



Reference Cart and One Page Checkout BETA

February 3, 2014

Version 2014 Release 1

Copyright NetSuite, Inc. 2009–2013 All rights reserved.

This document is the property of NetSuite, Inc., and may not be reproduced in whole or in part without prior written approval of NetSuite, Inc.

Trademarks

The following marks are registered trademarks or service marks of NetSuite, Inc. in the United States and other countries.

- NETSUITE
- The "N" in NetSuite Logo
- SUITESCRIPT
- SUITEFLEX
- PREMIER PAYROLL SERVICE

NetSuite OpenAir is provided by NetSuite, Inc.

Other trademarks and service marks used or referenced in this document are the property of their respective owners and are hereby acknowledged.

Table of Contents

| | |
|---|----|
| 1. Reference Cart & One Page Checkout BETA | 1 |
| Reference Cart & One Page Checkout BETA Version 1.0 | 1 |
| Installing Reference Cart & One Page Checkout BETA Version1.0 | 2 |
| Installing Reference Cart & One Page Checkout BETA Version 1.0 | 3 |
| Deploying SSP Application Touch Points | 4 |
| Understanding Reference Cart & One Page Checkout BETA | 5 |
| Reference Cart & One Page Checkout BETA Limitations | 17 |
| Trouble Shooting Extensions to Reference Checkout | 18 |
| Extending Reference Cart & One Page Checkout BETA | 20 |
| Setting up the Custom Checkout Application Folder | 20 |
| Reference Checkout FAQ | 22 |
| Reference Cart & One Page Checkout BETA Template Reference | 23 |
| Creating a Custom Cart and Checkout Solution | 37 |
| Reference Cart & One Page Checkout BETA Version 1.1 | 58 |
| Installing Reference Cart & One Page Checkout Beta Version 1.1 | 58 |
| Support for Google Analytics | 59 |
| Project File for SuiteCloud IDE Compatibility | 61 |
| New JavaScript File for Site Settings | 62 |
| Resolution for Insecure Content Browser Notifications | 62 |
| Combination Files Copied to the Custom Checkout Folder by Default | 63 |
| Template for Terms and Conditions | 64 |
| Option for Shoppers to Save Credit Card Information | 65 |
| Displaying Relevant Delivery Options | 65 |
| Support for Multiple Languages | 66 |
| Layout Enhancements | 67 |

Chapter 1 Reference Cart & One Page Checkout BETA

Important: Reference Cart & One Page Checkout BETA will be deprecated. If you are implementing a new Checkout process for your web store, use the latest **Reference Checkout** implementation. The latest Reference Checkout implementation leverages the same modern, scalable, and extendable HTML and JavaScript architecture as the Reference Shopping and My Account bundles available for SuiteCommerce Advanced sites. Reference Checkout is available for both SuiteCommerce Advanced and Site Builder sites.

For detailed information on **Reference Checkout** see the help topic Reference Checkout.

Reference Cart & One Page Checkout BETA is NetSuite's legacy reference implementation of the web store checkout process, which includes customized login, cart, and checkout pages, along with associated source files and assets. The reference implementation is delivered in a bundle, and it leverages AJAX technology, as well as NetSuite's SSP applications and Commerce API. A web designer or NetSuite partner can use Reference Cart & One Page Checkout BETA as a base from which to create a unique and innovative ecommerce experience.

The reference implementation is supported for multiple web sites, external catalog sites, HTML sites hosted in NetSuite, NetSuite generated sites, and SuiteCommerce Advanced web sites that use the NetSuite shopping cart.

Currently, there are two versions of the reference implementation:

- [Reference Cart & One Page Checkout BETA Version 1.0](#)
- [Reference Cart & One Page Checkout BETA Version 1.1](#)
- To support seamless access between your externally hosted site and Reference Cart & One Page Checkout on the NetSuite web store, you can implement NetSuite's inbound single sign-on feature. See the help topic [Inbound Single Sign-on Access to Web Store](#).

Reference Cart & One Page Checkout BETA Version 1.0

The information below describes the original version of the reference implementation:

- To get started, see [Installing Reference Cart & One Page Checkout BETA Version 1.0](#). This section explains how to install the bundle and set it up to run in a test site. It also outlines the bundle's out of box functionality, describing the touch points (customized pages) where shoppers interact with Reference Cart & One Page Checkout BETA.

- For details about how this solution has been designed and implemented, see [Understanding Reference Cart & One Page Checkout BETA](#). This section outlines the components installed by the bundle and discusses the architecture of the solution. It also discusses some current limitations.
- For tips and example procedures for customizing Reference Cart & One Page Checkout BETA in order to create your own solution, see [Extending Reference Cart & One Page Checkout BETA](#).
- For descriptions of the template files included in the reference implementation, see [Reference Cart & One Page Checkout BETA Template Reference](#).

Note that the topics listed below include information and examples that apply to both Version 1.0 and Version 1.1 of Reference Cart & One Page Checkout BETA:

- [Reference Cart & One Page Checkout BETA Architecture](#)
- [Reference Cart & One Page Checkout BETA Limitations](#)
- [Creating a Custom Cart and Checkout Solution](#)

Installing Reference Cart & One Page Checkout BETA Version 1.0

Reference Cart & One Page Checkout BETA has been packaged as a bundle so you can install all of its components in your NetSuite account and review it in a test site on a sandbox environment prior to deployment in your web store.

This reference solution is an SSP application, supported by NetSuite's SuiteScript Server Pages feature. For details about this feature, see the help topic, [Using SuiteScript Server Pages for Web Store Customizations](#).

