agreement, or other license agreement which has been executed by you and Oracle and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Sample Code

Oracle may provide sample code in SuiteAnswers, the Help Center, User Guides, or elsewhere through help links. All such sample code is provided "as is" and "as available", for use only with an authorized NetSuite Service account, and is made available as a SuiteCloud Technology subject to the SuiteCloud Terms of Service at www.netsuite.com/tos.

Oracle may modify or remove sample code at any time without notice.

No Excessive Use of the Service

As the Service is a multi-tenant service offering on shared databases, Customer may not use the Service in excess of limits or thresholds that Oracle considers commercially reasonable for the Service. If Oracle reasonably concludes that a Customer’s use is excessive and/or will cause immediate or ongoing performance issues for one or more of Oracle’s other customers, Oracle may slow down or throttle Customer’s excess use until such time that Customer’s use stays within reasonable limits. If Customer’s particular usage pattern requires a higher limit or threshold, then the Customer should procure a subscription to the Service that accommodates a higher limit and/or threshold that more effectively aligns with the Customer’s actual usage pattern.

Beta Features

Oracle may make available to Customer certain features that are labeled “beta” that are not yet generally available. To use such features, Customer acknowledges and agrees that such beta features are subject to the terms and conditions accepted by Customer upon activation of the feature, or in the absence of such terms, subject to the limitations for the feature described in the User Guide and as follows: The beta feature is a prototype or beta version only and is not error or bug free and Customer agrees that it will use the beta feature carefully and will not use it in any way which might result in any loss, corruption or unauthorized access of or to its or any third party’s property or information. Customer must promptly report to Oracle any defects, errors or other problems in beta features to support@netsuite.com or other designated contact for the specific beta feature. Oracle cannot guarantee the continued availability of such beta features and may substantially modify or cease providing such beta features without entitling Customer to any refund, credit, or other compensation. Oracle makes no representations or warranties regarding functionality or use of beta features and Oracle shall have no liability for any lost data, incomplete data, re-run time, inaccurate input, work delay, lost profits or adverse effect on the performance of the Service resulting from the use of beta features. Oracle’s standard service levels, warranties and related commitments regarding the Service shall not apply to beta features and they may not be fully supported by Oracle’s customer support. These limitations and exclusions shall apply until the date that Oracle at its sole option makes a beta feature generally available to its customers and partners as part of the Service without a “beta” label.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item Search API</td>
<td>1</td>
</tr>
<tr>
<td>Items Available to the Item Search API</td>
<td>2</td>
</tr>
<tr>
<td>The Base URL</td>
<td>3</td>
</tr>
<tr>
<td>Item Search API Input Parameters</td>
<td>4</td>
</tr>
<tr>
<td>Item Search API Output Response</td>
<td>12</td>
</tr>
<tr>
<td>Sample Item Search Query and Results</td>
<td>16</td>
</tr>
<tr>
<td>Extensions API</td>
<td>1</td>
</tr>
<tr>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td>Extensibility Component Classes</td>
<td>3</td>
</tr>
<tr>
<td>Commerce API</td>
<td>1</td>
</tr>
<tr>
<td>Shopping Objects</td>
<td>2</td>
</tr>
<tr>
<td>Input Parameters/Return Values for Shopping Object Methods</td>
<td>2</td>
</tr>
<tr>
<td>ShoppingSession Methods</td>
<td>3</td>
</tr>
<tr>
<td>Customer Methods</td>
<td>27</td>
</tr>
<tr>
<td>Order Methods</td>
<td>38</td>
</tr>
<tr>
<td>Understanding Multiple Ship To</td>
<td>69</td>
</tr>
<tr>
<td>Working with Taxes using Commerce API</td>
<td>70</td>
</tr>
<tr>
<td>PageGenerator Methods</td>
<td>71</td>
</tr>
<tr>
<td>StandardTagLibrary Methods</td>
<td>74</td>
</tr>
<tr>
<td>JSON Object Fields</td>
<td>76</td>
</tr>
<tr>
<td>order</td>
<td>76</td>
</tr>
<tr>
<td>address</td>
<td>77</td>
</tr>
<tr>
<td>analyticsettings</td>
<td>78</td>
</tr>
<tr>
<td>checkoutsettings</td>
<td>78</td>
</tr>
<tr>
<td>creditcard</td>
<td>79</td>
</tr>
<tr>
<td>customer</td>
<td>80</td>
</tr>
<tr>
<td>discounts_impact</td>
<td>82</td>
</tr>
<tr>
<td>discounts</td>
<td>82</td>
</tr>
<tr>
<td>fulfillmentPreferences</td>
<td>83</td>
</tr>
<tr>
<td>giftcertificate</td>
<td>83</td>
</tr>
<tr>
<td>item</td>
<td>84</td>
</tr>
<tr>
<td>message</td>
<td>85</td>
</tr>
<tr>
<td>orderitem</td>
<td>85</td>
</tr>
<tr>
<td>ordersettings</td>
<td>87</td>
</tr>
<tr>
<td>ordersummary</td>
<td>87</td>
</tr>
<tr>
<td>payment</td>
<td>89</td>
</tr>
<tr>
<td>paymentmethod</td>
<td>91</td>
</tr>
<tr>
<td>promocode</td>
<td>92</td>
</tr>
<tr>
<td>registrationsettings</td>
<td>94</td>
</tr>
<tr>
<td>shipmethod</td>
<td>94</td>
</tr>
<tr>
<td>sitecategory</td>
<td>94</td>
</tr>
<tr>
<td>sitesettings</td>
<td>95</td>
</tr>
<tr>
<td>status</td>
<td>97</td>
</tr>
<tr>
<td>subscriptions</td>
<td>98</td>
</tr>
<tr>
<td>summarytax</td>
<td>98</td>
</tr>
<tr>
<td>taxdetails</td>
<td>98</td>
</tr>
<tr>
<td>threedssecure</td>
<td>99</td>
</tr>
<tr>
<td>touchpoints</td>
<td>100</td>
</tr>
<tr>
<td>Serversync Touchpoint</td>
<td>100</td>
</tr>
<tr>
<td>SSP Application Governance</td>
<td>102</td>
</tr>
<tr>
<td>SuiteScript API</td>
<td>104</td>
</tr>
</tbody>
</table>
Item Search API

Applies to: SuiteCommerce Web Stores

The Item Search API is a configuration driven API, which is determined by your Search Settings. For example, items returned in the API response data can be filtered by the facets you defined on the Search Index subtab when you set up Search Settings. For more information, see the help topic Search Settings Overview.

The Item Search API uses a REST/JSON style of communication, and supports searching, sorting, faceting, pagination, or any combination of these as they can operate independently. Web Developers can use this API to build facet filtering and keyword search on item data.

Note: To use the Item Search API, you must have a SuiteCommerce website, and you must have configured item search settings for your site.

See the following topics for more information about the Item Search API:

- Items Available to the Item Search API – This topic describes the factors that affect whether an item can be retrieved by the Item Search API.
- The Base URL – The base URL is the URL sent in the HTTP request to the Item Search API in order to retrieve a list of items. This topic describes the structure of the base URL.
- Item Search API Input Parameters – This topic describes all the input parameters that can be used in query strings to retrieve item record attributes.
- Sample Item Search Query and Results – This topic provides sample Item Search API queries and descriptions of response data.
Items Available to the Item Search API

 Applies to: SuiteCommerce Web Stores

The Item Search API returns data from all item records that have been indexed. For more information about the search index, see the help topic Search Index Overview.

The Item Search API returns the items depending on the item availability in a given subsidiary at the time of indexing. Therefore, items are returned only if the following conditions are met:

# If the Subsidiaries feature is enabled for your account, the item should be associated with a subsidiary that is online in the website.
# The item should be associated with an income account. For more information, see the help topic Entering Accounting Information on Items.
# The Display in Web Site box should be selected on the item record.
# The Inactive box should not be selected on the item record.
# The item should be assigned to a Site Builder Category or a Commerce Category, or the Show Uncategorized Items box should be selected on the Web Site Setup page. For more information, see the help topic Item Setup for Search Index.
# The item should be in stock, or the selected Out of Stock Behavior option should display the item even when out of stock.
# The item type should not be Other Charge Item or Discount Item. For more information, see the help topic Item Types.
# The item price should be a numeric value. For example, some item types such as Discounts can have prices in percentage (10% off instead of $10 off).

Note: All item records flagged to display in the web site are accessible to the Item Search API. Since the Item Search API does not provide any authentication, the data retrieved by this service is exposed to public traffic on the internet.

Important: If you are using an earlier release of SuiteCommerce Advanced, you require a patch to ensure that the Item Search API response is cached in the CDN. For more information, see the help topic Item Search API Response Data not Cached.
The Base URL

 Applies to: SuiteCommerce Web Stores

The Item Search API is typically consumed by an AJAX client. You can use a client-side JavaScript framework like jQuery to send an HTTP request to the API and get the JSON response back. The URL is accessible by any client that supports HTTP and JSON. You can retrieve results for a particular item search by sending an HTTP GET request.

The Item Search API base URL has the following format:

http://www.mywebstore.com/api/items

The components of the base URL is described here:

- **Custom Domain** – A unique URL is required, which must be a custom domain. For example, www.mywebstore.com is a custom domain.
- **Items** – This API is designed to query item records in your NetSuite account. Note that /api/items in the URL is part of the path and it is required. You cannot assign a value to it.
- **Parameters** – You can use parameters defined by NetSuite along with values from your account. You can use multiple parameters in an Item Search API request by separating them with an ampersand (&). Parameters are typically name-value pairs, such as fieldset=details. For more information, see Item Search API Input Parameters.

Note: JSONP (JSON with padding) wraps the JSON response from API requests in a JavaScript variable that can be included in a script tag to enable cross-domain AJAX requests. Web developers can use callback as the input parameter as described in Callback Function, and then create the variable name. For more information about the jQuery API, see api.jquery.com/jQuery.getJSON/.
# Item Search API Input Parameters

**Applies to:** SuiteCommerce Web Stores

The Item Search API Input Parameters are used to construct query strings that return item record attributes in a JSON response. To create a query string for the Item Search API, you should use the input parameters with the base URL http://www.mywebstore.com/api/items?{parameters}. You can also use multiple parameters with the base URL by separating them with an ampersand (&).

**Note:** Some parameters when used with the base URL just return the item IDs of the matching items. However, if you want to retrieve additional item details, you can use the parameters listed under the Retrieve Additional Item Details section with these parameters.

The base URL when used without any parameters retrieves the item IDs of all the items. To retrieve only the desired items and item details, you can specify the following input parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>c n region</td>
<td>Retrieve Items by NetSuite Account Number, SuiteCommerce Site ID, Subsidiary ID</td>
</tr>
<tr>
<td>id</td>
<td>Retrieve Items by Item ID</td>
</tr>
<tr>
<td>url</td>
<td>Retrieve Items by Item URL</td>
</tr>
<tr>
<td>q</td>
<td>Retrieve Items by Keyword Search</td>
</tr>
<tr>
<td>commercecategoryid</td>
<td>Retrieve Items by Commerce Categories</td>
</tr>
<tr>
<td>commercecategoryurl</td>
<td></td>
</tr>
<tr>
<td>commercecategoryname</td>
<td></td>
</tr>
<tr>
<td>use_pcv</td>
<td>Retrieve Items from Personalized Catalog for a specific customer or one or more customer groups</td>
</tr>
<tr>
<td>pcv_entity</td>
<td></td>
</tr>
<tr>
<td>pcv_groups</td>
<td></td>
</tr>
<tr>
<td>fields fieldset</td>
<td>Retrieve Additional Item Details</td>
</tr>
<tr>
<td>matrixchilditems_fieldset</td>
<td></td>
</tr>
<tr>
<td>correlateditems_fieldset</td>
<td>Flexibility in Specifying the Field Sets of Related Items</td>
</tr>
<tr>
<td>relateditems_fieldset</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>Pagination of Query Results</td>
</tr>
<tr>
<td>offset</td>
<td></td>
</tr>
<tr>
<td>&lt;field_name&gt;</td>
<td>Facet and Filter Items</td>
</tr>
<tr>
<td>&lt;numeric_facet_field_ID&gt;.ranges</td>
<td></td>
</tr>
<tr>
<td>&lt;numeric_facet_field_ID&gt;.from</td>
<td></td>
</tr>
<tr>
<td>&lt;numeric_facet_field_ID&gt;.to</td>
<td></td>
</tr>
</tbody>
</table>
Item Search API Input Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>include</td>
<td>Include or Exclude Item Attributes</td>
</tr>
<tr>
<td>facet.exclude</td>
<td>Sort Items</td>
</tr>
<tr>
<td>sort</td>
<td></td>
</tr>
<tr>
<td>pricelevel</td>
<td>Specify Price Level</td>
</tr>
<tr>
<td>currency</td>
<td>Specify Currency</td>
</tr>
<tr>
<td>language</td>
<td>Specify Language and Country</td>
</tr>
<tr>
<td>country</td>
<td></td>
</tr>
<tr>
<td>callback</td>
<td>Callback Function</td>
</tr>
<tr>
<td>ssdebug</td>
<td>Retrieve Debug Information</td>
</tr>
</tbody>
</table>

Retrieve Items by NetSuite Account Number, SuiteCommerce Site ID, Subsidiary ID

# c — You can use the parameter c to specify the NetSuite account and retrieve all the items in the specified NetSuite account. For example, if your NetSuite account is 3925062, you can use the following query string to retrieve all the items that belongs to your NetSuite account:

http://www.mywebstore.com/api/items?c=3925062

# n — You can use the parameter n to specify the ID of your SuiteCommerce website and retrieve all the items in the specified SuiteCommerce website. For example, if the ID of your SuiteCommerce website is 3, you can use the following query string to retrieve all the items that belongs to this SuiteCommerce website:

http://www.mywebstore.com/api/items?n=3

# region — You can use the parameter region to specify the Subsidiary in a NetSuite OneWorld account and retrieve all the items in the specified Subsidiary. For example, if your Subsidiary ID is 4, you can use the following query string to retrieve all the items that belongs to this Subsidiary:

http://www.mywebstore.com/api/items?region=4

You can also use these parameters together to further refine the search results. For example, if have more than one SuiteCommerce Website in a NetSuite account and you would like to retrieve items belonging to a particular site, you can use the following query string:

http://www.mywebstore.com/api/items?c=3925062&n=3

Note: If you specify an incorrect NetSuite account number, an incorrect SuiteCommerce Site ID, an incorrect or inactive Subsidiary ID, or if your website is not reachable, no items are returned and an error message is displayed.

Retrieve Items by Item ID

You can use the parameter id to specify the item record ID and retrieve information for a particular item. For example, you can use the following query string to retrieve the item with Item ID 123:

http://www.mywebstore.com/api/items?id=123

To return multiple items in response data, list multiple item IDs separated by commas. For example, you can use the following query string to retrieve the items with Item ID 123, 456, 789:

http://www.mywebstore.com/api/items?id=123,456,789
Retrieve Items by Item URL

You can use the parameter url to specify the value in the URL Component field on the item record. The url parameter returns the Item ID of item that matches the URL component value in the request.

http://www.mywebstore.com/api/items?url=Vok-500-Cellular-Phone

You can specify other parameters along with the url parameter to retrieve other item details. For example, you can use the fields parameter to retrieve associated images:

http://www.mywebstore.com/api/items?url=Vok-500-Cellular-Phone&fields=itemimages_detail

You can also use a field set that contains image information:

http://www.mywebstore.com/api/items?url=Vok-500-Cellular-Phone&fieldset=details

Note: If the item is marked as Inactive or if the Display in Web Store box is not selected on the item record, no items are returned. If you specify an invalid item ID, an error message is displayed. In case of Matrix items, active child items are returned only if the parent item is active and the Display in Web Store box is selected.

Retrieve Items by Keyword Search

You can use the parameter q to retrieve all the items that match the specified keyword. For example, q=jeans searches for that keyword in the fields listed on the Search Fields subtab.

http://www.mywebstore.com/api/items?q=jeans

To do a multi-term search, specify two or more terms in the URL separated by whitespace. For example, q=denim jeans searches for these keywords in the fields listed on the Search Fields subtab.

http://www.mywebstore.com/api/items?q=denim jeans

Note: Relevance is the default sort order when you use the q parameter to query the Item Search API. Your preferred sort order specified on the Sort Fields subtab is overridden. For information on selecting and configuring Sort Fields, see Select and Configure Sort Fields.

Retrieve Items by Commerce Categories

# commercecategoryid — You can use the parameter commercecategoryid to retrieve items that belong the specified Commerce Category ID.

http://www.mywebstore.com/api/items?commercecategoryid=12345

# commercecategoryurl — You can use the parameter commercecategoryurl to retrieve items that belong the specified Commerce Category URL.


# commercecategoryname — You can use the parameter commercecategoryname to retrieve items that belong the specified Commerce Category.

http://www.mywebstore.com/api/items?commercecategoryname=Summer Sale

This parameter is dependent on the Commerce Category facet field being defined in the Web Site Setup record. For more information, see the help topic Add Commerce Categories to Website Search Index.
Retrieve Items from Personalized Catalog for a specific customer or one or more customer groups

Important: To be able to use the pcv_entity and pcv_groups parameters, the user must have a valid NetSuite user account and be assigned a role that provides edit website and list items permissions. For more information, see Personalized Catalog View Roles and Permissions.

# use_pcv — For the Personalized Catalog Views feature to take effect while retrieving items, you must use the use_pcv parameter with the Item Search API. When the use_pcv parameter is set to true, the Item Search API will always serve non-cacheable responses and return tailored selection of catalog items from the user's customer groups.

When the use_pcv parameter is set to false, the Item Search API will serve items for anonymous users regardless of whether the user is logged in or recognized. Also, if the user is logged in or recognized, then the response won't be cacheable.

However, if the Item Search API request is made without the use_pcv parameter, the response is cached even though the user is logged in or recognized.

Note: You must always use the use_pcv parameter with the pcv_entity or the pcv_groups parameter.

# pcv_entity — You can use the pcv_entity parameter to specify a single entity ID to be used as the basis for customer groups to apply query filtering.

For example, let's consider there are items on the site assigned to these four customer groups: ALL_USERS, GUEST_USERS, RECOGNIZED_USERS and VIP_USERS. Items assigned to the VIP_USERS customer group are exclusive items and are not available for customers in other customer groups. There is a customer, say CUSTOMER1, with Customer ID 111 who belongs to the VIP_USERS customer group. A NetSuite user with sufficient permissions for PCV posts the following query:

http://www.mywebstore.com/api/items?use_pcv=T&pcv_entity=111

The Item Search API returns all the items available for customer with customer ID 111 in the response, which includes items from ALL_USERS, RECOGNIZED_USERS and VIP_USERS customer group.

Note: If you specify an incorrect entity ID or more than one entity ID with the pcv_entity parameter, no items are returned in Item Search API response.

# pcv_groups — You can use the pcv_groups parameter to specify one or more customer group IDs to apply query filtering.

For example, let's consider there are items on the site assigned to these three customer groups: ALL_USERS (Customer Group ID= 1), GUEST_USERS (Customer Group ID= 2), RECOGNIZED_USERS (Customer Group ID= 3). A NetSuite user with sufficient permissions for PCV posts the following query:

http://www.mywebstore.com/api/items?use_pcv=T&pcv_groups=3

The Item Search API returns all the items assigned to the group RECOGNIZED_USERS.

Note: If you specify an incorrect customer group ID with the pcv_groups parameter, no items are returned in Item Search API response.

Retrieve Additional Item Details

If you do not specify any input parameters with the Item Search API base URL, only the Item IDs of the items are returned. You can use the following parameters to retrieve additional item details:
Item Search API Input Parameters

# fields — You can use the parameter fields to retrieve additional item record fields in the API response. Your request can include a valid field ID, or multiple field IDs separated by commas to return multiple fields.
http://www.mywebstore.com/api/items?fields=custitem1,upccode

# fieldset — You can use the parameter fieldset to retrieve all the item fields associated with a field set.
http://www.mywebstore.com/api/items?fieldset=details

Note: When the Commerce Category field is defined in the Web Site Setup record field sets, the Commerce Category details for a given item are returned.

