. The cart does not have to be passed to this method because the method knows to submit the global cart variable.

Parameters

None

Result

Cart – the updated cart object. This will have recalculated values as a result of the update. The global cart variable is automatically updated after this call is complete.

updateCustomerProfile

Method Signature

boolean updateCustomerProfile(CustomerProfile customerProfile)

Description

Update the customer profile associated with the cart using the information contained in the customerProfile object. If you want to change the password on a customer profile they must be successfully logged into the profile first, then set the new password on the customer profile object, and then call the updateCustomerProfile method.

Parameters

CustomerProfile customerProfile – All the customer profile information to update including their billing and shipping address books.

Result

Boolean – true if the update is successful.

updateItems

Method Signature

String[] updateItems(CartItem[] items);

Description

Updates all the items in the cart. The minimum amount of information that must be specified on the CartItem object is itemId and quantity. If you want to pass in options then you'll also need to populate those fields. This call should contain all of the items in the cart. It is a complete update, not a partial one.

Parameters

Item[] items – The array of items to update on the cart.

Result

String[] – All the errors that occurred while trying to update the items to the cart. If this array is empty then the addition was successful. If this array contains any values then you'll want to display the errors to the customer. This could include things like out of stock conditions, invalid item ids, etc. Even if the errors array contained something the global cart variable is updated.

validate

Method Signature

String[] validate(String[] checks);

Description

Validates the given cart for the set of validation checks. This is very useful for determining the errors that need to be displayed on a page. The checks that you choose to validate are dependent upon what your page is collecting. If you're building a single page checkout then it's better to use the validateAll method below. The global cart variable is automatically submitted with this call.

Parameters

String[] checks – the checks to perform. See the VALIDATE_ constants located in the checkoutapi.js file for a list of valid check constants.

Result

String[] – the errors present in the current cart based upon the checks performed.

validateAll

Method Signature

String[] validateAll();

Description

Validates the given cart for all validation checks. A full validation of the cart will always occur before a successful checkoutHandoff will occur so it's a good idea to perform the check and display the errors to the user gracefully.

Parameters

None – the global cart variable is automatically submitted with this call.

Result

String[]– the errors present in the current cart.

validateGiftCertificate

Method Signature

String[] validateGiftCertificate(String giftCertificateCode);

Description

Validates a gift certificate code.

Parameters

String giftCertificateCode – the gift certificate code to validate.

Result

String[]– any errors associated with the validation of the gift certificate. If this is array is empty then the gift certificate is valid.

Making an Asynchronous Call

All API calls are synchronous unless you specify an optional last parameter at the end of the function call. Below we'll show you how to make an asynchronous call.

Example of synchronous code

```
// Make the API call
var shippingEstimates = estimateShipping();

// Consume the result
for (var i; i < shippingEstimates.length; i++) {
    // Do something with the estimate
}
```

Example of asynchronous code

```
// Define a callback function
function myCallbackFunction(shippingEstimates) {
    for (var i; i < shippingEstimates.length; i++) {
        // Do something with the estimate
    }
}

// Make the asynchronous call
estimateShipping({'async': true, 'onComplete': myCallbackFunction});
```

Our recommendation is to make the the shipping estimate call asynchronous due to the time it can take calculate estimates. Make sure that you give the customer a visual indication that shipping estimates are being calculated. We do not recommend making calls that manipulate the cart asynchronous as it could cause inconsistencies in the global cart variables contents.

Global Variables

The javascript checkout API makes use of some global variables. These variables are described below.

Variable	Description
String secureHostName	The secure host name that the API should talk back to. This defaults to secure.ultracart.com. This should be changed only via the initialize call.
String merchantId	The merchant ID of the UltraCart account that this cart is for. This should be set via the initialize call.
String callbackUrl	The URL that the checkout api will make server callbacks to.
Cart cart	The customer's cart. This will default to null. This variable is automatically set when you call getCartInstance. All methods that modify this object will automatically update it when the method completes. There is never a need to set this variable manually.

Cookie Issues

The getCartInstance method call will read/write a cookie named UltraCartShoppingCartID on your domain. This cookie will contain the cart ID that is used to fetch the cart when they reload the page. Make sure you do not manipulate this cookie or remove it in any way. It will break the checkout API. It's also important that the customer have cookies turned on for this checkout API to work. In today's world of web 2.0 websites cookies are mandatory to have turned on for most websites to function properly.