Important: Ensure that these features are enabled before you install this bundle: Client SuiteScript, Server SuiteScript, and SuiteScript Server Pages in the SuiteCloud area, and Web Site and Host HTML Files in the Web Presence area.

The bundle includes the **Reference Checkout** SSP Application record and all of the source files and templates used to implement the reference shopping cart and one page checkout. All of these files are locked so that they cannot be modified. After you have installed the bundle, you must deploy the SSP application's touch points (customized pages) to replace default login, register, cart, and checkout pages.

For details about getting started with the reference solution, see:

- [Installing Reference Cart & One Page Checkout BETA Version 1.0](#)
- [Deploying SSP Application Touch Points](#)

For information about customizing the locked files, see [Extending Reference Cart & One Page Checkout BETA](#).

Installing Reference Cart & One Page Checkout BETA Version 1.0

The Reference Cart & One Page Checkout BETA bundle (ID 21011) is available as a **shared** bundle only from production account 3519604. An account administrator or another user with the SuiteBundler permission can install it.

Important: Reference Cart & One Page Checkout BETA will be deprecated. If you are implementing a new Checkout process for your web store, use the latest **Reference Checkout** implementation. The latest Reference Checkout implementation leverages the same modern, scalable, and extendable HTML and JavaScript architecture as the Reference Shopping and My Account bundles available for SuiteCommerce Advanced sites. Reference Checkout is available for both SuiteCommerce Advanced and Site Builder sites.

To install Reference Cart & One Page Checkout BETA:

1. Go to Customization > SuiteBundler > Search & Install Bundles.
2. On the Search & Install Bundles page:
 1. In the Location field, select Production Account.
 2. In the Account ID field, enter **3519604**.
 3. Click Search.
3. Click the link for Reference Cart & One Page Checkout BETA.
4. On the Bundle Details page, click Install.
5. Review the Preview Bundle Install page and click Install Bundle. (Also click OK in the popup that displays.)
6. Installation progress is displayed on the Installed Bundles page. Click Refresh to update this display.

After you have installed the bundle, you can review its installed components.

- SSP Application records are available at Customization > Scripting > SSP Applications. For details, see [Reference Cart & One Page Checkout BETA SSP Application Records](#).
- Source files are available in the Web Site Hosting Files folder in the file cabinet. For details, see [Reference Cart & One Page Checkout BETA Source Files](#).

Deploying SSP Application Touch Points

When you install the Reference Cart & One Page Checkout BETA bundle, the Reference Checkout SSP application and its source files are available in your NetSuite account, but each customized web page does not operate in a site until you deploy its touch point.

To deploy Reference Cart & One Page Checkout touch points to your site:

1. Go to Setup > Site Builder > SSP Applications.
2. On the list page, click the link for Reference Checkout.
3. On the SSP Application record, click Deploy to Site.
4. On the Deploy SSP Application to Site page:

| Deploy | Name | Path |
|-------------------------------------|---------------------|-----------------------------|
| <input checked="" type="checkbox"/> | Register | /checkout/loginregister.ssp |
| <input checked="" type="checkbox"/> | Log In | /checkout/loginregister.ssp |
| <input checked="" type="checkbox"/> | Proceed to Checkout | /checkout/checkout.ssp |
| <input checked="" type="checkbox"/> | View Cart | /checkout/cart.ssp |

1. Choose a Site. To verify that the reference implementation is compatible with your web store, be sure to choose a test site the first time you deploy touch points.
 2. Check Deploy for one or more of the touch points. Note that for each touch point you select here, incoming requests will go to the SSP application page from the reference cart and checkout bundle.
5. Click Save.

You can deploy one touch point at a time or deploy them all at once to see how they work together.

Note: If you want to deploy touch points to multiple sites, repeat these steps for each site.

To verify the touch points have been deployed properly, go to Setup > Site Builder > Set Up Web Site, and view the Touch Points subtab.

Replacing Cart, Login, and Checkout URLs

After you deploy the reference implementation touch points to your site, the default URLs for the shopping cart, login, and checkout pages are replaced with touch point URLs. Note that if you use a custom shopping domain or a custom secure domain, those appear instead of the default NetSuite domains in the table below.

| Page | Touch Point URL |
|---------------------|---|
| Register | https://checkout.netsuite.com/c.ACCT123/checkout/loginregister.ssp? |
| Login | https://checkout.netsuite.com/c.ACCT123/checkout/loginregister.ssp? |
| Proceed to Checkout | https://checkout.netsuite.com/c.ACCT123/checkout/checkout.ssp? |
| View Cart | http://shopping.netsuite.com/c.ACCT123/checkout/cart.ssp?n=1&sc=3 |

Understanding Reference Cart & One Page Checkout BETA

Before you begin extending the reference implementation to create your own solution, you should understand its components, architecture, and limitations:

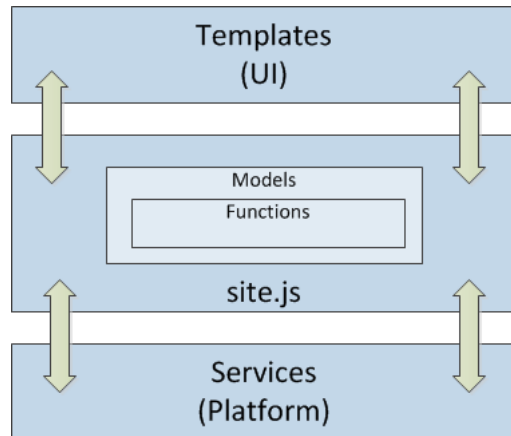
- [Reference Cart & One Page Checkout BETA Architecture](#)
- [Reference Cart & One Page Checkout BETA SSP Application Records](#)
- [Reference Cart & One Page Checkout BETA Source Files](#)
 - [SuiteScript Server Pages \(.ssp\)](#)
 - [JavaScript Files](#)
 - [SuiteScript Files \(.ss\)](#)
 - [CSS Files](#)
 - [Template Files](#)
- [Reference Cart & One Page Checkout BETA Limitations](#)
- [Trouble Shooting Extensions to Reference Checkout](#)

Reference Cart & One Page Checkout BETA Architecture

AJAX technology, on which the reference implementation is based, allows web developers to create a faster browsing experience. Data is sent to and received from the server asynchronously, so only certain sections of the page are refreshed as new data is updated.

The **site.js** file is used to control data communication between the NetSuite platform and the user interface. This file contains several models, such as the address model, the shipping model,

and the credit card model, that are used to encapsulate logic for communicating with NetSuite. Functions in **site.js** make calls to related SuiteScript files that access the NetSuite platform. You can modify **site.js** to affect all aspects of the checkout process.



Technology and Framework

The framework for Reference Cart and One Page Checkout is constructed using *TrimPath* and *Prototype*. You can use HTML and TrimPath to modify the files in the template folder. For more information, see the [JavaScript Templates](#) web site. For examples of syntax, click **JST Markup Syntax**. For more information about Prototype, see the www.prototypejs.org web site.

JSON is the data interchange format used in the reference implementation. For more information, see www.json.org.

Reference Cart & One Page Checkout BETA also supports the following scripting languages: SuiteScript, JavaScript, and HTML.

Note that shipping integration and address look-up are part of the platform exposed by NetSuite's Commerce API. For more information, see the help topic [Commerce API Reference](#).

For further details, read the topics listed below:

- [Using Models](#)
- [Using Templates](#)

Using Models

Using models eliminates the need for JavaScript logic in templates that render the UI. The **site.js** file includes logic that runs the checkout process, observes events, and communicates with the corresponding services (SuiteScript files) to send data to templates. There is a one-to-

one mapping between models and services. All JavaScript related to the UI communicates with a model to fetch data.

Models are responsible for:

- Communicating with services.
- Implementing data related to business logic.
- Implementing data related to cache.

Functions in each model ensure that data and UI elements are updated properly when a customer interacts with checkout on your web store. If you choose to modify any services included in the reference implementation, you must also modify related functions of the model in the **site.js** file.

For example, the address model is the object responsible for all actions related to addresses on your web site, such as find, refresh, get, and save. The following code samples, from **site.js**, illustrate this model.

- This sample shows the save function included in the address model.

```
save : function(mp) {  
    var model = this;  
  
    var options = {  
        params: {  
            method: BDK.undef(mp.args.input.internalid) ? "add" :  
            "update",  
            params : {  
                address: mp.args.input  
            }  
        }  
    };  
    this.comm(options, mp);  
};
```

- This sample calls the save function from the UI code.

```
_nsAddressModel.save({args:{input:input}, callbacks:{success:callback}});
```

Using Templates

Templates define the UI of the site. Data is passed to the templates to produce the resulting site. Templates contain mostly HTML markup with place holders for data.

For example: the code in the **checkout_shippingdelivery.txt** template includes calls to the address model and the shipping model. This template renders address and shipping method information, as well as the button that expands the next step on the checkout page.

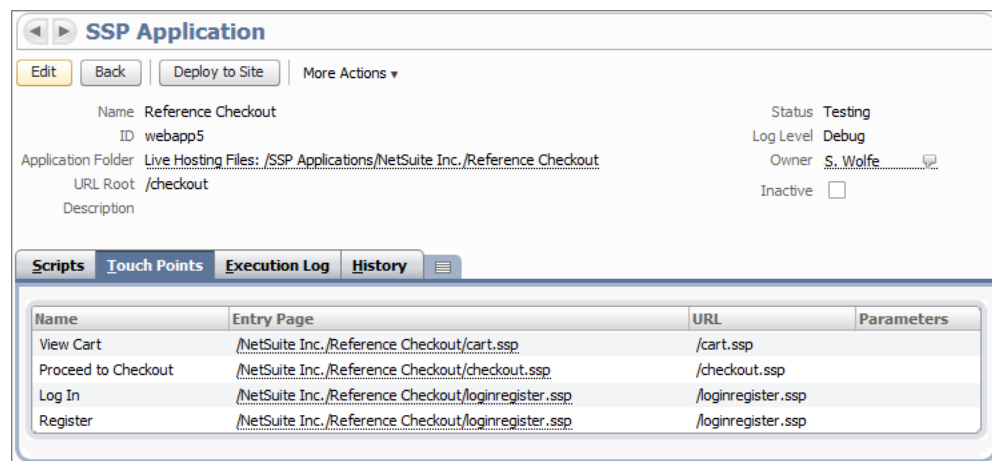
```
<div class="checkout-content clearfix">
  <!-- form -->
  <div class="form clearfix">
    <form action="">
      <fieldset>
        <div class="form-col-wrap clearfix">
          <!-- form col -->
          <div class="form-col clearfix">
            <div id="addresses"></div>
          </div>
          <div class="form-col fl-r clearfix">
            <div id="shippingMethods"></div>
          </div>
        </div>
        <div class="f-row f-row-btn">
          <div class="button"><input type="button"
onclick=BDK.fire("customSaveShippingStep") value="Next Step" /></div>
        </div>
      </fieldset>
    </form>
  </div>
</div>
```

The structure of the template files in the reference implementation is flexible, so you can customize the cart and checkout experience for your customers. For details, see [Template Files](#).