Flexibility in Specifying the Field Sets of Related Items

# matrixchilditems_fieldset — You can use the parameter matrixchilditems_fieldset to specify the field set that overrides the reserved matrixchilditems field set. The matrixchilditems field set is used by the matrixchilditems_detail derived field by default to retrieve the item fields of all the matrix child items. In the following example, the default matrixchilditems field set is overridden with a field set named mini using the Item Search API parameter matrixchilditems_fieldset.
http://www.mywebstore.com/api/items?fieldset=search&matrixchilditems_fieldset=mini

# correlateditems_fieldset — You can use the parameter correlateditems_fieldset to specify the field set that overrides the reserved correlateditems field set. The correlateditems field set is used by the correlateditems_detail derived field by default to retrieve the item fields of all the correlated items. In the following example, the default correlateditems field set is overridden with a field set named mini using the Item Search API parameter correlateditems_fieldset.
http://www.mywebstore.com/api/items?fieldset=search&correlateditems_fieldset=mini

# relateditems_fieldset — You can use the parameter relateditems_fieldset to specify the field set that overrides the reserved relateditems field set. The relateditems field set is used by the relateditems_detail derived field by default to retrieve the item fields of all the related items. In the following example, the default relateditems field set is overridden with a field set named mini using the Item Search API parameter relateditems_fieldset.
http://www.mywebstore.com/api/items?fieldset=search&relateditems_fieldset=mini

Pagination of Query Results

You can use the parameter offset in combination with another parameter limit to paginate Item Search API query results. The offset parameter specifies the offset from the first result and the limit parameter specifies the number of items that can be displayed on each page. For example, you can retrieve query results from the eleventh item to the fifteenth item using the following query string:

You can click next at the bottom of the query result to increment the offset parameter by the value of limit. Note that the value of limit remains unchanged. In the above example, when you click next, the value of offset changes to 15 and the next five items are displayed. Also, you can click prev at the bottom of the query result to decrease the value of the offset parameter by the value of limit. In the above example, when you click prev, the value of offset changes to 5 and the previous five items are displayed.

Note: The default value of the offset parameter is 0 and that of the limit parameter is 50. Therefore, when you do not specify offset and limit, the first 50 items are displayed in the query results.

Facet and Filter Items

# <facet_field_name> — You can specify the facet field name and facet values to retrieve the item IDs of the items associated with the specified facets and facet values. For example, you can filter items for the specified facets and facet values:
http://www.mywebstore.com/api/items?color=red,blue&gender=women
If you are unsure about the facet value or if special characters are included in the facet value you want to specify in the query, you can call the Search API requesting all facets. For example, you can request all facets using the following query string:

http://www.mywebstore.com/api/items?pricelevel=5&include=facets

You can then use the appropriate value that is contained in the response to construct your API query.

Note: If you create a custom list intended for use as a facet, do not include a comma in any of the list values. In an API request, the comma is interpreted as a separator between multiple options.

# <numeric_facet_field_ID>.ranges — You can use the parameter <numeric_facet_field_ID>.ranges to define range buckets for numeric facets fields such as price. Use the suffix ranges on the field ID to perform the search. For example, you can retrieve an array of range buckets with the number of items in each bucket using the following query string:

http://www.mywebstore.com/api/items?pricelevel5.ranges=1-10,10-20,20-*.

The range buckets in this example are 1 to 10, 10 to 20, and 20 or more.

The range buckets used to compute range faceting are inclusive of their lower bounds and exclusive of their upper bounds. In the above example, 10 is considered in the range bucket 10 to 20 and not in the range bucket 0 to 10.

Note: Facet ranges do not filter, but instead show additional data in the facet section of the JSON response pertaining to how many items are in each range.

# <numeric_facet_field_ID>.from&<numeric_facet_field_ID>.to — To specify a range of values to search for on item records, you can use the parameters <numeric_facet_field_ID>.from and <numeric_facet_field_ID>.to. Use the suffix from and to on the field ID to perform the search. Both from and to are required, and excluding one results in an error.

For example, you can retrieve all the items with a price between 10 and 20 (both inclusive) using the following query string:

http://www.mywebstore.com/api/items?pricelevel5.from=10&pricelevel5.to=20

Note: You can only use the field ID of the facet field in facet range filters, and not the URL component of the facet.

Include or Exclude Item Attributes

# include — You can use the parameter include to specify an item attribute in the API request and retrieve additional data in the JSON response. For example, the following sample query includes facet fields in the JSON response:

http://www.mywebstore.com/api/items?include=facets

# facet.exclude — You can use the parameter facet.exclude to exclude fields from faceting in Item Search API calls. Excluding facet fields that are not essential can help speed up the API calls. The facet.exclude parameter accepts comma delimited facet field names.

In the following example, the API call specifies a facet value for filtering (custitem_categories=Kitchen-Utensils), while excluding the field from faceting (facet.exclude=custitem_categories).

http://www.mywebstore.com/api/items?
include=facets&fieldset=search&language=en&country=US&currency=USD&pricelevel=5&c=1234567&n=2&custitem_Utensils&sort=pricelevel5&asc&limit=24&offset=0&c=1234567&facet.exclude=custitem_categories

Sort Items

You can use the parameter sort to define the sort order of search results on your website. By using the field ID you want to sort by in the API request, you can retrieve item record data sorted by the field ID included in the API request. You can find the field ID on the Sort Fields subtab on the Web Site Setup page. An Item Search API request overrides your settings on the Search Fields tab. Supported sort orders are ascending (asc) and descending (desc).
For example, you can sort the items by their store display name using the following query string:


Items are sorted by category when the value of commercecategory and either commercecategoryid or commercecategoryurl are specified. This is dependent on the Commerce Category sort field being defined in the Web Site Setup record.

For more information about setting up web site preferences for sorting in NetSuite, see the help topic Select and Configure Sort Fields.

Note: If you are sorting by a field which is empty for a particular item, the item is excluded from sorting and is displayed at the end of the list.

Specify Price Level

You can use the parameter pricelevel to specify the price level ID for items. The price level ID can be different depending on how price levels are configured in your NetSuite account. Typically, the default online price in NetSuite is pricelevel=5.

http://www.mywebstore.com/api/items?pricelevel=4&include=facets

Note: If you have set up multiple price levels for items, ensure that you add all the price level fields as sort fields. For more information, see the help topic Select and Configure Sort Fields. Also, if Alternate Price is not defined for an item but is set to be displayed in a field set, then the value of Alternate Price defaults to the Online Price. If Online Price is also not defined for the item, then the value of Alternate Price defaults to the Base Price.

When a customer with a custom price level is logged in on the web store, the price level assigned to that customer is retrieved instead of the default online price level. For example, the default online price is pricelevel=5, but the price level assigned to a certain customer may be pricelevel=4. When this customer is browsing in the web store, the customer’s assigned price, pricelevel=4 is sent in the search API call. Consequently, the custom price (pricelevel=4) is displayed to this customer for all the products in the web store.

Note: The pricelevel parameter is only usable when a user with permissions to see the requested pricelevel is logged in. In all other cases, the pricelevel displayed is in accordance with the default setting in the Web Site Set Up record.

Specify Currency

Prices are usually displayed in the default currency that you have set up for your website. In an account that uses the Multiple Currencies feature, you can use the parameter currency with the ISO Country Code to specify the currency format in which you want to display prices for the items.

http://www.mywebstore.com/api/items?fields=onlinecustomerprice_formatted,pricelevel5&currency=USD

If you are using the Multiple Currencies feature, ensure that you select the correct default locale on the currency record to avoid any errors related to currency mismatch. For example, select New Zealand (English) as the default locale for New Zealand Dollar. For more information, see the help topic Creating Currency Records.

Note: If you specify a currency that is not associated with your website or if the Online box is not checked for that currency on the Web Site Setup record, an error message is returned.

Specify Language and Country

A shopper on your website can use their local language to search and browse products on your website. The translated text that shoppers see in product views is based on the translated text you set up on item records. The JSON response
data includes item attributes and field set fields in the languages you have set up in your NetSuite account. You must add the corresponding fields to a field set and also check that the field set is included in the Reference ShopFlow SuiteApp. By default, items are returned with fields containing the default language. You can use the language and country parameter to specify the translated text you set up in NetSuite.

In the following example, the search results are displayed in Spanish:

http://www.mywebstore.com/api/items?fieldset=search&country=ES&language=es

Note: You must always use the language and country parameters together. The country code that you specify should be in uppercase and the language code should be in lowercase.

All ISO country and language codes are valid if the language is set up for use on your website. To affect the language displayed in a product view, the site template must be localized. For details on localizing site templates and using multiple languages, see the help topic Localization.

Callback Function

You can use the parameter callback to define the JavaScript callback function in order to wrap the JSON response.

For example, you can create a variable name mycallback by using the callback parameter as shown in the following sample query:

http://www.mywebstore.com/api/items?callback=mycallback

Retrieve Debug Information

You can use the parameter ssdebug and set its value to true to retrieve the debug information in the JSON response.

http://www.mywebstore.com/api/items?ssdebug=true
Item Search API Output Response

The response data includes the following properties:

# total – This is the number of items that match the search query, but without taking into account the stock level variance since the last search index update. Consequently, the actual number of items displayed on the page might be less than the total displayed.

For example, if your search query matches only nine items out of the 1000 items in your catalogue, the value of total displayed is nine. If three out of these nine matching items go out of stock since the last search index update, only six items are displayed even though the value of total displayed is nine.

Note: If you use the url parameter to request an item by specifying the url component value, total is not displayed in the Item Search API output response.

# items – This is an array of items that includes the fields specified using the fields or fieldset parameter. Note that the following output for the items property shows fields that are included in the search field set, which was specified in the sample query:

```
items: [
    {
      ispurchasable: false,
      custitem_ns_py_attributes_rating: "",
      showoutofstockmessage: true,
      custitem38: false,
      custitem33: "&nbsp;",
      itemid: "OL5299",
      onspecial: false,
      onlinecustomerprice: 19.99,
      pricelevel5: 19.99,
      outofstockbehavior: "Disallow back orders but display out-of-stock message",
      storedescription2: "Features<br>Cool.Qtm ZERO provides an immediate and ongoing cooling sensation<br>Mesh sides for ventilation<br>Lightweight foam brim folds up and packs down small for storing in a pack<br>Drawcord attachment under the chin",
      storedisplayname2: "Chiller Wide Brim Hat",
      custitem_onsale: false,
      internalid: 6548
    },
    ...
]
```

# facets – This shows the facets associated with items returned in the response data. Use the include parameter to show all of facets you defined in on the Web Site Setup page. Note the following sample from the output response shows IDs for two facets, Ideal For and Collection. Each facet has two facet values.

```
facets: [
    {
      id: "Ideal For",
      values: [
        {
          url: "Men",
          label: "Men"
        },
        {
          url: "Women",
          label: "Women"
        }
      ]
    }
]
```
For more information on defining facets in NetSuite, see the help topic Select and Configure Facet Fields. For more information about customizing faceted navigation in SuiteCommerce Advanced, see the help topic Faceted Navigation.

**# links** – These links support pagination and are automatically returned in response data. You can also build the URL for the next set of results using the limit and offset parameters returned by the API. You can click next href in the query results to increment the offset parameter by the value of limit and view the next set of results. You can also click prev href in the query results to decrease the value of the offset parameter by the value of limit. Note that the value of limit remains unchanged.

```json
links: [
  {
    rel: "next",
    href: "http://www.mywebstore.com/api/items?pricelevel=5&offset=10&limit=5&fieldset=search&include=facets"
  },
].
```

**# corrections** – This property displays the past and present values for facet URL components in the Item Search API request. If you have not entered any aliases in NetSuite for facets or facet values you used in the past, then the corrections property does not return any data.

For example, in the past you may have used custitem_category as a facet. At that time, the facet value was Hockey Player. Today, you are using Equipment Category as the facet, and the current facet value is Player. When you modify the facet URL values, you can add the original values to the Alias list in NetSuite. If you query the Item Search API using the original facet names and values, then corrections are returned in the response data. For more information about entering URL component aliases, see the help topic URL Components for Facets.

Since the following API request includes the original facet and facet value, the output response to the following query string includes corrections:

```
http://www.mywebstore.com/api/items?pricelevel=5&fieldset=search&include=facets&custitem_category=HockeyPlayer
```

Note that both past and current facet URL components are included in the response data:

```json
corrections: [
  {
    type: "facet",
    id: "custitem_category",
  }
].
```
# locale – This is an object that represents the locale with respect to internationalization. You can troubleshoot errors related to currency mismatch by verifying the details in the locale section. When you are using the Multiple Currencies feature, you should always select the correct default locale on the currency record. For example, select New Zealand (English) as the default locale for New Zealand Dollar.

```json
locale: {
    country: "US",
    language: "en",
    currency: "USD",
    region: 17
}
```

For more information, see the help topic Localization.

# volatility – This refers to the CDN cache settings for your SuiteCommerce Advanced implementation. CDN caching can be set for a shorter period of time or a longer period of time. The length of time that a certain type of content is cached on CDN depends on the value of TTL (Time To Live) in CDN, which could be one of the following:

- **Unique** — This asset is not cached.
- **Short** — This asset may change frequently, so cache it for five minutes.
- **Medium** — This asset may or may not change frequently, so cache it for two hours.
- **Long** — This asset is not expected to change frequently, so cache it for seven days.

By default, the Item Search API response data has a volatility of **Short**. CDN caching can be set for a shorter period of time or a longer period of time as described in Configure CDN Caching. However, changing the CDN cache setting in your application can have a significant performance impact.

**Note:** If you exclude the pricelevel parameter from the Item Search API URL, the volatility is automatically set to **Unique**.

In pre-Elbrus releases of SuiteCommerce Advanced, the Item Search API response data is not cached in the CDN by default. To enable caching of the Item Search API response, you must customize your implementation to include the pricelevel input parameter as described in Item Search API Response Data not Cached.

# code – This is the HTTP status code. All responses should have a status code or a message that allows the client to continue processing the results or handle the error if necessary. If the request is successful, a 200 status code indicates that the request was received, understood, and has been processed.

# warnings – If you have specified a parameter in the query that the API does not recognize, the parameter is ignored and a warning message is displayed in the API response. In the following Item Search API response, a warning message is displayed indicating that the non-existent parameter foo has been ignored.

```json
"warnings": {
    "processeduri": "http://www.mywebstore.com/api/items?fieldset=search&foo=&include=facets"
}
```
"ignorodparams": [ "foo" ]}
Sample Item Search Query and Results

Applies to: SuiteCommerce Web Stores

This section provides sample Item Search API queries and their response data.

- Sample Search Query for Returning Item Quantity
- Sample Search Query for Returning Categories
- Sample Search Query for Returning Items Available for Store Pickup

Sample Search Query for Returning Item Quantity

If you use the Multi-Location Inventory feature, you can access item inventory per location using fields specifically designed to return item location details. You can use these fields to retrieve the Item Quantity details:

- Quantity Available (quantityavailable_detail)
- Quantity Backordered (quantitybackordered_detail)
- Quantity Committed (quantitycommitted_detail)
- Quantity On Hand (quantityonhand_detail)
- Quantity On Order (quantityonorder_detail)

By using these fields with the Item Search API, you can retrieve the item inventory data. You can also create a field set that includes these fields, and then use that field set in a query to the Item Search API. For information on creating field sets, see Define Field Sets.

The following example shows a query string that returns data from the Quantity Backordered field on the item record:

http://www.mywebstore.com/api/items?pricelevel=5&fieldset=myfieldsetwithqtybackordered

The item quantity information is written to the JSON response, which includes internal ID of the location and quantity backordered for each item:

```
items: [

{
  qtybackordered_detail:
 .qtybackordered:10
  locations:
  [
   internalid:1,
   qtybackordered:6
  ],
  internalid:2,
  qtybackordered:4
}
]
```

Sample Search Query for Returning Categories

The following query returns items where the comcat fieldset is configured with a commerce category ID 3. Facet information is also returned in the response. For each item returned, category details are returned including:
# The URL fragments defined for that category.
# The commerce category name used in the facet.

Note: To successfully run this query, the Commerce Category facet field and field set must be configured in the Web Site Set Up record. For more details on working with Categories, see the help topic Commerce Categories.

http://www.mywebstore_suite.com/api/items?fieldset=comcat&include=facets&commercecategoryid=3

```json
{
    "total": 4,
    "items": [
        {
            "internalid": 388,
            "commercecategory": {
                "primarypath": [
                    {
                        "urls": [
                            "/apparel/partywear/shoes"
                        ],
                        "name": "Shoes",
                        "id": 3
                    }
                ],
                "categories": [
                    {
                        "urls": [
                            "/apparel/partywear/shoes"
                        ],
                        "name": "Shoes",
                        "id": 3
                    }
                ]
            },
            "storedisplayname2": "Shoes - Reebok 9K"
        }
    ],
    "facets": [
        {
            "id": "commercecategoryname",
            
```
Sample Search Query for Returning Items Available for Store Pickup

When you use the isstorepickupallowed field with the Item Search API and set it to true, the Item Search API returns the item IDs of all the matching items for which store pickup is enabled. When this field is set to false, the Item Search API returns the item IDs of all the matching items for which store pickup is not enabled.

If you enable the Multi-Location Inventory feature and host a SuiteCommerce Advanced web site, you can access the item quantity per location. If you use the quantityavailableforstorepickup_detail field with the Item Search API, you can also access the item quantity available for store pickup per location.

The following example shows a query string that returns data from the Quantity Available and Quantity Available For Store Pickup fields on the item record. Alternatively, you can create a field set that includes Quantity Available and Quantity Available For Store Pickup fields, and then use that field set in a query to the Item Search API.

http://www.mywebstore.com/api/items?
isstorepickupallowed=true&fields=quantityavailable_detail,quantityavailableforstorepickup_detail

Note: To successfully run this query, the Store Pickup Allowed facet field must be configured in the Web Site Setup record. You must also define the Available For Store Pickup (Detail) field in the Web Site Setup record field sets. For more information, see Accessing Item Quantity Available for Store Pickup Per Location.