Building a Custom Upsell After Sequence

Normally when you execute the checkout handoff call you are handing off the browser into the tail end of the regular UltraCart checkout sequence. If you have any upsell afters configured inside of UltraCart then those will be displayed to the customer, then the in progress screen, and finally the receipt.

If for some reason you need to create custom upsell after pages on your own checkout it is easy to do. When the customer clicks your finalize order button then send them to your own upsell after page instead of doing a checkout handoff. When the upsell page loads, make sure to call the setFinalizeAfter method when the page loads. This will start the timer. If the customer accepts your upsell then add the new item to the cart and recalculate their shipping. You can chain together as many upsell after pages on your own site as you would like, but make sure that each time the page loads you call the setFinalizeAfter timer to reset it. We would recommend 45 minutes as your minute parameter on the method call. When you get to the end of your own custom upsells, then execute the checkoutHandoff call like you normally would.

XHR Issues and the Relay Script

The current browser security model does not allow for a script loaded from one domain to see or talk to javascript loaded from another domain. Make sure that you're referencing all scripts (MooTools, checkoutapi.js, etc.) using HTTPS URLs all from the same domain in your HTML file.

If you're going to create a custom checkout on your own web server then you need to follow of important steps.

1. Make sure your web server has an SSL certificate. Asking customers to enter their credit card on a non-SSL site would dramatically lower conversion and pose a security risk.
2. Download the mootools and checkout API javascript files to your server.
<https://secure.ultracart.com/checkoutapi/checkoutapi.js>
<https://secure.ultracart.com/js/mootools-1.2-core-yc.js>
3. Download the PHP proxy script from the integration center and place it on your web server. The PHP script requires that your server have the Curl module with SSL enabled. This is a very common module to have available in most LAMP hosting environments. An ASP version of the relay script will be available in the future for Microsoft hosting environments. The direct download link for the PHP proxy script is:
<https://secure.ultracart.com/merchant/integrationcenter/proxy.php>
4. Call the initializeCheckoutAPI method with your merchant ID, your custom SSL host name if you have one or null for the second parameter, and then the HTTPS URL to the relay script that you have installed on your server.

Let's pretend that I have a site called avkits and the domain is www.avkits.com. I've already installed an SSL certificate with my hosting company so that I can access the site via https://www.avkits.com or http://www.avkits.com. I've also downloaded the three files mentioned below to the root directory of my hosting account. Below is an example of my initialization code for my HTML page:

```
<script type="text/javascript" src="/mootools-1.2-core-yc.js"></script>
<script type="text/javascript" src="/checkoutapi.js"></script>
<script type="text/javascript">
    initializeCheckoutAPI('AVKIT', null, 'https://www.avkits.com/proxy.php');
</script>
```

Notice that the URL to proxy script is the complete HTTPS url.

Automatic Updating of the Global Cart Variable

To make development easier many of the methods in the checkoutapi.js file automatically submit the global cart variable parameter to the server as well as update it on the return. This dramatically reduces the complexity of coding your checkout as well as the scoping issues of trying to pass a cart object around to other methods. The following methods automatically update the global cart variable:

- addItem
- applyCoupon
- applyGiftCertificate
- clearItems
- establishCustomerProfile
- getCartInstance
- logoutCustomerProfile
- removeCoupon
- removeItems
- removeItem
- updateCart
- updateItems

Call Logging and Javascript Error Handling

During the development stage of your custom checkout it's a good idea to review the checkout API call log located under the integration center menu. The call log will display the last 100 errors that occurred. You can also enable the log to record all calls. This can be very helpful to understand not only the operations your checkout is calling, the order of the operations, the data being submitted as parameters as well as the results your client received. Once you put your custom checkout into production you should make sure to turn off the log all calls setting so that only errors are logged. You can then review your call log periodically to see if any server side or client side errors are occurring.