Reference Cart & One Page Checkout BETA SSP Application Records

An SSP Application record provides a single NetSuite object that stores details about a web store customization. This record facilitates grouping of assets, debugging of script files, and packaging of the customization for use in other NetSuite accounts. Reference Cart & One Page Checkout BETA installs two SSP applications: Reference Checkout, and Custom Checkout. You can access these SSP Application records at Setup > Site Builder > SSP Applications.

The **Reference Checkout** SSP Application record links all Reference Cart & One Page Checkout BETA components together.



The **Reference Checkout** SSP Application record defines the following:

- application folder: the physical location of source files in the NetSuite file cabinet

Web Site Hosting Files/Live Hosting Files/SSP Applications/NetSuite Inc/Reference Checkout

- URL root: the externally visible URL used to access pages

/checkout

- four touch points: customized pages that you can deploy to replace the default login, register, cart, and checkout pages.
- scripts: JavaScript files associated with the reference implementation.

The **Custom Checkout** SSP Application record is for use when you create your own solution to extend the reference implementation.

- This SSP application does not include touch points or source files by default.
- You can copy files that you want to customize into the application folder for the Custom Checkout SSP application:

Web Site Hosting Files/Live Hosting Files/SSP Applications/NetSuite Inc/Custom Checkout

- The Custom Checkout SSP application shares the same URL root as the Reference Checkout SSP application:

/checkout

SSP applications support a precedence (sequence) mechanism to define which application's assets should be served when they share a URL root.

| SSP Applications | | | | | |
|------------------|------------|--------------------|---------|-----------|--|
| Edit View | Precedence | Name | ID | URL Root | |
| Edit View | 1 | Custom Checkout | webapp6 | /checkout | |
| Edit View | 2 | Reference Checkout | webapp5 | /checkout | |

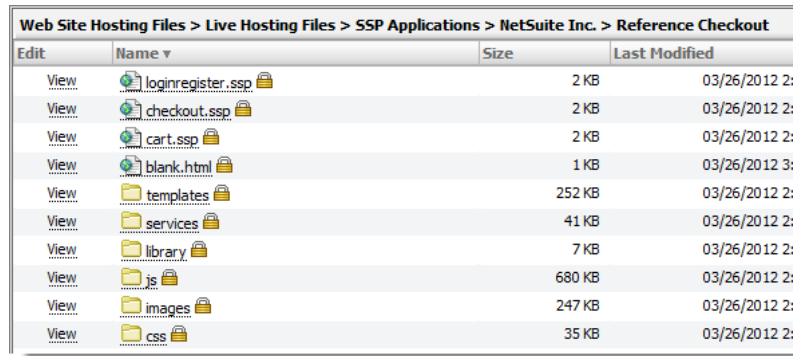
The Custom Checkout SSP application has precedence over the Reference Checkout SSP application, so any touch points that you extend and add to the Custom Checkout SSP application will be served before the Reference Checkout SSP application touch points.

Reference Cart & One Page Checkout BETA Source Files

After you have installed Reference Cart & One Page Checkout BETA, two folders are added to the file cabinet, under the application publisher, NetSuite Inc.

- The Custom Checkout folder is empty. It is available for you to copy files from the Reference Checkout folder and customize them.

- The Reference Checkout folder includes all of the source files and assets associated with the reference solution, organized into subfolders, as shown in the following screenshot.



| Edit | Name ▾ | Size | Last Modified |
|------|-------------------|--------|-----------------|
| View | loginregister.ssp | 2 KB | 03/26/2012 2:27 |
| View | checkout.ssp | 2 KB | 03/26/2012 2:27 |
| View | cart.ssp | 2 KB | 03/26/2012 2:27 |
| View | blank.html | 1 KB | 03/26/2012 3:06 |
| View | templates | 252 KB | 03/26/2012 2:27 |
| View | services | 41 KB | 03/26/2012 2:27 |
| View | library | 7 KB | 03/26/2012 2:27 |
| View | js | 680 KB | 03/26/2012 2:27 |
| View | images | 247 KB | 03/26/2012 2:27 |
| View | css | 35 KB | 03/26/2012 2:27 |

The following types of files are included in the Reference Checkout folder:

- SuiteScript Server Pages (.ssp)
- JavaScript Files
- SuiteScript Files (.ss)
- CSS Files
- Template Files
- Combination Files

Because all of the source files and assets in Reference Cart & One Page Checkout BETA are locked, you must copy files to the Custom Checkout folder to extend the reference implementation. For more information, see [Extending Reference Cart & One Page Checkout BETA](#).

Important: The following sections are intended to provide an introduction to Reference Cart & One Page Checkout BETA files. After reading these sections to get a basic understanding of reference implementation files, you can review the files in your file cabinet for more details.

SuiteScript Server Pages (.ssp)

SSP files, stored in the root folder of the reference implementation, provide the framework for login, registration, the shopping cart, and checkout. These files include headers and footers that render NetSuite site themes, and include content from [Main Templates](#) to generate the login, registration, shopping cart, and checkout pages. Each SSP file is associated with a touch point on the SSP Application record.