For each item returned in the JSON response, the data includes the internalid of the item, total item quantity available, item quantity available by location, and item quantity available for store pickup by location:

```
"items": [
  {
    "internalid":388,
    "quantityavailable_detail":{
      "quantityavailable":90.0,
      "locations":{
        "internalid":2,
        "quantityavailable":39.0
      }
    }
  }
]```
Sample Item Search Query and Results

{
  "internalid":3,
  "quantityavailable":51.0
},
{
  "internalid":2,
  "qtyavailableforstorepickup":39.0
},
{
  "internalid":3,
  "qtyavailableforstorepickup":51.0
},
"quantityavailableforstorepickup_detail":{
  "locations":[
    {
      "internalid":2,
      "quantityavailable":150.0
    },
    {
      "internalid":3,
      "quantityavailable":150.0
    }
  ]
},
"quantityavailableforstorepickup_detail":{
  "locations":{
    "internalid":387,
    "quantityavailable_detail":{
      "quantityavailable":150.0,
      "locations":{
        "internalid":2,
        "quantityavailable":150.0
      }
    }
  }
},
"quantityavailableforstorepickup_detail":{
  "locations":{
    "internalid":107,
    "quantityavailable_detail":{
      "quantityavailable":4.0,
      "locations":{
        "internalid":2,
        "quantityavailable":3.0
      },
      {
        "internalid":3,
        "quantityavailable":1.0
      }
    }
  }
}
In the above JSON response, we can observe the following:

- The first item with internal ID 388 is available at two locations, and both locations allow store pickup for the quantity available at the respective locations.
- The second item with internal ID 387 is only available at one location, and that location allows store pickup for the quantity available.
- The third item with internal ID 107 is available at two locations, but only one location allows store pickup for the quantity available at that location.

Accessing Item Quantity Available for Store Pickup Per Location

To access the quantity available for Store Pickup per location using the Item Search API, you need to create a field set that includes fields listed under Item Fields Related to Store Pickup.

To create a field set that includes information about item quantity available for store pickup:

1. Go to Setup > SuiteCommerce Advanced > Set Up Web Site.
2. Click the Field Sets subtab.
   1. Enter a Name for the field set. For example, enter Store Pickup.
   2. Enter a Field Set ID for the field set. For example, enter storepickup_details.
   3. Select Item for Record Type.
   4. (Optional) Enter a description for the field set.
   5. In the Field Set popup window, select the store pickup related field names listed in the subsequent table.
   6. Click Add.
3. Click Save.
4. Use the field set in a query to the Item Search API. For example,
   http://www.mywebstore.com/api/items?fieldset=storepickup_details

The store pickup related information is returned in the JSON response.

Item Fields Related to Store Pickup

The following table shows the item record fields that correspond with fields exposed to field sets:

<table>
<thead>
<tr>
<th>Field Label on the Locations Subtab of the Item Record</th>
<th>Field Set Field Name</th>
<th>Field ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow Store Pickup</td>
<td>Store Pickup Allowed</td>
<td>isstorepickupallowed</td>
</tr>
<tr>
<td>Quantity Available For Store Pickup</td>
<td>Available For Store Pickup</td>
<td>quantityavailableforstorepickup_detail</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Detail)</td>
</tr>
</tbody>
</table>
Overview

Applies to: SuiteCommerce Web Stores | Aconcagua

SuiteCommerce includes a conceptual layer on top of existing SuiteCommerce architecture. Conceptually, this layer provides an environment for extension developers to customize a Commerce web store and SuiteCommerce InStore (SCIS) through extensions, without the need to extend or customize the underlying application code.

Developers create extensions in their local environment using the extension developer tools. This environment/workspace is represented by the Developer Code Layer in the image above. Each extension contains any number of JavaScript, SuiteScript, or configuration files to perform some purpose. Within this code, developers include API calls to components in the Extensibility API. This is represented by the Extensibility API Layer.

The API then accesses frontend and backend modules (routers, models, views, etc.) in the Application Layer. In this way, extensions are separated programmatically from the application and interact with the API only, not with the application directly. This makes customization more seamless and intuitive for extension developers and ensures that future updates to the application do not compromise developer customizations.

See the following topics for more information about the Extensibility API:

# Extensibility Component Classes – This topic provides an overview of the Component Classes available to you when creating extensions.

# Extensibility API Tutorials – This topic explains how to use the components plus their methods and events when building your extensions. This includes everything from accessing components to building CCTs, Views, etc.

# SCIS Extensibility API – This topic includes information about the components available to you when creating extensions for SCIS.
Go to the Extensibility API Reference for complete documentation on the available component classes.
Extensibility Component Classes

A component is a piece of functionality in a SuiteCommerce application that connects with specific aspects of the application to accomplish a task. The Extensibility API is organized around these components. Each component includes an associated class and various exposed methods for obtaining or modifying data, views, or template context.

Different components are available for SuiteCommerce (and SuiteCommerce Advanced) and SuiteCommerce InStore. The Extensibility API includes the following components:

### SuiteCommerce and SuiteCommerce Advanced Components
- CartComponent
- CheckoutComponent
- CMSComponent
- EnvironmentComponent
- LayoutComponent
- LoginRegisterPage
- MyAccountMenu
- PageType
- ProductDetailsPageComponent
- ProductListPageComponent
- UserProfile

### SuiteCommerce InStore (SCIS) Components
- CartComponent
- SCISLayoutComponent

For more information on how to use these components when building extensions, see the help topic Extensibility API Tutorials.

### SuiteCommerce and SuiteCommerce Advanced Components

The SuiteCommerce components apply to both SuiteCommerce and SuiteCommerce Advanced.

See the full extensibility API reference for detailed information about each component.

### CartComponent

This component interacts entirely with the LiveOrder, allowing you to add, update, and remove lines from the order, apply promotions to the order, select a shipping method to the order, associate the shipping and billing address to the order, add payment methods, etc. This component can access objects available in all three ssp applications: shopping, myaccount and checkout.
This component is available in the Frontend and the Backend.

<table>
<thead>
<tr>
<th>ID</th>
<th>Cart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td></td>
</tr>
<tr>
<td>addLine()</td>
<td></td>
</tr>
<tr>
<td>addLines()</td>
<td></td>
</tr>
<tr>
<td>addPayment()</td>
<td></td>
</tr>
<tr>
<td>addPromotion()</td>
<td></td>
</tr>
<tr>
<td>clearEstimateShipping()</td>
<td></td>
</tr>
<tr>
<td>estimateShipping()</td>
<td></td>
</tr>
<tr>
<td>getAddresses()</td>
<td></td>
</tr>
<tr>
<td>getBillAddress()</td>
<td></td>
</tr>
<tr>
<td>getLines()</td>
<td></td>
</tr>
<tr>
<td>getPromotions()</td>
<td></td>
</tr>
<tr>
<td>getShipAddress()</td>
<td></td>
</tr>
<tr>
<td>getShipMethod()</td>
<td></td>
</tr>
<tr>
<td>getShipMethods()</td>
<td></td>
</tr>
<tr>
<td>getSummary()</td>
<td></td>
</tr>
<tr>
<td>removeLine()</td>
<td></td>
</tr>
<tr>
<td>removePromotion()</td>
<td></td>
</tr>
<tr>
<td>removeShipping()</td>
<td></td>
</tr>
<tr>
<td>submit()</td>
<td></td>
</tr>
<tr>
<td>updateLine()</td>
<td></td>
</tr>
</tbody>
</table>

| Events |       |
| beforeAddLine |       |
| afterAddLine |       |
| beforeUpdateLine |       |
| afterUpdateLine |       |
| beforeRemoveLine |       |
| afterRemoveLine |       |
| beforeEstimateShipping |       |
| afterEstimateShipping |       |
| beforeClearEstimateShipping |       |
| afterClearEstimateShipping |       |
| beforeAddPromotion |       |
| afterAddPromotion |       |
| beforeRemovePromotion |       |
| afterRemovePromotion |       |
| beforeAddPayment |       |
| afterAddPayment |       |
CheckoutComponent

This component interacts entirely with the Checkout page and the checkout steps flow. This component lets you modify the steps visually, such as adding, removing, and reordering steps, etc. This component can access objects available in the checkout application only.

<table>
<thead>
<tr>
<th>ID</th>
<th>Checkout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getCurrentStep()</td>
</tr>
<tr>
<td></td>
<td>setCurrentStep()</td>
</tr>
<tr>
<td></td>
<td>getStepGroupsInfo()</td>
</tr>
<tr>
<td></td>
<td>getStepsInfo()</td>
</tr>
<tr>
<td></td>
<td>getCheckoutFlow()</td>
</tr>
<tr>
<td></td>
<td>addModuleToStep()</td>
</tr>
<tr>
<td></td>
<td>removeModuleFromStep()</td>
</tr>
<tr>
<td></td>
<td>addStepsGroup()</td>
</tr>
<tr>
<td></td>
<td>removeStepsGroup()</td>
</tr>
<tr>
<td></td>
<td>addStep()</td>
</tr>
<tr>
<td></td>
<td>removeStep()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>beforeAddModuleToStep</td>
</tr>
<tr>
<td></td>
<td>afterAddModuleToStep</td>
</tr>
<tr>
<td></td>
<td>beforeSetCurrentStep</td>
</tr>
<tr>
<td></td>
<td>afterSetCurrentStep</td>
</tr>
<tr>
<td></td>
<td>beforeAddStep</td>
</tr>
<tr>
<td></td>
<td>afterAddStep</td>
</tr>
<tr>
<td></td>
<td>beforeAddStepsGroup</td>
</tr>
<tr>
<td></td>
<td>afterAddStepsGroup</td>
</tr>
</tbody>
</table>

CMSComponent

This component is created only for the purpose of registering new Custom Content Types. If you want to add views inside of your CCT you can interact with all the rest of the components available. This component can access objects available in all three ssp applications: shopping, myaccount, and checkout.

<table>
<thead>
<tr>
<th>ID</th>
<th>CMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>registerCustomContentType()</td>
</tr>
<tr>
<td>Events</td>
<td>–</td>
</tr>
</tbody>
</table>
EnvironmentComponent

This component is general to the entire SuiteCommerce Application and provides the utils method and access to Configuration values. This component does not provide interaction with any particular view. This component can access any object available in all three ssp applications: shopping, myaccount and checkout.

<table>
<thead>
<tr>
<th>ID</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>getConfig()</td>
</tr>
<tr>
<td></td>
<td>isPageGenerator()</td>
</tr>
<tr>
<td></td>
<td>getSiteSetting()</td>
</tr>
<tr>
<td></td>
<td>getSession()</td>
</tr>
<tr>
<td></td>
<td>setTranslation()</td>
</tr>
<tr>
<td>Events</td>
<td>–</td>
</tr>
</tbody>
</table>

LayoutComponent

This component gives you an option for modifying an aspect of SuiteCommerce, but do not have a specific component available. For example, no components currently exist that let you add or modify views on the Home Page, the Header, Footer, or MyAccount. Using Layout, you can access all the view methods for each view (addChildViews, addChildView, removeChildView, etc.). As more components become available in future SuiteCommerce updates, you can simply replace the use of Layout for the appropriate (new) component, keeping the same methods.

<table>
<thead>
<tr>
<th>ID</th>
<th>Layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Since</td>
<td>SuiteCommerce 2018.2</td>
</tr>
<tr>
<td>Methods</td>
<td>addChildView()</td>
</tr>
<tr>
<td></td>
<td>addChildViews()</td>
</tr>
<tr>
<td></td>
<td>addToViewContextDefinition()</td>
</tr>
<tr>
<td></td>
<td>addToViewEventsDefinition()</td>
</tr>
<tr>
<td></td>
<td>registerView()</td>
</tr>
<tr>
<td></td>
<td>removeChildView()</td>
</tr>
<tr>
<td></td>
<td>removeToViewContextDefinition()</td>
</tr>
<tr>
<td></td>
<td>removeToViewEventsDefinition()</td>
</tr>
<tr>
<td></td>
<td>setChildIndex()</td>
</tr>
<tr>
<td></td>
<td>showContent()</td>
</tr>
<tr>
<td>Events</td>
<td>–</td>
</tr>
</tbody>
</table>

LoginRegisterPage

The LoginRegisterPage component is used to listen for login and register events on the Log In / Register page in the web store, and enables you to capture additional form data from the login and registration forms.
MyAccountMenu

This component lets you add menu items to the menu on the My Account page. You can add group items and then add subitems to the group items. All group items and subitems have one or more permissions, which lets you restrict which users can view the menu items. Subitems point to a landing page URL that is loaded in the main content area of the page.

<table>
<thead>
<tr>
<th>ID</th>
<th>MyAccountMenu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Since</td>
<td>SuiteCommerce 2019.1</td>
</tr>
<tr>
<td>Methods</td>
<td>addGroup()</td>
</tr>
<tr>
<td></td>
<td>addGroupEntry()</td>
</tr>
<tr>
<td>Events</td>
<td>–</td>
</tr>
</tbody>
</table>

PageType

Use the PageType component when you want to register new page types in an extension. A page type record can be created by logging in to NetSuite and creating the record, or by including the page type in the extension manifest file.

<table>
<thead>
<tr>
<th>ID</th>
<th>PageType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Since</td>
<td>SuiteCommerce 2019.1</td>
</tr>
<tr>
<td>Methods</td>
<td>registerPageType()</td>
</tr>
<tr>
<td></td>
<td>registerTemplate()</td>
</tr>
<tr>
<td>Events</td>
<td>–</td>
</tr>
</tbody>
</table>

ProductDetailsPageComponent

This component interacts entirely with the Product Details page. This accesses objects inside the Product Details View only. This is undefined if invoked from any other part of the application.

<table>
<thead>
<tr>
<th>ID</th>
<th>PDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>getAllMatrixChilds</td>
</tr>
<tr>
<td></td>
<td>getItemInfo</td>
</tr>
</tbody>
</table>
**ProductListPageComponent**

This component interacts entirely with the Search Page, allows navigating to the next or previous page, set the current page, change the rendering, search for a product, get the last search term, and so on. This component accesses objects inside the Facets Browse View in the shopping application only. This is undefined if invoked from any other part of the application.

<table>
<thead>
<tr>
<th>ID</th>
<th>PLP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Methods</strong></td>
<td></td>
</tr>
<tr>
<td>getPagination()</td>
<td></td>
</tr>
<tr>
<td>setCurrentPage()</td>
<td></td>
</tr>
<tr>
<td>getSorting()</td>
<td></td>
</tr>
<tr>
<td>getAllSorting()</td>
<td></td>
</tr>
<tr>
<td>setSorting()</td>
<td></td>
</tr>
<tr>
<td>getDisplay()</td>
<td></td>
</tr>
<tr>
<td>getAllDisplay()</td>
<td></td>
</tr>
<tr>
<td>setDisplay()</td>
<td></td>
</tr>
<tr>
<td>getPageSize()</td>
<td></td>
</tr>
<tr>
<td>getAllPageSize()</td>
<td></td>
</tr>
<tr>
<td>setPageSize()</td>
<td></td>
</tr>
<tr>
<td>getFilters()</td>
<td></td>
</tr>
<tr>
<td>getAllFilters()</td>
<td></td>
</tr>
<tr>
<td>setFilters()</td>
<td></td>
</tr>
<tr>
<td>getSearchText()</td>
<td></td>
</tr>
<tr>
<td>setSearchText()</td>
<td></td>
</tr>
<tr>
<td>getItemsInfo()</td>
<td></td>
</tr>
<tr>
<td>getCategoryInfo()</td>
<td></td>
</tr>
<tr>
<td>getURL()</td>
<td></td>
</tr>
<tr>
<td><strong>Events</strong></td>
<td></td>
</tr>
<tr>
<td>beforeOptionSelection</td>
<td></td>
</tr>
<tr>
<td>afterOptionSelection</td>
<td></td>
</tr>
<tr>
<td>beforeQuantityChange</td>
<td></td>
</tr>
<tr>
<td>afterQuantityChange</td>
<td></td>
</tr>
<tr>
<td>beforeImageChange</td>
<td></td>
</tr>
<tr>
<td>afterImageChange</td>
<td></td>
</tr>
</tbody>
</table>
UserProfile

The UserProfile component lets you retrieve information about the current logged in user.

<table>
<thead>
<tr>
<th>ID</th>
<th>UserProfile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Since</td>
<td>SuiteCommerce 2019.1</td>
</tr>
<tr>
<td>Methods</td>
<td>getUserProfile()</td>
</tr>
<tr>
<td>Events</td>
<td>–</td>
</tr>
</tbody>
</table>

SuiteCommerce InStore (SCIS) Components

The SCIS (SuiteCommerce InStore) components in the extensibility API let you create extensions that add, remove, or update lines in the cart. You can also show modal dialogs to present information to the user, and require the user to confirm or cancel an action.

See the full extensibility API reference for detailed information about each component.

Cart

The CartComponent in SCIS lets you add, remove, and update lines in the cart. It also provides methods to perform other operations in the cart, such as applying promotions and discounts.

<table>
<thead>
<tr>
<th>ID</th>
<th>Cart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Since</td>
<td>SuiteCommerce 2018.2</td>
</tr>
<tr>
<td>Methods</td>
<td>addGlobalDiscount()</td>
</tr>
<tr>
<td></td>
<td>addLine()</td>
</tr>
<tr>
<td></td>
<td>addLines()</td>
</tr>
<tr>
<td></td>
<td>addLineDiscount()</td>
</tr>
<tr>
<td></td>
<td>addPromotion()</td>
</tr>
<tr>
<td></td>
<td>getLines()</td>
</tr>
<tr>
<td></td>
<td>getSummary()</td>
</tr>
<tr>
<td></td>
<td>removeLineDiscount()</td>
</tr>
<tr>
<td></td>
<td>removePromotion()</td>
</tr>
<tr>
<td></td>
<td>returnLine()</td>
</tr>
<tr>
<td></td>
<td>setCustomer()</td>
</tr>
<tr>
<td></td>
<td>undoLine()</td>
</tr>
<tr>
<td></td>
<td>voidLine()</td>
</tr>
<tr>
<td>Events</td>
<td>beforeAddGlobalDiscount</td>
</tr>
<tr>
<td></td>
<td>afterAddGlobalDiscount</td>
</tr>
<tr>
<td></td>
<td>beforeAddLine</td>
</tr>
</tbody>
</table>
SCISLayout

The SCISLayoutComponent gives you access to two methods, both of which show a modal dialog to the user, but with different options to dismiss the dialog.

<table>
<thead>
<tr>
<th>ID</th>
<th>SCISLayout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Since</td>
<td>SuiteCommerce 2018.2</td>
</tr>
<tr>
<td>Methods</td>
<td>showConfirmationPopup()</td>
</tr>
<tr>
<td></td>
<td>showMessagePopup()</td>
</tr>
<tr>
<td>Events</td>
<td>–</td>
</tr>
</tbody>
</table>
Commerce API

Applies to: SuiteCommerce Web Stores

The Commerce API exposes a set of shopping objects and functions, that you can use to:

- Develop web store customizations
- Create SSP application scripts, along with the full range of existing server-side SuiteScript. For details about the SuiteScript API, see the help topic SuiteScript Functions.

See the following topics for more information about the Commerce API:

- **Shopping Objects** – The Commerce API provides several shopping objects whose associated methods enable you to control all aspects of web store behavior. These topics describe all the shopping objects and their associated methods.
- **JSON Object Fields** – These topics describe the structure of all JSON objects that are available for use with the Commerce API shopping methods.
- **SSP Application Governance** – The governance model for SSP applications limits the number of usage units that can be executed by an SSP application script. This topic lists the number of usage units consumed by Commerce API shopping methods so that you can use them to create SSP applications that meet the requirements of the governance model.
Shopping Objects

 Applies to: SuiteCommerce Web Stores

The Commerce API includes only one object that is globally accessible in the SuiteScript context, the webContainer object.

The following table lists the objects that are accessible after you have gotten a webContainer. The table also lists the calls you can use to get a reference to each object. You need a reference to a shopping object to call its methods.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
<th>Call to get Object</th>
<th>For available methods, see:</th>
</tr>
</thead>
<tbody>
<tr>
<td>shoppingSession</td>
<td>Tracks session context.</td>
<td>nlapiGetWebContainer().getShoppingSession()</td>
<td>ShoppingSession Methods</td>
</tr>
<tr>
<td>customer</td>
<td>Holds data for the logged in customer.</td>
<td>nlapiGetWebContainer().getCustomer()</td>
<td>Customer Methods</td>
</tr>
<tr>
<td>order</td>
<td>Holds shopping cart data and methods for placing the order.</td>
<td>nlapiGetWebContainer().getOrder()</td>
<td>Order Methods</td>
</tr>
<tr>
<td>pageGenerator</td>
<td>Used to generate web pages.</td>
<td>nlapiGetWebContainer().getPageGenerator()</td>
<td>PageGenerator Methods</td>
</tr>
<tr>
<td>standardTagLibrary</td>
<td>Implements web site tags.</td>
<td>nlapiGetWebContainer().getStandardTagLibrary()</td>
<td>StandardTagLibrary Methods</td>
</tr>
</tbody>
</table>

Note the following:

# These customer and order objects are different from NetSuite customer and order records, because they hold information within the given shopping session only.
# To access a customer or an order, in addition to getting a webContainer, you also need to get a shoppingSession.
# You can only use Commerce API shopping methods against Commerce API objects. You cannot use shopping methods against NetSuite records.
# The Commerce API includes JSON objects to support shopping object methods. For details about supported fields for JSON objects, see JSON Object Fields.
# For information about governance for shopping object methods, see SSP Application Governance.

Input Parameters/Return Values for Shopping Object Methods

Set Functions

Either a primitive string or a JSON object can be passed in to set methods, depending on their implementation. A primitive string would be used primarily to hold an object key, while a JSON object would provide a flat representation of an object's name/value pairs.

Get Functions

An optional array can be passed in as a filter to get methods. If field names are passed in to a get method, only the name/value pairs for the passed field names are returned. If no field names are passed, all name/value pairs for the expected object are returned. If no value is available for a field, null is returned for that field.
ShoppingSession Methods

A shoppingSession object tracks context.

The following methods are available:

# constructDomainBridgingUrl(link)
# doChangePassword(Params, newPassword)
# getAbsoluteUrl(domaintype, path) (Deprecated)
# getAbsoluteUrl2(domaintype, path)
# getCampaignID()
# getCorrelatedItems(item)
# getCountries()
# getCountryTaxPreferences()
# getCustomer()
# getEffectiveShoppingDomain()
# getInformationItemFieldValues(infoItems)
# getItemFieldValues(items)
# getPartnerID()
# getPasswordHint(customerEmail) (Deprecated)
# getPaymentMethod(paymentMethodId, fields)
# getPaymentMethods(fields)
# getRedirectURL()
# getRelatedItems(item)
# getRelatedCorrelatedItems()
# getShipToCountries()
# getShopperCurrency()
# getShopperLanguageLocale()
# getShopperPreferences()
# getShopperSubsidiary()
# getSiteCategoryContents(siteCategory, recursive) (Deprecated)
# getSiteCategoryFieldValues(siteCategories)
# getSiteCategoryTree(siteCategory)
# getSiteSettings(fields)
# getSSPApplication()
# getStates(countries)
# isChangePasswordRequest()
# isLoggedin()
# isRecognized()
# logIn(customer)
# logout()
# proceedToCheckout(checkoutSettings)
constructDomainBridgingUrl(link)

Converts cross-domain link to domain bridging URL containing encrypted parameters. This method provides a secure way to transfer shopper data between different domains.

**Parameters**

- **link** [required] {String} – Absolute or relative URL of link targeting a different shopping or checkout domain from the one the shopper is currently on

**Returns**

- **link** {String} – Domain bridging URL; the original URL is returned if bridging is either not possible or not necessary
- **null** – Returned if the input parameter is invalid

**Supported Domains**

Checkout, Shopping

**Login Required?**

No

Back to ShoppingSession Methods | Back to Shopping Objects

doChangePassword(Params, newPassword)

Changes the password for the customer identified by email. Returns true if the password is changed correctly.

**Throws the following errors:**

- **ERR_WS_INVALID_LINK** – if parameters contain a token that is expired or incomplete. Note that it is not possible to determine if the link is expired or incomplete. The same error is thrown in either case.
- **ERR_WS_INVALID_PASSWORD** – if the password is empty or exceeds the maximum allowed length of 255 characters.
# ERR_WS_WEAK_PASSWORD – if the password at least 6 characters.

Parameters

- **Params** – pass parameters from original request
- **newPassword** – {String} value for new password to be set

Returns

- Boolean

Supported Domains

- Checkout

Login Required?

- No

**getAbsoluteUrl(domaintype,path)**

**Important:** In certain scenarios, the `getAbsoluteUrl(domaintype,path)` method returns a relative URL instead of an absolute URL. For example, the `getAbsoluteUrl(domaintype,path)` method returns a relative URL when the target domain is same as the current domain.

Therefore, the `getAbsoluteUrl(domaintype,path)` method has been deprecated as of NetSuite Release 2017.1. Use the `getAbsoluteUrl2(domaintype,path)` method instead.

Gets absolute URL for given path and domain type.

Parameters

- **domaintype** [optional] {String} - value should be either “shopping” or “checkout”
- **path** [required] {String}

Returns

- String

Supported Domains

- Checkout, Shopping

Login Required?

- No

**getAbsoluteUrl2(domaintype,path)**

Gets absolute URL for given path and domain type.
ShoppingSession Methods

Note: This method should be used instead of the deprecated getAbsoluteUrl(domaintype,path) method.

Parameters

# domaintype [optional] {String} - value should be either “shopping” or “checkout”
# path [required] {String}

Returns
String

Supported Domains
Checkout, Shopping

Login Required?
No

Back to ShoppingSession Methods | Back to Shopping Objects

cgetCampaignID()

Gets the campaign ID of the leadsource parameter from the current session.

Parameters
No parameters to set.

Returns
campaignid {String}

Supported Domains
Checkout, Shopping

Login Required?
No

Back to ShoppingSession Methods | Back to Shopping Objects

cgetCorrelatedItems(item)

Gets array of correlated items for given item.

Note: Complete details are returned for a correlated item only if the item is:
+ available in the store from which the API was called
+ available in personalized catalog view for the customer (if Personalized Catalog Views is enabled - for more information, see the help topic Personalized Catalog Views)

Parameters

# item [required]{object with values for fields}
InternalID [required]

ItemType [required]

ItemFields [optional] (array of item field names to query)

Note: For better performance limit the number of queried fields by specifying only fields you need in the itemfields array.

Returns

Array of objects of type item

Supported Domains

Checkout, Shopping

Login Required?

No

Back to ShoppingSession Methods | Back to Shopping Objects

countries()

Gets all countries

Parameters

No parameters to set.

Returns

Array of objects with fields:

- name {string}
- code {string}
- isZipRequired {boolean}

Supported Domains

Checkout, Shopping

Login Required?

No

Back to ShoppingSession Methods | Back to Shopping Objects

customer()

Gets the Customer details from the current session.

Parameters

No parameters to set.
Returns

Returns the customer shopping object. For more information, see Customer Methods.

Note: The customer object holds information within the given shopping session only and is different from the NetSuite customer record.

Supported Domains
Checkout, Shopping

Login Required?
Yes

Back to ShoppingSession Methods | Back to Shopping Objects

getCountryTaxPreferences()

Returns a JSON object with tax preferences related a shopper’s country. The values returned are based on the default shopper subsidiary derived from the site or the country selected by the shopper.

Parameters
No parameters to set.

Returns

vatregistration {boolean} — When set to T, the VAT registration field can be displayed during checkout and the vatregistration can be set on the customer object.

Supported Domains
Checkout, Shopping

Login Required?
No

Back to ShoppingSession Methods | Back to Shopping Objects

getEffectiveShoppingDomain()

Returns name of shopping domain associated with current session.

Parameters
No parameters to set.

Returns
domainname {String} – The shopping domain related to the current session. For example: shopping.corp.netsuite.com.

Supported Domains
Checkout, Shopping
getInformationItemFieldValues(infoItems)

Gets attribute values for given information items.

Parameters

- infoItems [required] {Array}
  Each object in array has values for field:
  - internalid [required]

Returns

Array of objects of type item.

Note: Depending on item fields configured, the array of objects returned might differ.

Supported Domains

Checkout, Shopping

Login Required?

No

getItemFieldValues(items)

Gets attribute values for given items.

Parameters

- items [required] {Array} Each object in array has values for fields:
  - internalid [required]
  - itemtype [required]

Returns

Array of objects of type item.

Note: Complete details are returned for an item only if the item is:
  - available in the store from which the API was called
  - available in personalized catalog view for the customer (if Personalized Catalog Views is enabled - for more information, see the help topic Personalized Catalog Views)
getPartnerID()

Gets the Partner ID of the leadsource parameter from the current session.

Parameters
No parameters to set.

Returns

partnerid {String}

getPasswordHint(customerEmail)

Retrieves password hint for customer with given email address.

Important: Password hints are no longer necessary. The password retrieval email message includes a URL with a password reset code. Therefore, the getPasswordHint() method has been deprecated as of NetSuite Release 2017.2. As of 2017.2, the method will return an empty string, but generate no errors.

Parameters

# customerEmail [required] {String}

Returns

An empty string
getPaymentMethod(paymentMethodId, fields)

Gets payment method for given ID.

Parameters

- `paymentMethodId` [required] {String}
- `fields` [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Object of type paymentmethod.

Supported Domains

Checkout, Shopping

Login Required?

No

getPaymentMethods(fields)

 Gets payment methods accepted by the web store.

Parameters

- `fields` [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Array of objects of type paymentmethod.

Supported Domains

Checkout, Shopping

Login Required?

No

getRedirectURL()

Gets the URL which is the result of a redirect given the input. Returns null if there is no redirect. Wildcards are also supported (*).
getRelatedItems(item)

Gets array of related items for given item.

Note: Complete details are returned for a related item only if the item is:

- available in the store from which the API was called
- available in personalized catalog view for the customer (if Personalized Catalog Views is enabled - for more information, see the help topic Personalized Catalog Views)

Parameters

- item [required] {object with values for fields}
  - internalid [required]
  - itemtype [required]
  - itemfields [optional] {array of item field names to query}

Note: For better performance limit the number of queried fields by specifying only fields you need in the itemfields array.

Returns

Array of objects of type item.

Supported Domains

Checkout, Shopping

Login Required?

No

Back to ShoppingSession Methods | Back to Shopping Objects
getRelatedCorrelatedItems()

Gets a list of related and/or correlated items for a given item ID.

Note: Complete details are returned for a related item only if the item is:

- available in the store from which the API was called
- available in personalized catalog view for the customer (if Personalized Catalog Views is enabled - for more information, see the help topic Personalized Catalog Views)

Parameters

- type: [required] [string] Available options are RELATED, CORRELATED, or BOTH.
- fieldsetid : set of fields returned.
- items : list of item IDs. When not specified, items from the cart are returned.
- limit : Maximum count of returned items. When not specified, the default 10 is used. -1 sets no limit.

Important: Use caution when setting the limit to -1 as this can have an impact on performance.

Example