Implementing buySAFE on a Custom Checkout

Implementing the buySAFE order bonding option on your custom checkout is pretty simple. The two main pieces code that you'll have to implement are the

Including the buySAFE javascript. This should be included after the checkoutapi.js file, but before any of your custom checkout javascript.

```
<script type="text/javascript" src="https://seal.buysafe.com/private/rollover/rollover.js"></script>
```


Define the click handler function. The buySAFE flash control for the bonding will call this method when they toggle their bonding choice.

```
<script type="text/javascript">
var buySAFEOnClick = function() {
    // Toggle whether they want buySAFE
    cart.buysafeBondWanted = !cart.buysafeBondWanted;

    // Update the cart. This will fetch new buySAFE bonding code, totals, etc.
    updateCart();

    // Call your piece of code that updates the summary section of the checkout
    // Note you have to implement some kind of method like displayCartTotals
    displayCartTotals();
}
</script>
```

The final step is displaying the bonding control and the price of the bond. The convention for displaying the buySAFE control is just above the total. Below is an example of what a checkout should look like that implements buySAFE:

Subtotal	\$59.95
Shipping	\$4.80
Tax	\$0.00
 YOUR PURCHASE IS BONDED	\$1.80
Learn more about buySAFE	
Total	\$66.55

In the javascript that you write to display the summary section of the checkout you should first check to see if the buySAFE bond is available by looking at `cart.buysafeBondAvailable` and the displaying the buySAFE line or hiding it. If the bond is available then you need to look at whether the customer is currently bonded by checking `cart.buysafeBondWanted`. If this Boolean flag is true then you should display the bonding signal on the left with the learn more link below and the bond cost to the right. If the flag is false then just display the bonding signal and learn more link. The cost of the bond is available from `cart.buysafeBondCost`.

To display the buySAFE bonding signal your HTML needs to contain an empty span tag with the id of `BuySafeButtonDiv`. Below is an example of the HTML:

```
<span id="BuySafeButtonDiv"></span>
```

To display the bonding signal inside of the span tag you simply evaluate the javascript contained on the cart:

```
<script type="text/javascript">
    eval(cart.buysafeBondingSignalJavascript);
</script>
```

The javascript contained in the buysafeBondingSignalJavascript field will properly call the buySAFE javascript and write out the bonding signal.

The learn more link is also very easy to output to your page. Below is some example code:

```
<script type="text/javascript">
    document.writeln("<a href=\"\" + cart.buysafeCartDisplayUrl + \"\" target=\"_blank\">" + cart. buysafeCartDisplayText + "</a>");
</script>
```

Handling Reloads with Error Messages

When you a checkout handoff the customer's browser is redirected into the UltraCart checkout for completion. If there are any errors that need to be redisplayed to the customer then UltraCart will redirect their browser to the URL that you specified in the handoff call. The error messages will appear in the query string as the parameter specified in the handoff call. Please note that there can be more than one error that needs to be displayed to the customer. For your convience we have provided the method `getParameterValues` which fetches the error messages. Make sure that your javascript code not only displays the errors, but also should reload all the fields on the page so that customer only needs to fix the error fields and not respecify everything.

Performance Considerations

The best customer experience will result from the fewest API calls possible. It is your responsibility to optimize your checkout code to reduce API calls. If you are lazy in your programming your checkout requires additional optimization then you will have two choices. Either implement the required suggestions made by UC Pro Services or utilize the premium support of UC Pro Services to have them implemented for you. Having a poorly written custom checkout that places unnecessary load on the UltraCart system will not be tolerated. You will have your ability to use the javascript checkout API removed if you do not follow the rules.

Using JQuery with the Mootools Framework

The Mootools framework that UltraCart's javascript checkout API can be incompatible with other frameworks. We've tested and built checkouts with the JQuery framework being used as well. In order to use JQuery with Mootools you should include it in your page after Mootools and use the following code to object your JQuery instance.

```
var $j = jQuery.noConflict();
```

Working with JSON Dates

Currently there is no standardized syntax for JSON serialized date objects. The JavaScript Checkout API uses a few date objects within the API (primary the shipOnDate and deliveryDate fields on the Cart object). When are you working with these fields they actually come down from the XHR response as strings. You will need to use the provided helper methods to transform the strings to JavaScript native Date objects and vice versa. The syntax of these two helper methods in the checkoutapi.js is:

```
function ucJsonStringToDate(String s ) => Date;  
function ucDateToJsonString(Date d) => String;
```

Below is an example of how to retrieve the shipOnDate, do something with it, and then update the cart objects:

```
var d = ucJsonStringToDate(cart.shipOnDate);  
// Do some manipulation of the date here.  
cart.shipOnDate = ucDateToJsonString(d);
```

Remember that shipOnDate and deliveryDate can both be null. The JavaScript helper methods will properly interpret nulls.