Edit SSP files to make changes like the following to the reference implementation:

- Add a new JavaScript library.
- Add your own graphics to the shopping cart, or to the checkout process.
- Add Ignite Commerce merchandising zones.
- Add your own advertisements.
- Build your shopping cart and checkout pages from scratch.

You can use the Commerce API and SuiteScript to extend SSP files. For more information, see the help topic [Commerce API Reference](#).

JavaScript Files

The reference solution includes **site.js** in the root JavaScript files folder. The **site.js** file includes logic for the reference checkout workflow and includes all of the functions associated with the reference shopping cart and checkout. For more information, see [Reference Cart & One Page Checkout BETA Architecture](#).

Modify the **site.js** file to make changes like the following:

- Customize the UI after a specific action occurs.
- Edit the data you show in each step in the checkout process.
- Change elements of checkout steps, such as options and lists.
- Add data validation logic to the shopping cart.
- Add a step to the checkout process.
- Change the data shown in a certain checkout step.

Note that [Reference Cart & One Page Checkout BETA Version 1.1](#) includes an additional JavaScript File, *site_settings.js*. For more information, see [New JavaScript File for Site Settings](#).

SuiteScript Files (.ss)

The reference implementation includes a specialized type of SuiteScript files, .ss files, that implement back-end handling. These files are in the services folder. You can use the SuiteScript API and Commerce API to extend .ss files. For more information, [SuiteScript API Overview](#).

The following services are included in Reference Cart & One Page Checkout BETA:

- **address.ss** - Verifies that the shopper is logged in, prior to starting the checkout process. Automatically selects the most recent address used by the shopper.
- **cart.ss** - Updates the cart to show any changes made by the shopper.
- **country.ss** - Checks the Web Site Setup page to find the list of countries supported by your company. Returns states and provinces for each country.

- **creditcard.ss** - Verifies that the shopper is logged in, prior to entering payment information. When the shopper enters credit card information, this script checks whether this is the only credit card associated with the shopper; if it is, then the credit card is automatically set as the payment method for the current order. If multiple credit cards are associated with a customer record, then the most recently used credit card is set as the payment method.
- **order.ss** - Verifies that the shopper is logged in, prior to submitting the order. Gets valid promotion codes from your NetSuite account. When the shopper enters a promotion code, it is applied to the order. This script also handles the case when a shopper chooses to remove promotion codes from the order. This script also applies gift certificates, and removes them, depending on the shopper's action in the UI.
- **payment.ss** - Verifies that the shopper is logged in. This script retrieves the order total and payment information. If payment information is not retrieved, or if payment information is invalid, an alert is generated for the shopper.
- **shipping.ss** - Verifies that the shopper is logged in, prior to entering shipping information. This script gets all active shipping methods, and then finds and sets the preferred shipping method for the shopper who is logged in.
- **storesettings.ss** - Gets the applicable preference settings from the Web Site Setup page. For information about related limitations, see [Web Site Preferences That Are Not Supported](#).
- **user.ss** - Includes all tasks related to the shopper, for example: logout, isLoggedIn, registerCustomer, registerGuest, and updateProfile.

CSS Files

CSS files affect the look and feel of the login/registration, shopping cart, and checkout pages in your web site. Note that if you use NetSuite site themes on your web site, these are automatically applied to Reference Cart & One Page Checkout BETA.

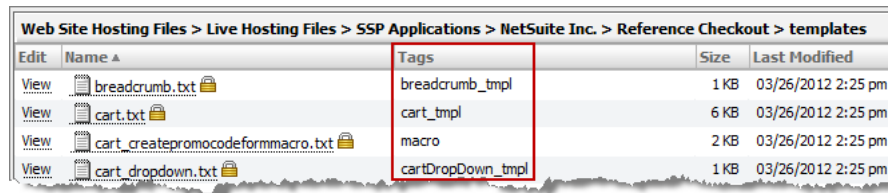
For information about making changes to the look and feel of the reference solution, see [Modifying CSS in Reference Cart & One Page Checkout BETA](#).

Note: Class names in the reference implementation files are mapped to specific templates. Changing or removing these class names can affect the layout of your site. Some class names included in CSS files are necessary for specific logic implemented in site.js. If you choose to edit CSS files, you may need to edit site.js.

Template Files

Reference Cart & One Page Checkout BETA includes a set of default templates that define key user interface (UI) elements of the shopping cart and checkout.

You can add new template files, or modify existing template files, to extend the reference implementation and build a unique shopping cart and checkout experience for your ecommerce customers.



| Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout > templates | | | | |
|---|-----------------------------------|-------------------|------|--------------------|
| Edit | Name ▲ | Tags | Size | Last Modified |
| View | breadcrumb.txt | breadcrumb_tmpl | 1 KB | 03/26/2012 2:25 pm |
| View | cart.txt | cart_tmpl | 6 KB | 03/26/2012 2:25 pm |
| View | cart_createpromocodeformmacro.txt | macro | 2 KB | 03/26/2012 2:25 pm |
| View | cart_dropdown.txt | cartDropDown_tmpl | 1 KB | 03/26/2012 2:25 pm |

Values in the **Tag** fields on template files link templates to the JavaScript (in site.js) running the shopping cart and checkout process. Filenames in the template folder are less important than tags. Dependencies in the JavaScript expect data from the Tag field. For more information about how templates relate to JavaScript in the reference implementation, see [Reference Cart & One Page Checkout BETA Architecture](#).