```javascript
function service(request, response)
{
    var session = nlapiGetWebContainer().getShoppingSession();
    items = ['101'];
    response.writeLine("Related Items: " + JSON.stringify(session.getRelatedCorrelatedItems('RELATED', 'relateditems', items, 10)));
    response.writeLine("Correlated Items: " + JSON.stringify(session.getRelatedCorrelatedItems('CORRELATED', 'relateditems', items, 10)));
    response.writeLine("Related & Correlated Items: " + JSON.stringify(session.getRelatedCorrelatedItems('BOTH', 'relateditems', items, 10)));
}
```

Returns

Array of correlated or related items.

Supported Domains

Checkout, Shopping

Login Required?

No

Back to Order Methods | Back to Shopping Objects

getShipToCountries()

Gets countries to which web store can ship.

Parameters

No parameters to set.
Returns

Array of objects with fields:

```plaintext
# name {string}
# code {string}
# isZipRequired {boolean}
```

Supported Domains
Checkout, Shopping

Login Required?
No

Back to ShoppingSession Methods | Back to Shopping Objects

getShopperCurrency()

Gets currency for current shopping session.

Parameters
No parameters to set.

Returns

Object with values for fields:

```plaintext
# internalId {String}
# name {String}
# symbol {String}
# precision {Number}
```

Supported Domains
Checkout, Shopping

Login Required?
No

Back to ShoppingSession Methods | Back to Shopping Objects

getShopperLanguageLocale()

Gets language and locale for current shopping session.

Parameters
No parameters to set.
Returns
localeid {String}

Supported Domains
Checkout, Shopping

Login Required?
No

getShopperPreferences()

Gets language, subsidiary, and currency preferences for the current shopping session.

Parameters
No parameters to set.

Returns
Object with values for fields:
#  language {String}
#  subsidiary {String}
#  currency {String}

Supported Domains
Checkout, Shopping

Login Required?
No

getShopperSubsidiary()

Gets subsidiary for current shopping session.

Parameters
No parameters to set.

Returns
subsidiaryid {String}

Supported Domains
Checkout, Shopping
ShoppingSession Methods

Login Required?
No

Back to ShoppingSession Methods | Back to Shopping Objects

getSiteCategoryContents(siteCategory, recursive)

Important: The `getSiteCategoryContents()` API has been deprecated as of NetSuite Release 2014.2. Use the `getSiteCategoryTree()` method instead. See `getSiteCategoryTree(siteCategory)`. 

Gets contents of given site category.

If not provided, `siteCategory` will be defaulted to the root category, and `recursive` will be defaulted to false. When `recursive` is set to false, the API returns the category contents that include subcategories and items; when `recursive` is set to true, the API returns the complete tree of subcategories but not items.

Note: To get the complete category tree from the root, use `getSiteCategoryContents(true)`.

Parameters

- `siteCategory` [optional] {Array} Each object in array has values for field: `internalid` [required]
- `recursive` [optional]

Returns

Array of objects of type `sitecategory` or `item`.

Supported Domains

Checkout, Shopping

Login Required?
No

Back to ShoppingSession Methods | Back to Shopping Objects

getSiteCategoryFieldValues(siteCategories)

Gets attribute values of given site categories.

Parameters

- `siteCategories` [required] {Array} Each object in array has values for field:
  - `internalid` [required]

Returns

Array of objects of type `sitecategory`.

Supported Domains

Checkout, Shopping
Login Required?

No

Back to ShoppingSession Methods | Back to Shopping Objects

getSiteCategoryTree(siteCategory)

When the internalid attribute is present in the siteCategory object, then the method returns the category sub-tree from this category. When the internalid attribute is not specified, then the whole category tree of the specified site is returned.

When calling this API, you must specify which fields to return. When no fields are specified, then no fields are returned except the internalids of categories. Specify fields using the itemfields attribute on the siteCategory object.

Examples

# The following example returns the entire category tree for the current site. Each category record contains the fields internalid, pagetitle2, and urlcomponent.

```
getSiteCategoryTree({ itemfields : ["pagetitle2","urlcomponent"] });
```

# The following example returns the category sub-tree of category 65. Each category record contains only the internalid field.

```
getSiteCategoryTree({ id : 65 });
```

Note: The examples above return only the category tree, not the category contents.

Parameters

# siteCategory [optional] [Array] Each object in array has values for field: internalid [required]

Returns

Array of objects of type sitecategory

Supported Domains

Checkout, Shopping

Login Required?

No

Back to ShoppingSession Methods | Back to Shopping Objects

getSiteSettings(fields)

Gets settings of current web store.

Parameters

# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned
Returns
Object with fields listed for sitesettings JSON object.

Supported Domains
Checkout, Shopping

Login Required?
No

getSSPApplication()
Gets URL root of SSP application powering the current touchpoint.

Parameters
No parameters to set.

Returns
urlroot field value for sitesettings JSON object.

Supported Domains
Checkout, Shopping

Login Required?
No

getStates(countries)
Gets states and/or provinces.

Parameters
# countries [optional] Array of strings for country codes

Returns
Returns an array of state/province names and state/province codes in the following format:
{ "states":[{ "name":"Alabama","code":"AL"},{ "name":"Alaska","code":"AK"},...], "countrycode":"US" }

Supported Domains
Checkout, Shopping

Login Required?
No
isChangePasswordRequest()

Returns true if all page parameters from the current request are valid for the password reset.

Throws the error “ERR_WS_INVALID_LINK” if all page parameters from the current request contain required parameters for password reset but the link has expired or is incomplete.

Parameters

No parameters to set.

Returns

Boolean

Supported Domains

Checkout

Login Required?

no

isLoggedIn()

Important: This method is no longer supported or recommended. For new implementations, use the isLoggedIn2() method. The isLoggedIn() method works correctly in the Checkout domain but may return false when used in the Shopping domain even when the user is logged in. If you encounter session issues when using this API, contact support to migrate to using the isLoggedIn2() method.

Checks whether customer has logged in. This method can be used with the isRecognized() method. See Using isRecognized() with isLoggedIn2().

Parameters

No parameters to set.

Returns

Boolean

Supported Domains

Checkout

Login Required?

No
isLoggedin2()

Checks whether customer has logged in. This method can be used with the isRecognized() method. See Using isRecognized() with isLoggedIn2().

Note: This method should be used instead of the deprecated isLoggedIn() method.

Parameters

No parameters to set.

Returns

Boolean

Supported Domains

Checkout, Shopping

Login Required?

No

isRecognized()

Determines whether a user is recognized. This method can be used with the isLoggedIn2() method. See Using isRecognized() with isLoggedIn2().

Parameters

No parameters to set.

Returns

Boolean

Supported Domains

Checkout, Shopping

Login Required?

No

logIn(customer)

Logs in a customer.

Parameters

# customer [required][object with values for fields]
ShoppingSession Methods

email [required]
password [required]

Returns

Object with fields:

customerid {String} - internal id of customer record for logged in customer
redirecturl {String} - URL to which user is redirected after logging in

Supported Domains

Checkout

Login Required?
No

logout()

Logs out a customer.

Note: If logout() is attempted from a shopping context, it does not succeed and an error is returned.

Parameters

No parameters to set.

Returns

Object with field:

redirecturl

The field redirecturl has the complete home URL to redirect to after logout. If not using the redirecturl field returned from logout(), the custom redirecturl should have the logoff=T parameter.

Supported Domains

Checkout

Login Required?
No

proceedToCheckout(checkoutSettings)

Used for integration with a third party checkout provider such as PayPal Express. This API can only be called from a secure scheme (https).
Parameters

# checkoutSettings Object with values for fields:
  # type; either “paypalexpress” or “google”
  # continueurl
  # cancelurl

Returns

No value returned.

Supported Domains

Checkout

Note: Must have at least one order.

Login Required?

No

registerCustomer(customer)

Registers the customer and logs them in.

Parameters

# customer [required]{object with values for fields}
  # firstname [required] {String}
  # lastname [required] {String}
  # companyname [optional] {String}
  # email [required] {String}
  # password [required] {String}
  # password2 [required] {String}
  # passwordhint [optional] {String}
  # emailsubscribe [optional] {String}
  # leadsource [optional] {String}

Returns

Object with values for fields:

# customerid {String} - internal id of record for registered customer
# redirecturl {String} - URL to which user is redirected after registering

Supported Domains

Checkout
registerGuest(guest)

Registers a guest.

Note: Throws exception if register guest with password.

Parameters

- guest [optional] {object with values for fields}
  - firstname [optional] {String}
  - lastname [required] {String}

Returns

Object with values for fields:

- customerid {String} - internal id of record for registered customer
- redirecturl {String} - URL to which user is redirected after registering

sendPasswordRetrievalEmail(customeremail)

Sends password retrieval email message to the given email address.

Important: The password reset link generated by the sendPasswordRetrievalEmail() includes the customer’s email address. Therefore the method has been deprecated as of NetSuite Release 2017.2. Use the sendPasswordRetrievalEmail2(customeremail) method instead.

Parameters

- customerEmail [required] {String}

Returns

No value returned.

Supported Domains

Checkout
sendPasswordRetrievalEmail2(customeremail)

Sends password retrieval email message to the given email address.

Note: This method does not include the customer's email address in the reset password link. Use this method instead of the deprecated sendPasswordRetrievalEmail(customeremail).

Parameters

#  customerEmail [required] {String}

Returns

No value returned.

setShopperCurrency(currencyid)

Sets currency for current shopping session

Note: Multiple Currencies feature must be enabled, specified currency must be supported in sitesettings, and logged in user must have permission to change currency.

Parameters

#  currencyid [required] {string}

Returns

No value returned.
setShopperLanguageLocale(localeid)
Sets language and locale for current shopping session

Note: Multi-Language feature must be enabled, and specified language must be supported in sitesettings.

Parameters

# localeid [required] {String}

Returns
No value returned.

Supported Domains
Checkout, Shopping

Login Required?
No

Back to ShoppingSession Methods | Back to Shopping Objects

setShopperPreferences(prefs)
Sets language, subsidiary and currency for current shopping session.

Parameters

# prefs [required] {object with values for fields}
  # language [optional] {String}
  # subsidiary [optional] {String}
  # currency [optional] {String}

Returns
No value returned.

Supported Domains
Checkout, Shopping

Login Required?
No

Back to ShoppingSession Methods | Back to Shopping Objects

setShopperSubsidiary(subsidiary)
Sets subsidiary for current shopping session (for NetSuite OneWorld users).
Important: The subsidiary set in a session is **not** currently supported out-of-the-box for One World in the SuiteCommerce Advanced SuiteApp. For the SuiteCommerce Advanced SuiteApp, further customization is necessary. Also, the subsidiary for an existing recognized customer can **not** be changed using this API.

### Parameters

**subsidiary** [required] {String}

### Returns

No value returned.

### Supported Domains

Checkout, Shopping

### Login Required?

No

---

**verifyEmailChange(key)**

Takes an email change key, verifies it, and then changes the email. If the request can't be processed, an exception is thrown.

### Parameters

**key** [required] {String} — value of the email change key.

### Returns

Boolean

Returns true if the key value is valid and the email has been changed. Returns false in all other cases (for example, if the key is invalid or expired).

### Supported Domains

Checkout

### Login Required?

Yes

---

**Using isRecognized() with isLoggedIn2()**

There are four possible cases for using the isLoggedIn2() method and isRecognized() method to correctly determine the logged in status of a user:

1. **new (anonymous) user:** isRecognized() == false and isLoggedIn2() == false
2. **registered and logged in user:** isRecognized() == true and isLoggedIn2() == true
Shopping Session Methods

# user who clicked logout: isRecognized() == false and isLoggedIn2() == false
# user who had logged in, then closed his browser and later returns to the shop: isRecognized() == true and isLoggedIn2() == false

Note: This last scenario is the only case where the isRecognized() status differs from the isLoggedIn2() status. In this case the user didn’t click on the logout link and is therefore still recognized but isLoggedIn2() returns false.

Customer Methods

A customer object holds data for the logged in customer. This object is different from a NetSuite customer record, because it holds information for the given shopping session only.

The following Customer methods are available:

# addAddress(address)
# addCreditCard(creditcard)
# emailCustomer(subject, body)
# getAddress(addressid, fields)
# getAddressBook(fields)
# getCampaignSubscriptions(fields)
# getCampaignSubscriptions(subscriptionId, fields)
# getCreditCard(creditcardid, fields)
# getCreditCards(fields)
# getCustomFields()
# getCustomFieldValues()
# getFieldValues(fields)
# isGuest()
# removeAddress(addressid)
# removeCreditCard(creditcardid)
# setLoginCredentials(customer)
# updateAddress(address)
# updateCampaignSubscriptions(subscriptions)
# updateCreditCard(creditcard)
# updateProfile(customer)

addAddress(address)

Adds an address for current customer.

Parameters

# address [required] [Object with values for fields]
  # addressee [required]
  # addr1 [required]
# Customer Methods

## addCreditCard(creditcard)

Adds a credit card for current customer.

**Note:** This method is not compatible with SuiteCommerce 2019.1 if the Payments Instruments feature is enabled. SuiteScript methods should be used instead.

### Parameters

- creditcard [required] {Object with values for fields}
  - ccnumber [required]
  - ccname [required]
  - authcode [optional]
  - customercode [optional]
  - paymentmethod [required] (numerical ID for credit card type in NetSuite)
  - expmonth [required]
  - expyear [required]
  - validfrommon [required for UK]
  - validfromyear [required for UK]
  - debitcardissueno [required for UK]
  - ccdefault [optional]
Returns

String - key of added credit card.

Supported Domains

Checkout

Login Required?

Yes

Back to Customer Methods | Back to Shopping Objects

emailCustomer(subject, body)

Sends email to current customer with given subject and body.

Parameters

# subject [required] {String}
# body [required] {String}

Returns

No value returned.

Supported Domains

Checkout

Login Required?

Yes

Back to Customer Methods | Back to Shopping Objects

getAddress(addressid, fields)

Gets address with given ID for current customer.

Parameters

# addressid [required] {string}
# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Object of type address.

Supported Domains

Checkout
Login Required?
Yes

**getAddressBook(fields)**

Gets all addresses for current customer

**Parameters**

`# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned`  

**Returns**

Array of objects of type `address`.

**Supported Domains**

Checkout

Login Required?
Yes

**getCampaignSubscriptions(fields)**

Gets subscription information for the current customer

**Parameters**

`# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned`  

**Returns**

Array of objects of type `subscriptions`.

**Supported Domains**

Checkout, Shopping

Login Required?
Yes

**getCampaignSubscriptions(subscriptionId,fields)**

Gets data for a single campaign subscription with passed id.
Parameters

# subscriptionId [required] {String}
# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Array of objects of type `subscriptions`

Supported Domains

Checkout, Shopping

Login Required?

Yes

getCreditCard(creditcardid, fields)

Gets credit card with given ID for current customer.

Note: This method is not compatible with SuiteCommerce 2019.1 if the Payments Instruments feature is enabled. SuiteScript methods should be used instead.

Parameters

# creditcard [required] {Object with values for fields}
   # internalid [required]
   # ccnumber [required]
   # ccname [required]
   # authcode [optional]
   # customercode [optional]
   # paymentmethod [required]
   # expmonth [required]
   # expyear [required]
   # validfrommon [required for UK]
   # validfromyear [required for UK]
   # debtcardissueno [required for UK]
   # ccdefault [optional]

Returns

Object of type `creditcard`.

Supported Domains

Checkout
Login Required?

Yes

Back to Customer Methods | Back to Shopping Objects

**getCreditCards(fields)**

Gets all credit cards for current customer.

---

### Note:
This method is not compatible with SuiteCommerce 2019.1 if the Payments Instruments feature is enabled. SuiteScript methods should be used instead.

---

**Parameters**

- customer [required] {Object with values for fields}
  - internalid [required]
  - email [optional]
  - emailsubscribe [optional]
  - firstname [optional]
  - lastname [optional]
  - middlename [optional]
  - phoneinfo [optional]
  - customfields [optional] {Object with {customfield id : customfield value} }

**Returns**

Array of objects of type creditcard.

**Supported Domains**

Checkout

Login Required?

Yes

Back to Customer Methods | Back to Shopping Objects

**getCustomFields()**

Gets custom fields on customer record for current customer.

**Parameters**

No parameters to set.

**Returns**

Object with custom field names.
Supported Domains
Checkout, Shopping

Login Required?
No

getCustomFieldValues()

Gets custom field values on customer record for current customer.

Parameters
No parameters to set.

Returns
Object with custom field values. Multiple field values are separated by the Unicode character 'ENQUIRY' ("\u0005").

Supported Domains
Checkout, Shopping

Login Required?
Yes

getFieldValues(fields)

Gets standard field values on customer record for current customer.

Parameters

```
#   fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned
```

Returns
Object of type customer.

Supported Domains
Checkout

Login Required?
Yes

Back to Customer Methods | Back to Shopping Objects
isGuest()

Checks whether the shopper is logged in as a guest. Returns False when the shopper is not logged in or is logged in as a customer.

Parameters

No parameters to set.

Returns

Boolean

Supported Domains

Checkout, Shopping

Login Required?

No

removeAddress(addressid)

Removes address with given ID from current customer’s list of addresses.

Parameters

# addressid [required] {string}

Returns

No value returned.

Supported Domains

Checkout

Login Required?

Yes

removeCreditCard(creditcardid)

Removes given credit card from current customer.

Note: This method is not compatible with SuiteCommerce 2019.1 if the Payments Instruments feature is enabled. SuiteScript methods should be used instead.

Parameters

# creditcardid [required] {string}
Note: Websites on SuiteCommerce versions earlier than 2019.1 must have a patch applied to be compatible with the Payments Instruments feature. See <link to patch documentation>. If the correct patch has been applied, the Payment Instruments feature is enabled, and the credit card has been tokenized, the creditcardid parameter should contain the numerical ID of the payment token.

Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes

Back to Customer Methods | Back to Shopping Objects

setLoginCredentials(customer)
Sets login credentials for current guest customer.

Parameters

# customer [required] {Object with values for fields}
  # internalid [required]
  # email [required]
  # password [required]
  # customfields [Optional] {Object with {customfield id : customfield value} }

Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes

Back to Customer Methods | Back to Shopping Objects

updateAddress(address)
Updates a specified address.

Parameters

# address [required] {Object with values for fields}
  # internalid [required]
# addressee [required]
# addr1 [required]
# addr2 [optional]
# addr3 [optional]
# city [required]
# state [required]
# country [required]
# zip [required]
# phone [optional]
# isresidential [optional]
# defaultshipping [optional]
# defaultbilling [optional]

Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes

Back to Customer Methods | Back to Shopping Objects

updateCampaignSubscriptions(subscriptions)

Updates a specified campaign subscription.

Parameters

# subscriptions [required] {Object with values for fields}
  # internalid [required]
  # subscribed [required]
  
  Customer can only opt-in or opt-out of an email campaign category by setting subscribed to ‘T’ or ‘F’.

Returns
No value returned.

Supported Domains
Checkout, Shopping

Login Required?
Yes
updateCreditCard(creditcard)

Updates given credit card for current customer.

Parameters

- creditcard [required] {Object with values for fields}
  - internalid [required]
  - ccnumber [required]
  - ccname [required]
  - authcode [optional]
  - customercode [optional]
  - paymentmethod [required]
  - expmonth [required]
  - expyear [required]
  - validfrommon [required for UK]
  - validfromyear [required for UK]
  - debtcardissueno [required for UK]
  - ccdefault [optional]

Note: If given ccnumber does not match an existing record, internalid is ignored and a new credit card record is added for customer.

Returns

String - key of added or updated credit card.

Supported Domains

Checkout

Login Required?

Yes

updateProfile(customer)

Updates profile of current customer.

Parameters

- customer [required] {Object with values for fields}
Customer Methods

```
# internalid [required]
# email [optional]
# emailsubscribe [optional]
# firstname [optional]
# lastname [optional]
# middlename [optional]
# phoneinfo [optional]
# customfields [optional] {Object with {customfield id : customfield value} }
```

Returns

No value returned.

Supported Domains

Checkout

Login Required?

Yes

Order Methods

An order object holds shopping cart data and methods for placing the order. This object is different from a NetSuite order record, because it holds information for the given shopping session only.

The following Order methods are available:

```
# addFreeGiftItem(item)
# addItem(item)
# addItems(items)
# applyGiftCertificate(giftcertificate)
# applyPromotionCode(promocode)
# estimateShippingCost(address)
# setItemExcludedFromShipping(cartItemid, excluded)
# getAppliedGiftCertificate(giftcertificate, fields)
# getAppliedGiftCertificates(fields)
# getAppliedPromotionCode(promocode, fields)
# getAppliedPromotionCodes(fields)
# getAvailableShippingMethod(shipmethodid, fields)
# getAvailableShippingMethods(orderLineIds, shipAddressId)
# getAvailableShippingMethods(fields)
# getBillingAddress()
# getCustomFields()
# getCustomFieldValues()
```
# getDeviceFingerPrintingHtml()
# getEligibleFreeGiftItems()
# getFieldValues(fields)
# getItem(orderitemid, fields)
# getItemOption(orderitemid, fields)
# getItems(bConsolidated), getItems(fields, bConsolidated)
# getItems(fields)
# getItemShippingFieldValues(orderitemid)
# getOrderSummary(fields)
# getPayment(fields)
# getPurchaseNumber()
# getShippingAddress()
# getShippingMethod(fields)
# getSummaryTaxTotals()
# getTaxDetails(), getTaxDetails(fields)
# mergeItems(orderLineIds)
# recalculateTaxes()
# rejectFreeGiftItem(item)
# removeAllItems()
# removeBillingAddress()
# removeItem(orderitemid)
# removePayment()
# removePurchaseNumber()
# removePromotionCode(promocode)
# removeAllPromotionCodes()
# removeShippingAddress()
# removeShippingMethod()
# setBillingAddress(addressid)
# setCustomFieldValues(customfields)
# setEnableItemLineShipping()
# setItemOptions(itemlineid, optionMap)
# setItemShippingAddress(orderItemId, shipAddressId)
# setItemShippingAddress(orderItemIds, shipAddressId)
# setItemShippingMethod(orderItemId, shipMethodId)
# setItemShippingMethod(orderItemIds, shipMethodId)
# setPayment(payment)
# setPurchaseNumber(purchaseNumber)
# setShippingAddress(addressid)
# setShippingMethod(shipmethod)
# setTermsAndConditions(readandagree)
# splititem(orderLine)
# submit(ordersettings)
# updateItemFulfillmentPreferences(item)
# updateItemsFulfillmentPreferences(items)
# updateItemQuantity(item)
# updateItemQuantities(items)

## addFreeGiftItem(item)

Adds the eligible item to the cart and marks it as a free item.

### Parameters

- **item [required]** {Object with values for fields}
  - **item_id [required]** {Number}
  - **promotion_id [required]** {Number}
  - **quantity [required]** {Number or String}
  - **options [required for item with item options]**

**Note:** `quantity` can be a negative number.

### Returns

String - the orderitemid of added item.

### Supported Domains

Checkout, Shopping

### Login Required?

No

[Back to Order Methods] [Back to Shopping Objects]

## addItem(item)

Adds an item to order.

**Note:** If your selected Out Of Stock Behavior option is **Disallow back orders but display out-of-stock message** or **Remove out-of-stock items from store**, you might still be able to add an out of stock item to your cart using the `addItem(item)` order method. However, when you try to place the order with such out of stock item in your shopping cart, an error message is displayed and you will not be able to place the order successfully.

### Parameters

- **item [required]** {Object with values for fields}
  - **internalid [required]**
  - **quantity [required]** {Number or String}
  - **options [required for item with item options]**
addItems(items)

Adds multiple items to cart/order within a single request.

Parameters

- items [required] (Array of objects with values for fields)
  - internalid [required]
  - quantity [required] (Number or String)
  - options [required for item with item options]
  - fulfillmentPreferences [optional] (fulfillmentPreferences object)

Returns

String - the orderitemid of added items.

Supported Domains

Checkout, Shopping

Login Required?

No

Back to Order Methods | Back to Shopping Objects

applyGiftCertificate(giftcertificate)

Applies given gift certificate to the order.

Note: quantity can be a negative number.

Returns

String - the orderitemid of added item.

Supported Domains

Checkout, Shopping

Login Required?

No

Back to Order Methods | Back to Shopping Objects
applyPromotionCode(promocode)
Applies given promotion code to the order.

Parameters
- promocode [required] {String}

Returns
No value returned.

Supported Domains
Checkout, Shopping

Login Required?
No

estimateShippingCost(address)
Gets estimated shipping cost for given address.

Note: Estimated shipping cost depends on order’s shipping method, enabling of shipping estimator on site administration page, and whether order requires shipping.

Parameters
- address [optional] {Object with fields}
  - zip [required]
  - country [required]
  - state [optional]
Returns
Number

Supported Domains
Checkout, Shopping

Login Required?
No

Back to Order Methods | Back to Shopping Objects

setItemExcludedFromShipping(cartItemId, excluded)
Identifies when an item on an order is excluded from shipping.

Parameters

# cartItemId [required] {string}
# excluded [required] {boolean}

Returns
No value returned.

Supported Domains
Checkout, Shopping

Login Required?
No

Back to Order Methods | Back to Shopping Objects

getAppliedGiftCertificate(giftCertificate, fields)
Gets specified gift certificate.

Parameters

# giftCertificate [required] Certificate(giftCertificate, fields)
# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns
Object of type giftCertificate.

Supported Domains
Checkout, Shopping
getAppliedGiftCertificates(fields)

Gets all applied gift certificates.

Parameters

# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Array of objects of type giftcertificate

Supported Domains

Checkout

Login Required?

No

getAppliedPromotionCode(promocode, fields)

Gets specified promotion code.

Parameters

# promoCode [required] {String}
# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Object of type promocode

Supported Domains

Checkout, Shopping

Login Required?

No

getAppliedPromotionCodes(fields)

Gets all applied promotion codes.
Parameters

# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Array of objects of type promocode.

Supported Domains

Checkout, Shopping

Login Required?

No

getAvailableShippingMethods(orderLineIds, shipAddressId)

Returns available shipping methods and addresses for the given item in the order.

Note: The setEnableItemLineShipping() flag must be set before using this method. See setEnableItemLineShipping().

Parameters

# orderLineIds [required] Array of orderitemids to merge
# shipAddressId [required] String

Returns

Array of objects of type shipmethod.

Supported Domains

Checkout

Login Required?

No

getAvailableShippingMethod(shipmethodid,fields)

Gets specified shipping method.

Parameters

# shipmethodid [required] [String]
# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned
Returns
Object of type shipmethod.

Supported Domains
Checkout, Shopping

Login Required?
No

getAvailableShippingMethods(fields)
Gets all available shipping methods.

Parameters
# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns
Array of objects of type shipmethod.

Supported Domains
Checkout, Shopping

Login Required?
No

getBillingAddress()
Gets billing address for order.

Parameters
No parameters to set.

Returns
Object of type address.
getCustomFields()

Gets custom fields on order record.

Parameters

No parameters to set.

Returns

Object with custom field names.

Supported Domains

Checkout, Shopping

Login Required?

No

getCustomFieldValues()

Gets custom field values for current order.

Parameters

No parameters to set.

Returns

Object with custom field values.

Supported Domains

Checkout, Shopping

Login Required?

Yes

getDeviceFingerPrintingHtml()

Sets device fingerprinting for CyberSource for the device a shopper is logged in from when accessing a web store. This can be used with the Cybersource payment gateway for added site security. The API call must be made in the last step of checkout. For more details on device fingerprinting in SuiteCommerce Advanced sites, see the help topic Device Fingerprinting.

Parameters

No parameters to set.
Returns

Returns an HTML string that needs to be rendered on the last page of a checkout that includes the Submit Order button.

Supported Domains

Checkout

Login Required?

Yes

getEligibleFreeGiftItems()

Gets a list of all Free Item promotions that are currently applied. It also gets the Item IDs of all the free items.

Parameters

No parameters to set.

Returns

Array of objects with the following fields:

- promotion_id
- eligible_items
- eligible_quantity
- rejected_quantity
- added_quantity

Supported Domains

Checkout, Shopping

Login Required?

No

getFieldValues(fields)

Gets standard field values for the current order.

Parameters

# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Object of type order.
**Supported Domains**
Checkout, Shopping

**Login Required?**
No

**Back to Order Methods | Back to Shopping Objects**

**getItem(orderitemid, fields)**

Gets specified order item.

**Parameters**

- **# orderitemid [required] {String}**
- **# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned**

**Returns**
Object of type orderitem.

**Supported Domains**
Checkout, Shopping

**Login Required?**
No

**Back to Order Methods | Back to Shopping Objects**

**getItemOption(orderitemid, fields)**

Gets item option for specified order item.

**Parameters**

- **# orderitemid [required] {String}**
- **# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned**

**Returns**
The options field from the object of type orderitem.

**Supported Domains**
Checkout, Shopping

**Login Required?**
No
getItems(bConsolidated), getItems(fields, bConsolidated)

Used to configure multiple shipping routes on web orders. Depending on the value for bConsolidated, order items are returned in either a split view or in a consolidated view.

For example, displaying an un-consolidated view of all the items in the cart allows shoppers to select the shipping address for each item. You can then use the consolidated view to display items to be shipped, grouped by shipping address.

Note: The setEnableItemLineShipping() flag must be set before using this method. See setEnableItemLineShipping().

Parameters

- **bConsolidated** Boolean
- **fields** Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Object of type `orderitem`.

Supported Domains

Checkout

Login Required?

No

getItems(fields)

Gets order items.

Parameters

- **fields** [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Array of objects of type `orderitem`.

Supported Domains

Checkout, Shopping

Login Required?

No
**getItemShippingFieldValues(orderitemid)**

Gets shipping field values for specified order item. This API is only available when the Multiple Shipping Routes feature is enabled.

**Parameters**

# orderitemid [required] {String}

**Returns**

Object with fields:

# shipaddress

# shipmethod

**Supported Domains**

Checkout

**Login Required?**

Yes

**Back to** Order Methods | **Back to** Shopping Objects

---

**getOrderSummary(fields)**

Gets the order summary.

⚠️ **Important:** For the most optimized results, use the itemcount argument with this method. When this method is called with no arguments, all information about the order is returned which results in poor performance.

**Example**

```javascript
var itemcount = nlapiGetWebContainer().getShoppingSession().getOrder().getOrderSummary(['itemcount']);
```

**Parameters**

# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

**Returns**

Object of type ordersummary.

**Supported Domains**

Checkout, Shopping

**Login Required?**

No

**Back to** Order Methods | **Back to** Shopping Objects
getPayment(fields)

Gets payment details for order.

Parameters

# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

Object of type payment.

Supported Domains

Checkout

Login Required?

Yes

getPurchaseNumber()

Gets the Purchase order number associated with the order.

Parameters

No parameters to set.

Returns

String

Supported Domains

Checkout

Login Required?

Yes

getShippingAddress()

Gets shipping address for order.

Parameters

No parameters to set.

Returns

Object of type address
getShippingMethod(fields)

Gets shipping method for order.

Parameters

```
# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned
```

Returns

Object of type `shipmethod`.

Supported Domains

Checkout, Shopping

Login Required?

Yes

Back to Order Methods | Back to Shopping Objects

getSummaryTaxTotals()

Gets details of the tax amount for each type of tax applicable for a sales order. This method works only when the SuiteTax feature is enabled. For information on SuiteTax, see the help topics SuiteTax, Working with Taxes using Commerce API and SuiteCommerce SuiteTax Support.

Parameters

No parameters to set.

Returns

```
# If SuiteTax is enabled, returns object of type `summarytax`.
# If SuiteTax is not enabled, returns error message.
```

Supported Domains

Checkout, Shopping

Login Required?

Yes
getTaxDetails(), getTaxDetails(fields)

Gets details of tax applicable for each item in the sales order. Use these methods only if the SuiteTax feature is enabled. For information on SuiteTax, see the help topics SuiteTax, Working with Taxes using Commerce API and SuiteCommerce SuiteTax Support.

Parameters

# fields [optional] Array of field names to be included in returned JSON object; if omitted, all supported fields are returned

Returns

# If SuiteTax is enabled, returns object of type taxdetails.
# If handling is set up as separate from shipping, returns separate taxdetails objects for shipping and handling. If multiple tax types are applicable to the order, returns one taxdetails object per tax type per item.
# If SuiteTax is not enabled, returns error message.

Supported Domains

Checkout, Shopping

Login Required?

Yes

mergeItems(orderLineIds)

Merges the given line items into a single order line item. The given order lines can only differ in quantity and shipping address and method.

Note: The setEnableItemLineShipping() flag must be set before using this method. See setEnableItemLineShipping().

Parameters

# orderLineIds[required]: Array of orderitemids to merge

Returns

No value returned.

Supported Domains

Checkout

Login Required?

Yes
Sample Code

```
nlapiGetWebContainer().getShoppingSession().getOrder().mergeItems(orderLineIds);
```

Back to Order Methods | Back to Shopping Objects

recalculateTaxes()

Triggers tax calculation for a sales order. Use this method only if the SuiteTax feature is enabled.

You must call this method to calculate taxes for a sales order every time you need updated tax details. Taxes are not calculated or recalculated unless this method is called.

SuiteTax uses external tax engines to calculate taxes and this can have an impact on performance. You should limit the use of this method to situations where tax recalculation is necessary. For information on SuiteTax, see the help topics SuiteTax, Working with Taxes using Commerce API and SuiteCommerce SuiteTax Support.

Parameters

No parameters to set.

Returns

# If calculation is successful, does not return a value.
# If calculation fails, returns error message.
# If SuiteTax is not enabled, returns error message.

Supported Domains

Checkout, Shopping

Login Required?

Yes

Back to Order Methods | Back to Shopping Objects

rejectFreeGiftItem(item)

Allows shoppers to reject the free item provided the item is not already in the cart.

Parameters

# item [required] {Object with values for fields}
  # promotion_id [required] {Number}
  # quantity [required] {Number or String}

Note: quantity can be a negative number.

Returns

No value returned.
Supported Domains
Checkout, Shopping

Login Required?
No

removeAllGiftCertificates()
Removes all gift certificates from an order.

Parameters
No parameters to set.

Returns
No value returned.

Supported Domains
Checkout, Shopping

removeAllItems()
Removes all items from order.

Parameters
No parameters to set.

Returns
No value returned.

Supported Domains
Checkout, Shopping

Login Required?
No
**removeBillingAddress()**

Removes the current billing address from the order.

**Parameters**

No parameters to set.

**Returns**

No value returned.

**Supported Domains**

Checkout

**Login Required?**

yes

---

**removeItem(orderitemid)**

Removes specified item from order.

**Parameters**

#  orderitemid [required]  {String}

**Returns**

No value returned.

**Supported Domains**

Checkout, Shopping

**Login Required?**

No

---

**removeItemOptions(itemlineid)**

Removes specified item options from a row in the cart.

**Parameters**

#  itemlineid [required]
removePayment()
Removes the current payment method for the order.

Parameters
No parameters to set.

Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes

Back to Order Methods | Back to Shopping Objects

removePurchaseNumber()
Removes the associated Purchase order number from order.

Parameters
No parameters to set.

Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes
removePromotionCode(promocode)
Removes specified promotion code from order.

Parameters
# promocode [required] {String}

Returns
No value returned.

Supported Domains
Checkout, Shopping

Login Required?
No

removeAllPromotionCodes()
Removes all promotion codes from the order.

Parameters
No parameters to set.

Returns
No value returned.

Supported Domains
Checkout, Shopping

Login Required?
No

removeShippingAddress()
Removes the current shipping address from the order.

Parameters
No parameters to set.
Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes

**Back to** Order Methods | **Back to** Shopping Objects

removeShippingMethod()
Removes the current shipping method for the order.

Parameters
No parameters to set.

Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes

**Back to** Order Methods | **Back to** Shopping Objects

setBillingAddress(addressid)
Sets the billing address for the order.

Parameters