For summary information about the files in the template folder, see:

- [Main Templates](#)
- [Templates and Macros](#)
- [Working with Template Combination Files in Version 1.0](#)

For further details about individual templates, see [Reference Cart & One Page Checkout BETA Template Reference](#).

Main Templates

All the markup code included in each of the main templates is inserted into a corresponding SSP file using a <div> tag.

For example, to draw the cart page, **cart.ssp** uses the code, templates, and macros included in the cart template file, **cart.txt**. All SSP files included in the reference implementation contain a <div> similar to the code sample below.



```
<div id="main">
</div>
</body>
```

There are three main templates included in the reference implementation. Click one of the following links to see details about each.

- [Cart Templates](#)
- [Shipping & Delivery Step](#)
- [Login and Registration Page Templates](#)

Templates and Macros

The reference implementation differentiates between **templates** and a subtype of templates called **macros**.

Templates

Templates are identified with the **_tmpl** tag. These files are used as containers for other templates and include calls to macros. You can drill down into a template file to view the macros associated with it. For a description of each template, see [Reference Cart & One Page Checkout BETA Template Reference](#).

For example, the code sample below, from **cart.txt**, includes calls to the following macros: **cartPromotions**, which renders a portlet for receiving coupon codes, and **cartEstimateShipping**, which displays shipping information on the shopping cart page.

```
{var total = 0}
<div class="panel-layout">
<div id="container">
  <div class="cart-heading">
    <a class="button-back-to-search"
      onclick="BDK.fire('backToSearchResults')">Continue Shopping</a>
  </div>
  <div class="position">
    <div class="d"> </div>
    <div class="check-out">
      ${cartItemList(obj, true)}
    </div>
  </div>
  <!-- cart actions -->
  <div class="cart-actions">
    ${cartPromotions(obj)}
    ${cartEstimateShipping(obj)}
  </div>
```

Macros

Macros are template files identified with the **macro** tag. These are reusable elements that further define specific views and areas where shoppers interact with the cart and checkout. Macros are identified by the way they are invoked within other template files. Note the code sample below from **cart_promotions.txt** :

```
{macro cartPromotions(cart)}
<div class="cart-promotion">
  <div class="cart-actions-title">Have a Promo Code?</div>
  <div class="cart-actions-text">Enter your promotional code or coupon
    code below. Limit one per customer, please.</div>
```

Note: If you do not want to use a macro, you can edit the template, and use a comment to hide it. You do not need to remove the macro from the template folder.

When you change a macro declaration in a template, you must also make the change in the JavaScript that runs the shopping cart. For example, you can change the macro declaration in a template file, from **cartPromotions** to **cartPromotions1**. Because macros are typically referred to by other templates, you must edit any file that refers to the macro by name, including **site.js**.

Combination Files

Reference Cart & One Page Checkout BETA includes a set of files that are associated with the aggregation, or packaging, of content in JavaScript files and template files.

Read below to learn more:

- [Working with JavaScript Combination Files in Version 1.0](#)
- [Working with Template Combination Files in Version 1.0](#)

Note that JavaScript and template combination files are automatically copied to the Custom Checkout folder when you install Reference Cart & One Page Checkout BETA Version 1.1. For more information, see [Combination Files Copied to the Custom Checkout Folder by Default](#).

Working with JavaScript Combination Files in Version 1.0

When you copy JavaScript files from the reference solution into the Custom Checkout directory, you must also copy the .config file into the associated folder. The .config file triggers production of a time-stamped aggregate file for **site.js**. The .config file includes commands that will also update references to the highest precedence aggregate files in your SSPs or other pages. For more information, see [Setting up the Custom Checkout Application Folder](#).

The combination files below are included in the js directory:

- **js/site-0123abc.js** — This JavaScript file is regenerated with a new time stamp when you edit site.js. When you make any changes to site.js, SSP files in the reference solution automatically point to this generated file.

Note: This file is difficult to debug because when it is regenerated, spaces are removed and variable names are shortened to reduce the file size. For debugging purposes, you can point to site.js (without the time stamp) in your custom SSPs, and then revert the change after your JavaScript has been finalized.

- **js/current.js** — This file points to the most recent time-stamped version of the site.js file in the js directory with the highest precedence. This file is not used or referenced.
- **js/combiner.config** — This file includes configuration settings for the files in the js folder. You can edit this file to set your preference for the quantity of regenerated site.js files you want to keep in the templates directory. If you plan to modify any JavaScript files, you must copy this file into the Custom Checkout folder.

Note the configuration settings included:

- Input-files: *.js — Identifies the JavaScript file extension.
- Combined-file: site.js — Identifies the file that aggregates JavaScript content.

- **Keep-files: 1** — Sets the number of time-stamped site.js files to keep. Note that a new file is generated each time you save a change to site.js.
- **Method: Minify** — Identifies the method that removes spaces and shortens variable names to reduce file size.
- **Referring-files: ../*.ssp** — Identifies all the files that reference the time-stamped site.js file.

Working with Template Combination Files in Version 1.0

The files described below are associated with the aggregation, or packaging, of content in template files.

When you copy any template file from the reference solution into the Custom Checkout directory, you must also copy the .config file to trigger production of a time-stamped aggregate file for the templates files. The .config file will also update references to the highest precedence aggregate files in your SSPs or other pages defined by commands in the .config file. For more information, see [Setting up the Custom Checkout Application Folder](#).

- **templates/combined-0123abc.js** — Every time you edit a template file, this file is regenerated with a new time stamp appended to the file name. This file aggregates the content in all template files and provides one URL that points to all the templates.