```
# addressid [required] [String]
```

Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes
**setCustomFieldValues(customfields)**

Sets custom field values for order.

**Parameters**

# customfields [required] {Object with name/value pair for each field}

⚠️ Important: Custom fields of the type **Document** are NOT supported for non-Employee external centers.

**Returns**

No value returned.

**Supported Domains**

Checkout

**Login Required?**

Yes

**setEnableItemLineShipping()**

Toggles the item line shipping flag for the current order. This flag **must** be set prior to using other Multiple Ship To APIs.

⚠️ Important: The Multiple Shipping Routes feature is required.

An exception is thrown when:

# The Multiple Shipping Routes feature is not enabled
# The site is not using the Standard Sales Order Form for scripting

**Parameters**

# enable [required]: Boolean

**Returns**

No value returned.

**Supported Domains**

Checkout

**Login Required?**

Yes
Sample Code

```javascript
nlapiGetWebContainer().getShoppingSession().getOrder().setEnableItemLineShipping(true);
```

**Back to** Order Methods | **Back to** Shopping Objects

**setItemOptions(itemlineid,optionMap)**

Updates the item options for a particular cart row.

**Parameters**

- `# itemlineid [required]`
- `# optionMap [required] {Array of Item Option ID value pairs}`

**Note:** `itemlineid` is a combination of item id and set. This is included because of the possibility for multiple quantities of the same item but with different item options.

**Returns**

`itemLineId` of the new line with the new item options.

**Supported Domains**

Checkout, Shopping

**Login Required?**

No

**Back to** Order Methods | **Back to** Shopping Objects

**setItemShippingAddress(orderItemId, shipAddressId)**

Sets the shipping address for the given order line item.

**Note:** The `setEnableItemLineShipping()` flag must be set before using this method. See `setEnableItemLineShipping()`.

**Parameters**

- `# orderItemId[required]: String`
- `# shipAddressId[required]: String`

**Returns**

No value returned.

**Supported Domains**

Checkout
Login Required?
Yes

Sample Code

```
nlapiGetWebContainer().getShoppingSession().getOrder().setItemShippingAddress(orderItemId, shipAddressId);
```

**Back to** Order Methods  |  **Back to** Shopping Objects

**setItemShippingAddress(orderItemIds, shipAddressId)**

Sets the shipping address for an array of given order line items.

**Parameters**

- `orderItemIds [required]`: Array of orderId fields.
- `shipAddressId [required]`: String

**Returns**

No value returned.

**Supported Domains**

Checkout

Login Required?
Yes

**Back to** Order Methods  |  **Back to** Shopping Objects

**setItemShippingMethod(orderItemId, shipMethodId)**

Sets the shipping method for the given order line item.

**Parameters**

- `orderItemId [required]`: String.
- `shipMethodId [required]`: String

**Returns**

No value returned.
Supported Domains
Checkout

Login Required?
Yes

Sample Code

```javascript
nlapiGetWebContainer().getShoppingSession().getOrder().setItemShippingMethod(orderItemId, shipMethodId);
```

**Back to Order Methods | Back to Shopping Objects**

**setItemShippingMethod(orderItemIds, shipMethodId)**

Sets the shipping method for an array of given order line items.

**Note:** The setEnableItemLineShipping() flag must be set before using this method. See setEnableItemLineShipping().

**Parameters**

# orderItemIds [required]: Array of orderItemId fields.
# shipMethodId [required]: String

**Returns**

No value returned.

Supported Domains
Checkout

Login Required?
Yes

**Back to Order Methods | Back to Shopping Objects**

**setPayment(payment)**

Sets payment for order; for credit card, creates a new record if internalid not provided.

**Parameters**

# payment [required] {Object with values for fields}
  # internalid [optional]
  # *ccname [optional]
  # *ccnumber [optional]
  # *expmonth [optional]
  # *expyear [optional]
# paymentmethod [required] (If Payment Instruments feature is enabled, this contains the numerical ID of payment card token)
# paymentterms [optional]
# creditcard [required] (this is an object within the payment object.)

Note: * Fields are required only for credit card payment methods.

For information on creating non-credit card payment methods, see the help topic Alternative Credit Card Payment Methods.

Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes

**setPurchaseNumber(purchaseNumber)**

Sets the purchase number for the order.

Parameters

# purchaseNumber [required] (String)

Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes

**setShippingAddress(addressid)**

Sets the shipping address for the order.

Parameters

# addressid [required] (String)
Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes

**setShippingMethod(shipmethod)**

Sets shipping method for order.

**Parameters**

```plaintext
#  shipmethod [required] {Object with values for fields}
  #  shipmethod [required]
  #  shipcarrier [required]
```

**Note:** Valid values that can be set for shipcarrier are ups or noups.

Returns
No value returned.

Supported Domains
Checkout

Login Required?
Yes

**setTermsAndConditions(readandagree)**

If required by site, sets whether user has read and agreed to terms and conditions.

**Parameters**

```plaintext
#  readandagree [required] {Boolean}, or {String}, where value must be “T” or “F”
```

Returns
No value returned.

Supported Domains
Checkout
splititem(orderLine)

Splits the given line item into multiple order line items. You can then set a shipping address for each order line.

Note: The setEnableItemLineShipping() flag must be set before using this method. See setEnableItemLineShipping().

Parameters

- orderLine [required] {Object with values for fields}
- orderitemid [required]
- quantities [optional] An array of quantities to split into. If not provided, all of the line items from the split are given a quantity of 1.

Returns

No value returned.

Supported Domains

Checkout

Login Required?

Yes

Sample Code

```javascript
nlapiGetWebContainer().getShoppingSession().getOrder().splitItem(orderitem);
```

submit(ordersettings)

Places the shopping order.

Parameters

- checkoutsettings

Returns

Object with fields:

- status
- internalid
- confirmationnumber
Supported Domains

Checkout

Login Required?

Yes

Back to Order Methods | Back to Shopping Objects

updateItemFulfillmentPreferences(item)

Updates specified item's fulfillment choices for the order.

Parameters

# item [required] {Object with values for fields}
  # orderitemid [required]
  # fulfillmentPreferences [required] {fulfillmentPreferences object}

Returns

No value returned.

Supported Domains

Checkout, Shopping

Login Required?

No

Back to Order Methods | Back to Shopping Objects

updateItemsFulfillmentPreferences(items)

Updates specified items' fulfillment choices for the order.

Parameters

# items [required] {Array of objects with values for fields}
  # orderitemid [required]
  # fulfillmentPreferences [required] {fulfillmentPreferences object}

Returns

No value returned.

Supported Domains

Checkout, Shopping

Login Required?

No
**updateItemQuantity(item)**

Updates specified item’s quantity for order.

**Parameters**

- `# item [required] {Object with values for fields}
  - # orderitemid [required]
  - # quantity [required] {Number or String}

**Returns**

No value returned.

**Supported Domains**

Checkout, Shopping

**Login Required?**

No

**updateItemQuantities(items)**

Updates specified items' quantities for order.

**Parameters**

- `# items [required] {Array of objects with values for fields}
  - # orderitemid [required]
  - # quantity [optional] {Number or String} Defaults to 1 if omitted.

**Returns**

No value returned.

**Supported Domains**

Checkout, Shopping

**Login Required?**

No

**Understanding Multiple Ship To**

APIs for Multiple Ship To provide the ability to code a checkout experience that allows shoppers to ship items in their cart to multiple shipping addresses.
Before working with the Multiple Ship To APIs, ensure the **Multiple Shipping Routes** feature is enabled in your account. For detailed information on the Multiple Shipping Routes feature, see the help topic **Multiple Shipping Routes**.

The set of APIs related to Multiple Ship To include:

```plaintext
# updateItemFulfillmentPreferences(item)
# updateItemsFulfillmentPreferences(items)
# setEnableItemLineShipping()
# splitItem(orderLine)
# mergeItems(orderLineIds)
# setItemShippingAddress(orderItemId, shipAddressId)
# setItemShippingAddress(orderItemIds, shipAddressId)
# setItemShippingMethod(orderItemId, shipMethodId)
# setItemShippingMethod(orderItemIds, shipMethodId)
# getItems(bConsolidated), getItems(fields, bConsolidated)
# getAvailableShippingMethods(orderLineIds, shipAddressId)
# addItem(item)
# addItems(items)
```

**Known Limitations**

- **Promotions**: Multiple Ship To does not work with promotions with shipping impact. The promo code is automatically removed from the order.
- **SuiteTax**: Multiple Ship To does not work in combination with SuiteTax.
- **One shipping method per shipping address**: Multiple Ship To is limited to defining one shipping method per shipping address. Shoppers cannot create multiple shipments to the same address. For example, a shopper cannot have one set of items shipping to their Home address using a Standard shipping method and another set of items shipping to their Home address using a One Day shipping method.
- **Real time rates and performance**: There is a known performance impact on the Proceed to Checkout flow when a ship method with real time rates is used. This can be exacerbated for orders with Multiple Ship To. Ensure that you do perform performance testing before going live when using real time rates.
- **Login Required**: In order to use the above mentioned Order methods for Multiple Ship To, you must be logged in.

**Working with Taxes using Commerce API**

There are two tax calculation features available in NetSuite — SuiteTax and Legacy Taxes. SuiteTax uses internal or external tax engines to calculate taxes for transactions in NetSuite. At present, SuiteTax is available only for SuiteCommerce web stores that cater to the US and Canada. See the help topics **SuiteTax** and **SuiteCommerce SuiteTax Support** for more information about SuiteTax.

**Important**: SuiteTax is not supported for tax calculation on SiteBuilder websites.

### SuiteTax

SuiteTax calculates the total taxes applicable for each item in a sales order. The following methods retrieve SuiteTax tax calculations for a web store order:
# recalculateTaxes() - triggers tax calculation for a sales order. Taxes are not calculated or recalculated unless this method is called.

# getTaxDetails(), getTaxDetails(fields) - retrieves tax details for each line item in the sales order, including items, shipping, and handling (if set up as separate from shipping). If multiple tax types apply to a sales order, this method retrieves a taxdetails object for each tax type for each line item.

  # The taxdetailsreference field for an item in the orderitem object matches the taxdetailsreference field in the related taxdetails objects returned by getTaxDetails(). If getTaxDetails() does not return any taxdetails objects with a matching taxdetailsreference field, it means no taxes are applicable for that item.

  # The taxamount field contains the total of the taxes applicable for the item. If multiple taxes apply to a line item, this is the total amount for the tax type in the taxtype field.

# getOrderSummary(fields) - retrieves the fields:

  # taxonshipping - shipping tax amount.
  # txtotal - total taxes for the whole order.

# getSummaryTaxTotals() - retrieves the tax amount per tax type for the whole sales order. If multiple tax types apply to a sales order, this method returns multiple summarytax objects.

Legacy Taxes

Note: The field used for legacy taxes must not be used if SuiteTax is enabled.

In the United States, per item taxes are not applied to an order. Instead tax is applied to the entire order. The following tax fields are present in the orderitem object and are only used when countries require per line tax:

  # taxtype1
  # taxrate1
  # taxtype2
  # taxrate2

Since every line can have a different tax type or tax rate you should define the taxes per order line. When a line is not taxed, omit those fields.

Note: You should also check the value of pricesincludevat in the sitesettings object to verify whether taxes are included in the price. These fields are only required when tax is not included in the price.

PageGenerator Methods

Methods on the pageGenerator object are used to generate a web store page.

The following PageGenerator methods are available:

  # addBreadCrumb(label, url)
  # addHeadHtml(html)
  # addStyleSheetHtml(html)
  # addTab(id, label, url)
  # addToPageInitScript(scriptLines)
  # removeBreadCrumbs()
  # removeTab(id)
addBreadCrumb(label, url)

Adds a bread crumb to the end of the existing bread crumb trail.

Parameters

- label [required] {string} - text displayed for the bread crumb
- url [required] {string} - URL to which the bread crumb links

addHeadHtml(html)

Adds the specified HTML to the top of the page's <head> section, before the stylesheet HTML.

Parameters

- html [required] {string} - HTML content

addStylesheetHtml(html)

Adds the specified HTML to the top of the page's <head> section, after the stylesheet HTML.

Parameters

- html [required] {string} - HTML content

addTab(id, label, url)

Adds a tab to the right of current tabs on the page.

Parameters

- id [required] {string} - unique string id for the tab
- label [required] {string} - label displayed within the tab
- url [required] {string} - URL to which the tab links

addToPageInitScript(scriptLines)

Adds lines of JavaScript to the page's initialization script.
Parameters

# scriptLines [required] {string}

removeBreadCrumbs()
Removes all bread crumbs from the page.

Parameters
No parameters to set.

removeTab(id)
Removes a specified tab from the page.

Parameters

# id [required] {string} - unique string id for the tab

removeTabs()
Removes all tabs from the page.

Parameters
No parameters to set.

setDocTypeHtml(html)
Establishes HTML for the page <doctype> tag.

Parameters

# html [required] {string} - HTML content

setMetaTagHtml(html)
Establishes HTML for the page's <meta> tags.

Parameters

# html [required] {string} - HTML content
setPageTitle(title)
Establishes a title for the page.

Parameters
#

Back to PageGenerator Methods | Back to Shopping Objects

setSelectedTab(id)
Causes the specified tab to be displayed as the currently active tab.

Parameters
#

Back to PageGenerator Methods | Back to Shopping Objects

showTabNavigation(show)
Indicates whether tab navigation should be shown or hidden.

Parameters
#

StandardTagLibrary Methods

Methods on a standardTagLibrary object implement NetSuite web store tags.

Each of the following methods returns the same result as the corresponding NetSuite page tag. All results are strings.
#

SuiteCommerce API Documentation
# getLogoutLinkHtml()
# getLogoutUrl()
# getPageFullHead()
# getPageHead()
# getPageLinks()
# getPageTabs()
# getPageTop()
# getRegionSelectHtml()
# getRegistrationLinkHtml()
# getRegistrationUrl()
# getReloginLinkHtml()
# getReloginUrl()
# getSideBarWidth()
# getSiteLogoHtml()
# getSiteNavigationHtml()
# getUserInfo()
# getUserInfo2()
# getWelcomeImageHtml()
JSON Object Fields

 Applies to: SuiteCommerce Web Stores

Shopping methods are implemented with JSON objects. **Set** methods can have JSON objects passed in as input parameter values and **get** methods return strings extracted from JSON objects. Note that parsing of returned JSON values is done when values are set to rows (for example, when cart items are built).

Tables for the following shopping method objects list the fields exposed for each. These fields are for use in web store carts and orders. For convenience, some fields are exposed here that can also be accessed through the existing SuiteScript API.

```
# order
# address
# analyticssettings
# checkoutsettings
# creditcard
# customer
# discounts_impact
# discounts
# fulfillmentPreferences
# giftcertificate
# item
# message
# orderitem
# ordersettings
# ordersummary
# payment
# paymentmethod
# promocode
# registrationsettings
# shipmethod
# sitecategory
# sitesettings
# status
# threedsecure
# touchpoints
```

**order**

This object contains other objects, and includes the complete set of data for a web store shopping order.
## Object Name

<table>
<thead>
<tr>
<th>Object Name</th>
<th>Object Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>billaddress</td>
<td>JSON object (type of address)</td>
<td>Billing address for order</td>
</tr>
<tr>
<td>customer</td>
<td>JSON object</td>
<td>Current shopper information</td>
</tr>
<tr>
<td>customfields</td>
<td>JSON object</td>
<td>List of custom fields name/value pairs set for order</td>
</tr>
<tr>
<td>giftcertificate</td>
<td>Array</td>
<td>Gift certificates applied to order</td>
</tr>
<tr>
<td>items</td>
<td>Array of orderitem</td>
<td>Items added to order</td>
</tr>
<tr>
<td>payment</td>
<td>JSON object</td>
<td>Payment information for order</td>
</tr>
<tr>
<td>promocode</td>
<td>Array</td>
<td>Promotion codes applied to order</td>
</tr>
<tr>
<td>purchaseNumber</td>
<td>JSON object</td>
<td>Purchase order number for the order</td>
</tr>
<tr>
<td>shipaddress</td>
<td>JSON object (type of address)</td>
<td>Shipping address for order</td>
</tr>
<tr>
<td>shipmethod</td>
<td>JSON object</td>
<td>Shipping method for order</td>
</tr>
<tr>
<td>status</td>
<td>JSON object</td>
<td>Status of order</td>
</tr>
<tr>
<td>summary</td>
<td>JSON object (type of ordersummary)</td>
<td>Summary information for order (subtotal, total, tax, etc.)</td>
</tr>
</tbody>
</table>

### address

This object contains address information for a web store shopping order.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>addressee</td>
<td>string</td>
<td>Person to which order is sent</td>
<td>yes</td>
<td>yes (reqd)</td>
</tr>
<tr>
<td>addr1</td>
<td>string</td>
<td>First line of address</td>
<td>yes</td>
<td>yes (reqd)</td>
</tr>
<tr>
<td>addr2</td>
<td>string</td>
<td>Second line of address</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>addr3</td>
<td>string</td>
<td>Third line of address</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>attention</td>
<td>string</td>
<td>Text for ATTN: line of address</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>city</td>
<td>string</td>
<td>City</td>
<td>yes</td>
<td>yes (reqd)</td>
</tr>
<tr>
<td>country</td>
<td>string</td>
<td>Country</td>
<td>yes</td>
<td>yes (reqd)</td>
</tr>
<tr>
<td>defaultbilling</td>
<td>string</td>
<td>Indicates whether address is default billing address</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>defaultshipping</td>
<td>string</td>
<td>Indicates whether address is default shipping address</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>internalid</td>
<td>string</td>
<td>Internal ID</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>isresidential</td>
<td>string</td>
<td>Indicates whether address is residential</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phone</td>
<td>string</td>
<td>Phone number</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
### Address

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>string</td>
<td>State for address</td>
<td>yes</td>
<td>yes (reqd)</td>
</tr>
<tr>
<td>zip</td>
<td>string</td>
<td>Zip code</td>
<td>yes</td>
<td>yes (reqd)</td>
</tr>
</tbody>
</table>

Note: Address forms on Reference Checkout 2.04 and earlier require the phone number field.

### AnalyticsSettings

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>clickattributes</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>confpagetrackinghtml</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>submitattributes</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

### CheckoutSettings

This object contains information used for checkout.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>cancelurl</td>
<td>string</td>
<td>Full URL path for redirect after cancel action during order submission</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used for PayPal Express support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>continueurl</td>
<td>string</td>
<td>Full URL path for redirect after order submission</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used for PayPal Express support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>createorder</td>
<td>boolean</td>
<td>If value is set to T, backend submits current shopping order when Continue link is clicked</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value is F</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used for PayPal Express support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>custchoosespaymethod</td>
<td>string</td>
<td>Value should be T or F.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>paymentauthorization</td>
<td>threedsecure object with fields</td>
<td>Settings to handle payment authorization</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used for 3D Secure support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>paymentmandatory</td>
<td>string</td>
<td>Indicates whether payment is mandatory</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>paypalexpress</td>
<td>Object with fields</td>
<td>Used for PayPal Express support</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
## checkoutsettings

Includes the following fields:

- **available** (boolean) (must be set to T to support PayPal Express)
- **imageurl** (string)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>requiretermsandconditions</td>
<td>string</td>
<td>Indicates whether terms and conditions text is required</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>saveccinfo</td>
<td>string</td>
<td>Indicates whether credit card should be saved by default</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shippingaddrfirst</td>
<td>string</td>
<td>Indicates whether shipping address should be displayed first</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>showpurchaseorder</td>
<td>string</td>
<td>Indicates whether purchase order should be displayed</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>showsavecc</td>
<td>string</td>
<td>Indicates whether the Save Credit Card field should be displayed</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>termsandconditions</td>
<td>string</td>
<td>Text of Terms and Conditions field</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Used for integration with third party checkout providers</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid values are: paypalexpress, google</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### creditcard

This object contains credit card information.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccdefault</td>
<td>string</td>
<td>Indicates whether credit card is the default</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cccexpiredate</td>
<td>string</td>
<td>Card expiration date</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>ccname</td>
<td>string</td>
<td>Name on card</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>cccnumber</td>
<td>string</td>
<td>Card number</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value returned by Get function is masked.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ccssecuritycode</td>
<td>string</td>
<td>Card security code</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Field Name</td>
<td>Field Type</td>
<td>Description</td>
<td>In Get Functions?</td>
<td>In Set Functions?</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>creditcard</td>
<td></td>
<td>Only needed if payment method is creditcard.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>customercode</td>
<td>string</td>
<td>Customer code</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>debtcardissueno</td>
<td>string</td>
<td>Card issue number for UK</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Note: Not exposed in Customer record.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expmonth</td>
<td>string</td>
<td>Card expiration month</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>expyear</td>
<td>string</td>
<td>Card expiration year</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>internalid</td>
<td>string</td>
<td>Internal ID of credit card.</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>If the Payment Instruments feature is enabled, and the credit card has been tokenized, this contains the internal ID of payment card token.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paymentmethod</td>
<td>Object of type paymentmethod</td>
<td>Payment method type</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Note: For credit card payments, this is the numerical code of the credit card type.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If the Payment Instruments feature is enabled, and the credit card has been tokenized, this is the numerical code of the payment token type.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>savecard</td>
<td>string</td>
<td>Indicates whether card should be saved</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Value should be T or F.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>validfrom</td>
<td>string</td>
<td>Card valid from date</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Note: Not exposed in Customer record.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>validfrommon</td>
<td>string</td>
<td>Card valid from month for UK</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>validfromyear</td>
<td>string</td>
<td>Card valid from month for UK</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

**customer**

This object contains information about the logged in customer.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>addressbook</td>
<td>List of JSON address objects</td>
<td>Address information</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>balance</td>
<td>string</td>
<td>Customer’s accounts receivable balance due, shown in customer’s currency.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>campaignsubscriptions</td>
<td>List of JSON subscription objects</td>
<td>Information on campaign subscriptions for the given customer.</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Field Name</td>
<td>Field Type</td>
<td>Description</td>
<td>In Get Functions?</td>
<td>In Set Functions?</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>companyname</td>
<td>string</td>
<td>Customer company name</td>
<td>yes (for Checkout</td>
<td>yes (for Checkout</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>and Shopping</td>
<td>domain only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>domains)</td>
<td></td>
</tr>
<tr>
<td>creditcards</td>
<td>List of JSON</td>
<td>Credit card information</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>creditcard objects</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>creditlimit</td>
<td>string</td>
<td>Maximum currency amount the customer is allowed to accrue in outstanding</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>receivables.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>string</td>
<td>Customer email address</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required in Set function for customer registration, guest registration, and</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>login.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>emailsubscribe</td>
<td>string</td>
<td>Email subscription</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>firstname</td>
<td>string</td>
<td>Customer first name</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>internalid</td>
<td>string</td>
<td>Internal ID</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>isperson</td>
<td>boolean</td>
<td>If false this customer is of customer type. If true this customer is of the</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>type individual.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastname</td>
<td>string</td>
<td>Customer last name</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>middlename</td>
<td>string</td>
<td>Customer middle name</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Customer name</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Important:** As of Version 2013 Release 1, this field is deprecated and included for backwards compatibility only. The firstname, lastname, and middlename fields are the preferred fields for getting and setting customer names.

| partner       | string              | Partner                                                                     | yes               | yes               |
| password      | string              | Password                                                                    | no                | yes               |
| password hint | string              | Hint                                                                       | no                | yes               |
| paymentterms  | JSON object of      | Payment terms associated with the customer                                | yes               | no                |
|               | fields              |                                                                             |                   |                   |
| phoneinfo     | JSON object of      | Returns or sets the phone number information for a customer. For example:   | yes               | yes               |
|               | fields              | {"phone":"65460780", "fax": "40540", "altphone" : null}                   |                   |                   |
discounts_impact

This object contains information about the overall discount impact for a given item.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>discounts</td>
<td>Array of JSON</td>
<td>Discounts applied to the order.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>object of type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>discounts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>total_amount</td>
<td>number</td>
<td>Total currency amount of the discount applied to the order.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>total_amount_formatted</td>
<td>string</td>
<td>Discount amount formatted with currency symbol.</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

discounts

This object contains information about the per line discount impact for a given item.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>promotion_couponcode</td>
<td>string</td>
<td>The name of the coupon code. This corresponds to what the user applies during the checkout process. For example, user might apply SUMMER or FREESHIP.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>promotion_name</td>
<td>string</td>
<td>Name of the associated promotion.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>promotion_id</td>
<td>integer</td>
<td>ID of the associated promotion.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>discount_item</td>
<td>string</td>
<td>Name of the associated discount item.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>discount_item_id</td>
<td>integer</td>
<td>ID of the associated discount item.</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
Field Name | Field Type | Description | In Get Functions? | In Set Functions?
--- | --- | --- | --- | ---
amount | number | Currency amount of the discount applied to the order as a result of applying this coupon code. | yes | no
amount_formatted | string | Discount amount formatted with currency symbol. | yes | no
is_auto_applied | boolean | Indicates whether the promocode was automatically or manually applied for the current order. Value should be true or false. | yes | no

**fulfillmentPreferences**

This object contains information about the fulfillment choices for a web store shopping order.

Field Name | Field Type | Description | In Get Functions? | In Set Functions?
--- | --- | --- | --- | ---
fulfillmentChoice | string | Indicates the type of fulfillment, and its value should be **ship** or **pickup**. | yes (reqd) | yes (reqd)

**Note:** The value **pickup** can be used only when the Store Pickup feature is enabled.

pickupLocationId | integer | Indicates the internal ID of the pickup location, and is available only if the fulfillment choice is **pickup**. | yes | no

shippingAddressId | integer | Indicates the shipping address ID, and is available only if the fulfillment choice is **ship**. | yes | no

**Note:** This can only be used only when the Multiple Ship To feature is enabled.

shippingMethodId | integer | Indicates the shipping method ID, and is available only if the fulfillment choice is **ship**. | yes | no

**Note:** This can only be used only when the Multiple Ship To feature is enabled.

giftcertificate

This object contains information about gift certificates in the order.

Field Name | Field Type | Description | In Get Functions? | In Set Functions?
--- | --- | --- | --- | ---
amountapplied | number | Amount applied to order | yes | no
### Field Table

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>amountremaining</td>
<td>string</td>
<td>Remaining gift certificate amount after applied to order</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>giftcertcode</td>
<td>string</td>
<td>Gift certificate code</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>originalamount</td>
<td>number</td>
<td>Original amount of gift certificate</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

### item

This object contains basic information about an item that can be included in the order.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>featuredescription</td>
<td>string</td>
<td>Featured description</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>internalid</td>
<td>string</td>
<td>Internal ID for item</td>
<td>yes</td>
<td>yes (reqd)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unique identifier that should be used when adding an item to the cart</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isavailable</td>
<td>string</td>
<td>Indicates whether item is available for web store order</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isdonationitem</td>
<td>string</td>
<td>Indicates whether item is donation item</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isdropshipitem</td>
<td>string</td>
<td>Indicates whether item is a dropship item</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be Yes or No.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isspecialorderitem</td>
<td>string</td>
<td>Indicates whether item is a special order item</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be Yes or No.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isstorespecial</td>
<td>string</td>
<td>Indicates whether item is on special in the web store</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>istaxable</td>
<td>string</td>
<td>Whether the item is taxable</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be Yes or No.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>itemtype</td>
<td>string</td>
<td>Type of the item, such as inventory, kit/package, etc.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>nopricemessage</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>nopricemessage2</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>outofstockmessage</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>outofstockmessage2</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>outofstocknobackorder message</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Field Name</td>
<td>Field Type</td>
<td>Description</td>
<td>In Get Functions?</td>
<td>In Set Functions?</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>-------------------------------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>pagetitle</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>pagetitle2</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>quantityavailable</td>
<td>string</td>
<td>Number of items available for ordering</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>rate</td>
<td>string</td>
<td>Unit price</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>rate_formatted</td>
<td>string</td>
<td>Unit price formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedetaileddescription</td>
<td>string</td>
<td>Store detailed description</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedescription</td>
<td>string</td>
<td>Store description</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedisplayimage</td>
<td>string</td>
<td>Store display image</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedisplayname</td>
<td>string</td>
<td>Store display name</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedisplayname2</td>
<td>string</td>
<td>Second store display name</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedisplaythumbnail</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>freegiftpromotionid</td>
<td>string</td>
<td>ID of the free gift promotion this item belongs to</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

**message**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>string (with HTML markup)</td>
<td>Text of message</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Type of message</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td># Confirmation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td># Error</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td># Information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td># Warning</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**orderitem**

The `orderitem` object is a subtype of `item`.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount</td>
<td>string</td>
<td>Currency amount for items</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>amount_formatted</td>
<td>string</td>
<td>Amount formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>internalid</td>
<td>string</td>
<td>Internal ID for item</td>
<td>yes</td>
<td>yes (reqd)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unique identifier that should be used when adding an item to the cart</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Name</td>
<td>Field Type</td>
<td>Description</td>
<td>In Get Functions?</td>
<td>In Set Functions?</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------</td>
<td>----------------------------------------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Display name</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>options</td>
<td>JSON object</td>
<td>List of name/value pairs</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>orderitemid</td>
<td>string</td>
<td>Unique ID for an item in an order</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>promotionamount</td>
<td>string</td>
<td>Amount after any line-level discounts</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>promotionamount_formatted</td>
<td>string</td>
<td>Amount after any line-level discounts formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>promotiondiscount</td>
<td>string</td>
<td>Amount of discount related to promotions</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>promotiondiscount_formatted</td>
<td>string</td>
<td>Amount of discount related to promotions formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>quantity</td>
<td>string</td>
<td>Number of items in order</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>rateschedule</td>
<td>string</td>
<td>Price schedule for quantity pricing</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>shipaddress</td>
<td>string</td>
<td>ID of shipping address for item</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>shipcarrier</td>
<td>string</td>
<td>Shipping carrier</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Note: Valid values that can be set for shipcarrier are ups or noups.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shipmethod</td>
<td>JSON object of type shipmethod</td>
<td>ID of shipping method for item Used when the Multiple Shipping Routes feature is enabled</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>taxamount</td>
<td>string</td>
<td>The total tax amount for the line item.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxamount</td>
<td>string</td>
<td>The total tax amount for the line item.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxdetailsreference</td>
<td>string</td>
<td>The ID that links the tax details returned by getTaxDetails() with the item to which they apply. Applicable only for tax calculations using SuiteTax. Tax calculation using SuiteTax is available only for US and Canada.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxtype1</td>
<td>string</td>
<td>Type of the item tax. For example, GST/VAT/WET. Not applicable if SuiteTax is enabled.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxrate1</td>
<td>string</td>
<td>Rate of the tax in percentage. Not applicable if SuiteTax is enabled.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Field Name</td>
<td>Field Type</td>
<td>Description</td>
<td>In Get Functions?</td>
<td>In Set Functions?</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>taxtype2</td>
<td>string</td>
<td>For Canada only. Currently returns PST. Not applicable if SuiteTax is enabled.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxrate2</td>
<td>string</td>
<td>For Canada only. The rate of PST tax in percentage when applicable. Not applicable if SuiteTax is enabled.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>tax1amt</td>
<td>string</td>
<td>The total tax amount for the line item. Not applicable if SuiteTax is enabled.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>fulfillmentPreferences</td>
<td>JSON object of type fulfillmentPreferences</td>
<td>Fulfillment preference for item</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>discounts_impact</td>
<td>JSON object of type discounts_impact</td>
<td>Per line discount impact for the line item.</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

**ordersettings**

This object contains general settings for the order.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemdisplayorder</td>
<td>string</td>
<td>Ordering of items in an order</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>outofstockbehavior</td>
<td>string</td>
<td>Behavior for out of stock items</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>upselldisplay</td>
<td>string</td>
<td># Related Items Only</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td># Upsell Items</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># Related Items and then Upsell Items</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># Upsell Items and then Related Items</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ordersummary**

This object contains summary information for the order.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>discountedsubtotal</td>
<td>number</td>
<td>Item subtotal - any applicable discount</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>discountedsubtotal_formatted</td>
<td>string</td>
<td>Item subtotal - any applicable discount formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>discountrate</td>
<td>string</td>
<td>Raw discount</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>discounttotal</td>
<td>number</td>
<td>Total amount of discounts applied</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Field Name</td>
<td>Field Type</td>
<td>Description</td>
<td>In Get Functions?</td>
<td>In Set Functions?</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>discounttotal_formatted</td>
<td>string</td>
<td>Total amount of discounts applied formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>giftcertapplied</td>
<td>number</td>
<td>Total amount of gift certificates applied</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>giftcertapplied_formatted</td>
<td>string</td>
<td>Total amount of gift certificates applied formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>handlingcost</td>
<td>number</td>
<td>Amount of handling cost</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>handlingcost_formatted</td>
<td>string</td>
<td>Amount of handling cost formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>itemcount</td>
<td>number</td>
<td>Returns the quantity of items in the cart.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Important:</strong> Use this argument with the getOrderSummary method for the most optimized call.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that the value returned equals the total quantity of items in the cart which may differ from the total lines in the cart. For example, if a shopper has three of the same items in the cart, the call returns 3.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shippingcost</td>
<td>number</td>
<td>Amount of shipping cost</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>shippingcost_formatted</td>
<td>string</td>
<td>Amount of shipping cost formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>subtotal</td>
<td>number</td>
<td>Total of quantity * rate per line item</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>subtotal_formatted</td>
<td>string</td>
<td>Total of quantity * rate per line item formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>tax2total</td>
<td>number</td>
<td>Total secondary tax on taxable elements of the order (items + shipping + handling) - effect of any discounts. Not applicable if SuiteTax is enabled.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>tax2total_formatted</td>
<td>string</td>
<td>Total secondary tax on taxable elements of the order (items + shipping + handling) - effect of any discounts formatted with currency symbol. Not applicable if SuiteTax is enabled.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxondiscount</td>
<td>number</td>
<td>For VAT countries, currency amount of the tax rate * the discount amount</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxondiscount_formatted</td>
<td>string</td>
<td>For VAT countries, currency amount of the tax rate * the discount amount formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxonhandling</td>
<td>number</td>
<td>Currency amount of the tax rate * taxable handling amount</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxonhandling_formatted</td>
<td>string</td>
<td>Currency amount of the tax rate * taxable handling amount formatted with currency symbol</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
### Field Name | Field Type | Description | In Get Functions? | In Set Functions?
---|---|---|---|---
| taxonshipping | number | Currency amount of the tax rate * taxable shipping amount | yes | no |
| taxonshipping_formatted | string | Currency amount of the tax rate * taxable shipping amount formatted with currency symbol | yes | no |
| taxtotal | number | Total tax on taxable elements of the order (items + shipping + handling) - effect of any discounts | yes | no |
| taxtotal_formatted | string | Total tax on taxable elements of the order (items + shipping + handling) - effect of any discounts formatted with currency symbol | yes | no |
| totalcombinedtaxes | number | A grand total of all the tax components (taxtotal + tax2total) associated with an order | yes | no |
| totalcombinedtaxes_formatted | string | A grand total of all the tax components (taxtotal + tax2total) associated with an order formatted with currency symbol | yes | no |
| total | number | Order subtotal + shipping, handling, and all tax, less the effect of any discounts | yes | no |
| total_formatted | string | Order subtotal + shipping, handling, and all tax, less the effect of any discounts formatted with currency symbol | yes | no |

### payment

This object contains payment information for the order.

| Field Name | Field Type | Description | In Get Functions? | In Set Functions? |
---|---|---|---|---
| creditcard | JSON creditcard object | Payment information when credit card is selected as the payment method. **Note:** See Formatting Values for Credit Card Payments for an example of how to format this information. If the Payment Instruments feature is enabled, and the credit card has been tokenized, this field contains the internal ID and payment method of payment card token. | yes | yes |
| paymentmethod | JSON paymentmethod object | Details of the payment method selected by customer. | yes | yes |
| paymentterms | string | Payment terms selected by customer. Value should be CreditCard or Invoice. **Note:** Set this value to Invoice to use an invoice as payment. See Using an Invoice as Payment for an example. | yes | yes |
**Field Name** | **Field Type** | **Description** | **In Get Functions?** | **In Set Functions?**
--- | --- | --- | --- | ---
paypal | JSON object | PayPal information if enabled | yes | no
status | JSON status object | Embedded status object used for web store business logic validation | yes | yes

**Formatting Values for Credit Card Payments**

The following example code illustrates how to format details for a payment using a new credit card:

```javascript
function service(request, response) {
    var payment = {
        creditcard: {
            'ccname': 'Herb Joseph',
            'ccnumber': '4111111111111111',
            'expmonth': '09',
            'expyear': '2013',
            'paymentmethod': {
                'internalid': '7',
                'name': 'Visa',
                'ispaypal': 'F',
                'creditcard': 'T',
                'key': '8,8,1135464'
            }
        }
    };

    var order = nlapiGetWebContainer().getShoppingSession().getOrder();
    order.setPayment(payment);
}
```

The following example code illustrates how to format details for a payment using a stored credit card:

```javascript
function service(request, response) {
    var payment = {
        creditcard: {
            'internalid': '35',
            'paymentmethod': {
                'internalid': '7',
                'name': 'Visa',
                'ispaypal': 'F',
                'creditcard': 'T',
                'key': '8,8,1135464'
            }
        }
    };

    var order = nlapiGetWebContainer().getShoppingSession().getOrder();
    order.setPayment(payment);
}
```
Using an Invoice as Payment

The following example code illustrates how to use an invoice as a payment method:

```javascript
function service(request, response)
{
    var retval = null;

    try
    {
        var payment = {
            paymentterms : 'Invoice'
        }; 
        nlapiGetWebContainer().getShoppingSession().getOrder().setPayment(payment);
    }
    catch (e)
    {
        var nle = nlapiCreateError(e);
        retval = { status : 'error', reasoncode : nle.getCode(), message : nle.getDetails()};
    }
    response.writeLine(JSON.stringify(retval));
}
```

**paymentmethod**

This object includes payment method information for the order.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalid</td>
<td>string</td>
<td>Internal ID</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>creditcard</td>
<td>string</td>
<td>Indicates whether payment method is a credit card</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the Payments Instruments feature is enabled, this contains T if the payment method is a payment card token.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>creditcardtoken</td>
<td>string</td>
<td>Indicates whether payment method is a payment card token.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Applicable only if the Payments Instruments feature is enabled.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ispaypal</td>
<td>string</td>
<td>Indicates whether payment method is PayPal</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the payment method</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>paypalemailaddress</td>
<td>string</td>
<td>Primary paypal email address</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Field Name</td>
<td>Field Type</td>
<td>Description</td>
<td>In Get Functions?</td>
<td>In Set Functions?</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>--------------------------------------------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>merchantid</td>
<td>string</td>
<td>Internal ID of the merchant account linked with the payment method</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>isexternal</td>
<td>string</td>
<td>Indicates whether payment method is handled outside of NetSuite Value should be T or F.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>imagesrc</td>
<td>string</td>
<td>Returns the URL of the image file used to represent the payment method.</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Note: You cannot set an invoice as a payment method. Instead, set invoice as the paymentterms value for the payment object. See Using an Invoice as Payment for an example.

promocode

This object contains promotion code information for the order.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>discountamount</td>
<td>string</td>
<td>Currency amount of the discount applied to the order.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>errmsg</td>
<td>string</td>
<td>Returns the error message when a promo code is not valid.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>internalid</td>
<td>string</td>
<td>Internal ID</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>isvalid</td>
<td>string</td>
<td>Indicates whether promocode is valid for the current order. Value should be T or F.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>promocode</td>
<td>string</td>
<td>The value of the Coupon Code ID. This corresponds to what the user sets during the checkout process.</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>promocodeid</td>
<td>string</td>
<td>The value of the Promocode ID. This corresponds to the promocode displayed in the NetSuite UI.</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Note: The value of this field is not changed when a coupon code is modified.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>purchase_discount_amount</td>
<td>string</td>
<td>Currency amount of the discount applied to the order.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>purchase_discount_amount_formatted</td>
<td>string</td>
<td>Discount amount formatted with currency symbol.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>shipping_discount_amount</td>
<td>string</td>
<td>Currency amount of the discount applied to the shipping cost.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>shipping_discount_amount_formatted</td>
<td>string</td>
<td>Discount shipped amount formatted with currency symbol.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>promotion_name</td>
<td>string</td>
<td>Name of the promotion associated with the applied coupon.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Field Name</td>
<td>Field Type</td>
<td>Description</td>
<td>In Get Functions?</td>
<td>In Set Functions?</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>discount_name</td>
<td>string</td>
<td>Name of the discount associated with the applied promotion.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>discount_type</td>
<td>string</td>
<td>Indicates the type of discount, and its value should be PERCENTAGE or FLAT.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>discount_rate</td>
<td>string</td>
<td>Rate of the discount in percentage if the type of discount is PERCENTAGE and currency amount of the discount if the type of discount is FLAT.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>is_auto_applied</td>
<td>boolean</td>
<td>Indicates whether the promocode was automatically or manually applied for the current order. Value should be true or false.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>applicability_status</td>
<td>string</td>
<td>Indicates whether the promocode is actually applied to the current order or not. Value should be one of the following:</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td># APPLIED — Indicates that the promocode is applied to the current order and you get a discount if the promocode is valid. It also indicates that the type of promotion is SuitePromotion.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># NOT_APPLIED — Indicates that the promocode does not give you a discount even though it is added to the current order. It also indicates that the type of promotion is SuitePromotion.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># NOT_AVAILABLE — Indicates that the applicability status information is not available. You get this value in case of Standard promotions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>applicability_reason</td>
<td>string</td>
<td>Specifies the reason for not applying the promocode.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be one of the following:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># DISCARDED_BEST_OFFER — Indicates that the promocode is not applied as there is another promocode or a combination of promocodes that is giving you a better discount.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># CRITERIA_NOT_MET — Indicates that the promocode is not applied as it does not meet the criteria. For example, if your order does not meet the minimum order amount requirement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># NOT_AVAILABLE — Indicates that the promocode is applied or the promotion is a Standard promotion.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>promotion_type</td>
<td>string</td>
<td>Indicates the type of promotion.</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
### registrationsettings

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>companyfieldmandatory</td>
<td>string</td>
<td>Indicates whether Company field is mandatory</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>registrationallowed</td>
<td>string</td>
<td>Indicates whether registration is allowed</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>registrationanonymous</td>
<td>string</td>
<td>Indicates whether registration is anonymous</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>registrationmandatory</td>
<td>string</td>
<td>Indicates whether registration is required</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>registrationoptional</td>
<td>string</td>
<td>Indicates whether registration is optional</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>showcompanyfield</td>
<td>string</td>
<td>Indicates whether to show Company field on registration form</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

### shipmethod

This object contains shipping method information for the order.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the shipping method</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>rate</td>
<td>string</td>
<td>Rate of the shipping method</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>shipcarrier</td>
<td>string</td>
<td>Shipping carrier ID</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Note:** Valid values that can be set for shipcarrier are ups or noups.

### sitecategory

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>canonicalurl</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>categorythumbnailhtml</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
### Field Data

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>description2</td>
<td>string</td>
<td>The value from the web site category record for the field Brief Description.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>internalid</td>
<td>string</td>
<td>Internal ID of the site category.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>itemfields</td>
<td>string</td>
<td>An array of field names returned for each category.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>itemid</td>
<td>string</td>
<td>Name of site category</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>metataghtml</td>
<td>string</td>
<td>The value from the web site category record for the field Meta Tag Html.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>pagetitle</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>pagetitle2</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>searchkeywords</td>
<td>string</td>
<td>The value from the web site category record for the field Search Keywords.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedetaileddescription</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedetaileddescription2</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedisplayimage</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedisplayimagehtml</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storedisplaythumbnail</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>storeurl</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>urlcomponent</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>urlcomponentaliases</td>
<td>array</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>welcomepagetitle</td>
<td>string</td>
<td></td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

### Site Data

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>analytics</td>
<td>JSON analyticssettings object</td>
<td>Analytics settings</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>checkout</td>
<td>JSON checkoutsettings object</td>
<td>Checkout settings</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>cookiepolicy</td>
<td>string</td>
<td>Returns the URL for the cookie policy file in the NetSuite file cabinet.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>currencies</td>
<td>Array of JSON objects with fields (internalid, name, isdefault, symbolplacement)</td>
<td>Ordered list of currencies supported by the site. isdefault field is a Boolean, with a value of T for default currency. F for others. symbolplacement uses the constants: SYMBOL_BEFORE_NUMBER = 1 and SYMBOL_AFTER_NUMBER = 2</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>defaultshipcountry</td>
<td>string</td>
<td>Default ship to country</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>defaultshippingmethod</td>
<td>string</td>
<td>Default shipping method</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>displayname</td>
<td>string</td>
<td>Site name</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Field Name</td>
<td>Field Type</td>
<td>Description</td>
<td>In Get Functions?</td>
<td>In Set Functions?</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>imagesizes</td>
<td>Array of JSON objects with image resize definitions (resized, resizew)</td>
<td>Each object has the following properties:</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td># maxWidth–maximum resized image width in pixels</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># maxHeight–maximum resized image height in pixels</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># urlSuffix–suffix that should be added to the image URL to return a resized image</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># internalId–internal system ID of image resize definition.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td># name–value from the Image Resize ID field (image resize definition) as entered on the Web Site Setup page.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iswebstoreoffline</td>
<td>boolean</td>
<td>Returns T or F depending on whether the web store is online or offline.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>iswsdk</td>
<td>boolean</td>
<td>Returns T or F depending on the settings of the WSDK checkbox on the site record (tab setup) and the WSDK feature settings for a site.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>languages</td>
<td>Array of JSON objects with fields (internalid, name, isdefault)</td>
<td>Ordered list of languages for the site</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>isdefault field is a Boolean, with a value of T for default language, F for others</td>
<td></td>
<td></td>
</tr>
<tr>
<td>loginallowed</td>
<td>string</td>
<td>Indicates whether login is allowed</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>loginrequired</td>
<td>string</td>
<td>Indicates whether login is required to access the site</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>order</td>
<td>JSON ordersettings object</td>
<td>Order settings</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>paymentmethods</td>
<td>array of paymentmethod objects</td>
<td>Payment methods</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>pricesincludevat</td>
<td>string</td>
<td>Indicates whether prices include VAT</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>registration</td>
<td>JSON registrationsettings object</td>
<td>Registration settings</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>requireloginforpricing</td>
<td>string</td>
<td>Indicates whether login is required to display item prices</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>requireshippinginformation</td>
<td>boolean</td>
<td>Indicates whether shipping information needs to be collected</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shipallcountries</td>
<td>string</td>
<td>Indicates whether shipping is supported to all countries</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shippingrequired</td>
<td>string</td>
<td>Indicates whether shipping is required for the site</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shiptocountries</td>
<td>Array of strings</td>
<td>Countries where shipping is supported</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>showextendedcart</td>
<td>string</td>
<td>Indicates whether extended cart should be displayed</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value should be T or F.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Field Name | Field Type | Description | In Get Functions? | In Set Functions?
--- | --- | --- | --- | ---
showshippingestimator | string | Indicates whether shipping estimator should be displayed. Value should be T or F. | yes | no
sitetype | string | Indicates the type of website. Value should be ADVANCED or STANDARD. ADVANCED indicates the site is a SuiteCommerce Advanced site where STANDARD indicates it is a Standard web site built with the NetSuite Site Builder. | yes | no
subsidiaries | Array of JSON objects with fields (internalid, name, isdefault) | Ordered list of subsidiaries for the site. isdefault field is a Boolean, with a value of T for default subsidiary, F for others. | yes | 
wsdkcancelcarturl | string | For use with an External Catalog Site (WSDK). Returns the Cancel Cart URL. | yes | no
wsdkcancelcheckouturl | string | For use with an External Catalog Site (WSDK). Returns the Cancel Checkout URL. | yes | no
wsdkcancelloginurl | string | For use with an External Catalog Site (WSDK). Returns the Cancel Login URL. | yes | no
wsdkcompletecheckouturl | string | For use with an External Catalog Site (WSDK). Returns the Complete Checkout URL. | yes | no
wsdkcompleteloginurl | string | For use with an External Catalog Site (WSDK). Returns the Complete Login URL. | yes | no

### status

Each JSON object returned by a shopping function has an embedded status object that provides business logic validation of the web store order. The caller of a shopping function, in addition to checking the request status, should also check the status object to determine whether the object can be submitted to the backend and whether there are messages to display.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
</table>
| code | string | Code of the status. Following values are supported: 
- # error = object contains error 
- # incomplete = object is not complete, so cannot be handled by backend 
- # ready = object is complete and ready to be submitted to backend 
- # success = object successfully processed by backend | | |
<p>| fieldstatuses | array of status objects | Status for object fields | | |</p>
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>messages</td>
<td>array of message objects</td>
<td>Status messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reasoncode</td>
<td>string</td>
<td>Code of possible reason for the current status</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**subscriptions**

This object contains information used for campaign subscriptions.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalid</td>
<td>string</td>
<td>The internalid of the campaign subscription.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the customer.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>subscribed</td>
<td>boolean</td>
<td>Indicates whether the customer is currently subscribed to this email campaign category.</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the campaign.</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

**summarytax**

This object contains a breakdown of the different types of taxes applicable to a web store shopping order. For example, in Canada, it would contain the total amount of PST and GST. The summarytax object is used only with tax calculations using SuiteTax. Tax calculation using SuiteTax is available only for US and Canada.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>taxTypeId</td>
<td>string</td>
<td>The ID of the type of tax.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxTypeName</td>
<td>string</td>
<td>The name of the tax type. For example, PST or GST.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>taxTotal</td>
<td>number</td>
<td>The total tax amount for the tax type used on the sales order.</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

**taxdetails**

This object contains tax information for each line item in a web store shopping order. Each taxdetails object contains details for one type of tax, for example PST. The taxdetails object is used only with tax calculations using SuiteTax. Tax calculation using SuiteTax is available only for US and Canada.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>taxdetailsreference</td>
<td>string</td>
<td>The ID that links the tax details in the taxdetails object with the orderitem to which they apply.</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The taxdetails object for shipping has the taxdetailsreference SHIPPING. If handling</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Field Name | Field Type | Description | In Get Functions? | In Set Functions?
---|---|---|---|---
| **linetype** | string | The type of line item in the sales order to which the tax applies. For example, ‘Item’, ‘Shipping’, and ‘Handling’. | yes | no |
| **linename** | string | The name of the line item in the sales order to which the tax applies. For items, this returns the name of the item from the item record, for example “Blue Shirt”. For shipping, this returns the name of the shipping record plus the term “Shipping Cost”, for example “US_CA_Ship_Item_1, Shipping Cost”. If handling is set up as separate from shipping, this returns the name of the handling record plus the term “Handling Cost”, for example “US_CA_Ship_Item_1, Handling Cost”. | yes | no |
| **netamount** | number | The cost of the line item, not including tax. | yes | no |
| **grossamount** | number | The cost of the line item, including tax. | yes | no |
| **taxtype** | string | The type of tax. For example, if the type of tax applicable is PST, this contains the ID of the corresponding tax type record. | yes | no |
| **taxcode** | string | The tax code associated with the tax type. | yes | no |
| **taxbasis** | string | The amount based on which tax is calculated. | yes | no |
| **taxrate** | number | The rate of tax being charged. | yes | no |
| **taxamount** | number | The total tax amount for the line item for the tax type returned in taxtype. | yes | no |
| **calcdetail** | string | The information returned by the tax engine used to calculate the tax. | yes | no |

### threedsecure

This object contains information used for 3D Secure payment integration.

| Field Name | Field Type | Description | In Get Functions? | In Set Functions? |
---|---|---|---|---|
<p>| <strong>noredirect</strong> | string | Default is F If set to T, when 3D Secure authorization is required, instead of a redirect, a status object | no | yes |</p>
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>servicehtml</td>
<td>string</td>
<td>Complete HTML for IFrame that triggers 3D Secure service</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>termurl</td>
<td>string</td>
<td>URL for page or handler where 3D Secure POST is sent</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>threedsecurekey</td>
<td>string</td>
<td>Key used by backend to identify current 3D Secure data object</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

**touchpoints**

Each field in this object stores the URL for a custom page that is integrated to the web store at a selected entry point. These URLs are defined on the Touch Points subtab of the web application record. See the help topic [Select Supported Touch Points](#).

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Description</th>
<th>In Get Functions?</th>
<th>In Set Functions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkout</td>
<td>string</td>
<td>URL for starting the checkout process</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>continueshopping</td>
<td>string</td>
<td>URL for continue shopping page</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>customercenter</td>
<td>string</td>
<td>URL for customer center</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>home</td>
<td>string</td>
<td>URL for home page</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>login</td>
<td>string</td>
<td>URL for customer login</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>logout</td>
<td>string</td>
<td>URL for customer logout</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>register</td>
<td>string</td>
<td>URL for customer registration</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>serversync</td>
<td>string</td>
<td>URL for sync shopper status between shopping and checkout domain</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>viewcart</td>
<td>string</td>
<td>URL for viewing the shopping cart</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>welcome</td>
<td>string</td>
<td>URL for welcome page</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

**Serversync Touchpoint**

Apply to: SuiteCommerce Advanced | Mont Blanc

You can customize your implementation of SuiteCommerce Advanced to use the serversync touchpoint. This touchpoint maintains the identity of the user and cart contents when a user opens items in other tabs or windows. This is useful during email marketing campaigns, when users open a separate instance of the application from a link in an email.

Using the serversync touchpoint requires customizing your implementation of SuiteCommerce Advanced to extend or override the following SuiteCommerce Advanced source files:

```
# CheckoutApplication@X.X.X!SuiteScript!checkout.environment.ssp
```
Important: Making changes to core JavaScript source files or changing any vital functionality of the application can make migrating to future releases difficult. Before making changes to SuiteCommerce Advanced, see the help topics Customize and Extend Core SuiteCommerce Advanced Modules and Best Practices for Customizing SuiteCommerce Advanced

In the case of the serversync touchpoint, NetSuite has provided a customization patch that you can use.

Important: If you have already made customizations to your implementation of SuiteCommerce Advanced, use caution when applying these custom modules.

To apply the serversync touchpoint custom modules:

1. Click the link to download the serversync zip file.
   Serversync.zip
2. Extract the files, then place the Serversync directory and all contents in the Modules > extensions directory for your implementation of SuiteCommerce Advanced.

   Note: If you have already made customizations to the checkout.environment.ssp SuiteScript file in the CheckoutApplication@X.X.X module, run a diff between the new custom version of checkout.environment.ssp and your existing override, incorporate NetSuite’s customizations into your custom module, and remove the EnvironmentCustom@1.0.0 folder from your new Serversync directory.

   Customizations you have made to the Header, ProductList, and Profile modules are only affected if you have modified the same methods and properties that have been extended in this patch. Run a diff on the appropriate files and, if you have made customizations in the same methods and properties, incorporate NetSuite’s changes into your custom module.

3. Go to Modules > extensions > Serversync to locate the distro.json provided with the patch.
4. Open the new distro.json file and locate the appropriate Serversync entries for CheckoutApplication, Header, ProductList, and Profile.
5. Copy the Serversync entries and paste them into the list of modules in your existing distro.json file.

   Important: If you have already made any customizations to the CheckoutApplication module, or customizations to the same methods and properties of the Header, ProductList, or Profile modules in your implementation of SuiteCommerce Advanced, do not use the Serversync entries from the new distro.json.

6. Open a command prompt from your source directory and deploy to NetSuite.

   Note: Since these customizations modify files stored as SSP libraries, changes are not visible in your local environment. You must deploy your custom modules directly to NetSuite. See the help topic Deploy to NetSuite for more information.
SSP Application Governance

Applies to: SuiteCommerce Web Stores

SSP applications have a governance model in accordance with SuiteScript governance model, with the same goal of optimizing performance. SSP application governance is based on usage units. Each shopping method consumes a system-defined number of processing units, and each SSP application script can execute a system-defined number of units. If the number of allowable units is exceeded by an SSP application script, the script is terminated.

The limit for SSP applications is 1000 units per script.

The units consumed per method are as follows:

## Shopping Session Methods
- `nlobjShoppingSessionImpl.logIn(20)`
- `nlobjShoppingSessionImpl.registerCustomer(20)`
- `nlobjShoppingSessionImpl.registerGuest(20)`
- `nlobjShoppingSessionImpl.setShopperCurrency(10)`
- `nlobjShoppingSessionImpl.setShopperLanguageLocale(10)`
- `nlobjShoppingSessionImpl.setShopperSubsidiary(10)`
- `nlobjShoppingSessionImpl.getPaymentMethods(10)`
- `nlobjShoppingSessionImpl.getCountries(10)`
- `nlobjShoppingSessionImpl.getShipToCountries(10)`
- `nlobjShoppingSessionImpl.getStates(10)`
- `nlobjShoppingSessionImpl.getItemFieldValues(10)`
- `nlobjShoppingSessionImpl.getSiteCategoryFieldValues(10)`
- `nlobjShoppingSessionImpl.getInformationItemFieldValues(10)`
- `nlobjShoppingSessionImpl.getMediaItemFieldValues(10)`

## Customer Methods
- `nlobjWebStoreCustomerImpl.addAddress(10)`
- `nlobjWebStoreCustomerImpl.updateAddress(10)`
- `nlobjWebStoreCustomerImpl.removeAddress(10)`
- `nlobjWebStoreCustomerImpl.addCreditCard(10)`
- `nlobjWebStoreCustomerImpl.updateCreditCard(10)`
- `nlobjWebStoreCustomerImpl.removeCreditCard(10)`
- `nlobjWebStoreCustomerImpl.getAddressBook(10)`
- `nlobjWebStoreCustomerImpl.getAddress(10)`
- `nlobjWebStoreCustomerImpl.getCreditCards(10)`
- `nlobjWebStoreCustomerImpl.getCreditCard(10)`

## Order Methods
- `nlobjWebStoreOrderImpl.addItems(20)`
- `nlobjWebStoreOrderImpl.addItem(10)`
- `nlobjWebStoreOrderImpl.removeItem(10)`
- `nlobjWebStoreOrderImpl.updateItemQuantity(10)`
# nlobjWebStoreOrderImpl.updateItemQuantities(20)
# nlobjWebStoreOrderImpl.removeAllItems(10)
# nlobjWebStoreOrderImpl.estimateShippingCost(20)
# nlobjWebStoreOrderImpl.applyPromotionCode(5)
# nlobjWebStoreOrderImpl.removePromotionCode(5)
# nlobjWebStoreOrderImpl.applyGiftCertificate(5)
# nlobjWebStoreOrderImpl.removeAllGiftCertificates(5)
# nlobjWebStoreOrderImpl.setShippingAddress(10)
# nlobjWebStoreOrderImpl.setShippingMethod(5)
# nlobjWebStoreOrderImpl.getAvailableShippingMethods(20)
# nlobjWebStoreOrderImpl.getAvailableShippingMethod(10)
# nlobjWebStoreOrderImpl.submit(20)

Note: There is not a concurrency limit for SSP applications but they are not recommended for high volume catalog pages or to deliver content that should come from a cache.
SuiteScript API & SuiteCommerce

Applies to: SuiteCommerce Web Stores

SuiteScript is a JavaScript API that enables you to programmatically access most NetSuite records, custom records, and event/trigger points. This API is a standard NetSuite feature. For complete details on all of SuiteScript including all APIs and the usage of those APIs, see the help topic SuiteScript 1.0.

SuiteCommerce Advanced uses SuiteScript to access data stored in an NetSuite record. This enables SuiteCommerce Advances to access data that is not accessible from the Item Search API or the Commerce API.

SuiteCommerce Advanced uses SuiteScript in the following contexts:

- Backend models: all backend models are combined into a file called ssp_libraries.js. This file is stored in the root directory of the SSP application.
- Services: all services in SuiteCommerce Advanced are deployed to the services directory within the SSP application.

Being installed within the SSP application enables backend models and services to access the SuiteScript API to access data stored in NetSuite records.

The Case application module, for example, uses SuiteScript to access information stored in a NetSuite case record. When creating a new case, Case.Model uses the following code:

```javascript
var case_record = nlapiCreateRecord('supportcase');
```

The string passed to the nlapiCreateRecord method is the internal identifier for the case record in NetSuite. Calling nlapiCreateRecord creates a new case record in NetSuite. It also returns a copy of the object. This object is stored in the case_record variable.

Storing a record as an object enables you to call additional methods on the object containing the case record. For example, the following code calls the getField method that returns the value of the record field passed to the method:

```javascript
var category_field = case_record.getField('category');
```

In this example, getField returns the value of the category field.

Other modules use the nlapiLoadRecord method to assign a NetSuite record to a variable containing an object. You can then call methods like getField on the object to retrieve the values of specific fields within the record.

In another context, a model may call the nlapiDeleteRecord method to remove a record from NetSuite.