Warning: Do not edit content in the combined-0123abc.js file. This file is referenced by other files to display templates properly. Any changes you make to this file could change its functionality.

- **templates/current.js** — This file points to the most recent time-stamped version of the combined.js file in the templates directory with the highest precedence. This file is not used or referenced.
- **templates/templates.config** — This file includes configuration settings for the files in the templates folder. You can edit this file to set your preference for the quantity of combined.js files you want to keep in the templates directory.

Note the configuration settings included:

- **Input-files: *.txt** — Identifies the file extension for template files.
- **Combined-file: combined.js** — Identifies the file that aggregates template files.
- **Keep-files: 3** — Sets the number of time-stamped combined.js files to keep. Note that a time-stamped version of combined.js is regenerated each time you save a change to a template file.
- **Referring-files: ../*.ssp** — Identifies all the files that reference the latest combined-0123abc.js file. Note that if you want to create a new reference to combined-0123abc.js, you must put combined-0.js into your file.

Reference Cart & One Page Checkout BETA Limitations

This section describes current limitations for Reference Cart & One Page Checkout BETA functionality:

- Web Site Preferences That Are Not Supported
- Web Site Tags That Are Not Supported
- Registration Issue when Customer Form has Required Custom Fields

Web Site Preferences That Are Not Supported

The following preferences at Setup > Site Builder > Set Up Web Site are not supported by Reference Cart & One Page Checkout BETA:

- Shopping Cart preferences on Appearance subtab
 - Show Shipping Estimator in Cart
 - Show Extended Cart
 - Display Order of Cart Items
- Shopping subtab preferences
 - Web Store Out Of Stock Items

Note expected outcomes for each of the settings listed below:

- Allow backorders but display out of stock message - The out of stock message is not displayed in the cart or in checkout. Shoppers can add back ordered items to the cart.
- Disallow backorders and display out of stock message - The add to cart links are removed from the store, but there is no messaging anywhere in checkout until order submission, at which time an error is displayed that one or more items is not available in sufficient quantity.
- Hide out of stock items from the store - Behaves as expected. The items are hidden from the store, and the add to cart backend throws an availability error.
- Allow back orders with no out of stock message - Behaves as expected.
- Display Company Field on Registration Page
- Shipping Information is Required
- Ask For Shipping Address First
- Default Web Site Shipping Method

- Default Shipping Country for Checkout
- Display Purchase Order Field on Payment Info Page
- Hide payment page if order total equals zero
- Cart subtab preferences
 - Show indicator for each column
 - Setting for percentage of width for each column

Your choices for these preferences are not supported with the Reference Cart & One Page Checkout BETA bundle installed. Instead of relying on these settings on the Web Site Setup page, you can extend the reference solution. For more information, see [Extending Reference Cart & One Page Checkout BETA](#).

For more information about preferences, see the help topic Web Site Setup Preferences.

Web Site Tags That Are Not Supported

The following tags are not supported in the reference solution:

```
<NLCARTTOTAL>  
<NLCARTITEMCOUNT>  
<NLCARTDISCOUNTEDTOTAL>  
<NLCARTAPPLIEDDISCOUNT>  
<NLCARTAPPLIEDGC>  
<NLCARTSHIPPING>  
<NLCARTSHIPPINGHANDLING>  
<NLCARTTAX>  
<NLCARTGRANDTOTAL>
```

You can edit the site.js file to work around this issue. For details about this file, see [Reference Cart & One Page Checkout BETA Source Files](#).

Registration Issue when Customer Form has Required Custom Fields

The login page template in the reference implementation does not display custom fields from customer records. If your customer form includes any required custom entity fields, shoppers will not be able to register unless you add the required field(s) to the login template. To avoid this issue, you can choose not to deploy the login touch point; or you can customize the login template to add custom fields. See [Adding a Custom Field in a Template](#).

Trouble Shooting Extensions to Reference Checkout

You can use one of the approaches listed below to trouble shoot as you create a custom checkout solution based on the reference implementation:

- [Using the Execution Log](#)

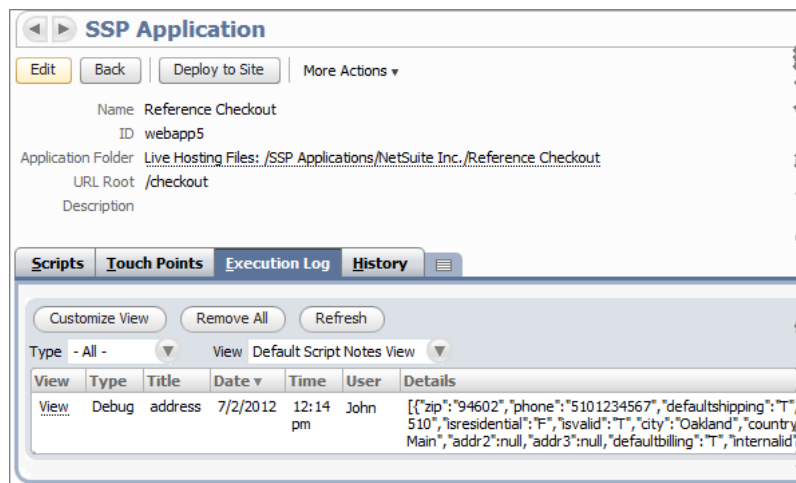
- Reverting Custom Checkout back to Reference Checkout
- Debugging with SSP Files

Using the Execution Log

The Reference Checkout and Custom Checkout SSP Application records include execution logs for tracking some error notifications and form submittal information.

To view the Execution Log:

1. Go to Setup > Site Builder > SSP Applications.
2. Click View next to the SSP Application you are working with.
3. Click the Execution Log subtab.



Reverting Custom Checkout back to Reference Checkout

To trouble shoot errors encountered in the process of extending the reference solution, you can always revert back to the reference implementation.

To revert to Reference Checkout:

1. Go to Setup > Site Builder > SSP Applications.
2. Click View next to Reference Checkout.
3. Click Deploy to Site.
4. Select the Site you want to revert back to Reference Checkout.
5. Select the touch points you want to change.

6. Save.

For more information, see also, [Deploying SSP Application Touch Points](#).

Debugging with SSP Files

You can use SSP files to debug changes you make when extending the reference solution.

1. Create a basic HTML file, and include any APIs you want to investigate, or blocks of custom code you are working on.
2. Save the file using the .ssp file extension.
3. Upload the SSP file you created to the file cabinet in the root Web Site Hosting Files Folder.
4. Use the URL on the file record to view the file in the browser. You can use JSBeautify to read the output.

For more details, see the help topic [Debugging an SSP Application](#).

Extending Reference Cart & One Page Checkout BETA

Review the following to get an understanding of how to extend the reference implementation:

- [Setting up the Custom Checkout Application Folder](#)
- [Creating a Custom Cart and Checkout Solution](#)

Setting up the Custom Checkout Application Folder

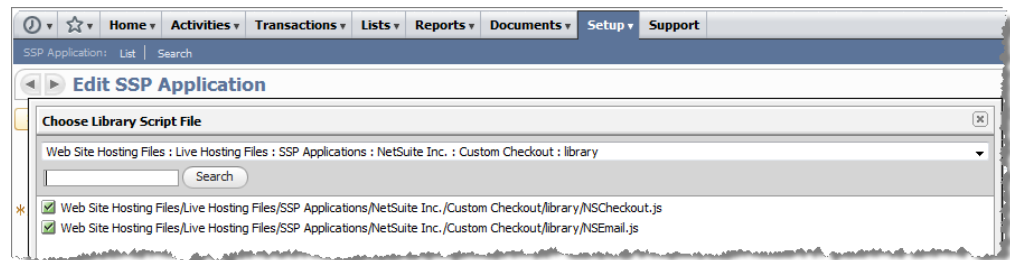
To modify any element of the reference solution, you must first copy the files you want to change from the Reference Checkout application folder into the Custom Checkout application folder. Next, associate the proper library scripts with your custom solution. After the touch points have been initially deployed, they do not need to change. The system of precedence automatically points to the files in the custom checkout folder.

To setup a custom cart and checkout application:

1. Copy files from Reference Checkout to Custom Checkout:
 1. Go to Setup > Site Builder > Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc > Reference Checkout.
 2. Click Copy Files.
 3. Select all files and folders by clicking Mark All.
 4. Select your Custom Checkout SSP application folder in the Copy To list:

Web Site Hosting Files: Live Hosting Files: SSP Applications: NetSuite Inc: Custom Checkout.

5. Click Copy.
2. Attach library scripts to the Custom Checkout application:
 1. Go to Setup > Site Builder > SSP Applications.
 2. Click Edit next to the Custom Checkout application.
 3. Click the Scripts subtab, click the double arrow on the Libraries subtab, and then select List. The Choose Library Script File window displays.

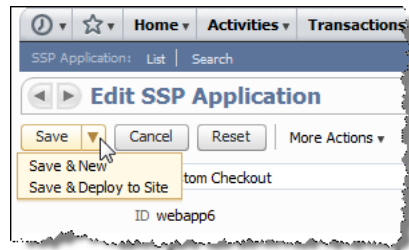


4. In the first dropdown list, select Web Site Hosting Files : Live Hosting Files : SSP Applications : NetSuite Inc. : Custom Checkout : library.
5. Select each file, and then add it to the Libraries subtab:
 - Web Site Hosting Files/Live Hosting Files/SSP Applications/NetSuite Inc./Custom Checkout/library/NSCheckout.js
 - Web Site Hosting Files/Live Hosting Files/SSP Applications/NetSuite Inc./Custom Checkout/library/NSEmail.js
3. Click the Touch Points subtab to select touch points for your Custom Checkout application.

Note: You can choose to deploy all the touch points listed below, which will override the Reference Checkout application. Or you can choose to deploy one or two touch points from your custom application, preserving others from the reference application.

| Name | Entry Page |
|----------------------|--|
| Register | /NetSuite Inc./Custom Checkout/loginregister.ssp |
| Log in | /NetSuite Inc./Custom Checkout/loginregister.ssp |
| Proceed to Checkout: | /NetSuite Inc./Custom Checkout/checkout.ssp |
| View Cart | /NetSuite Inc./Custom Checkout/cart.ssp |

1. Click Save & Deploy to Site.



In the Custom Checkout application folder, you can edit files to fit your specific requirements. The Reference Checkout SSP application and Custom Checkout SSP application share the same URL root (/checkout).

A system of precedence determines which SSP application files are used when multiple applications share the same URL root and file names. When you customize the solution, the files in the Custom Checkout SSP application are served first, because the Custom Checkout SSP application has the highest precedence for the /checkout URL root. If a file is not found in the Custom Checkout application folder, then the file is served from the Reference Checkout application folder.

For more information about extending Reference Cart & One Page Checkout BETA, see [Creating a Custom Cart and Checkout Solution](#).

Reference Checkout FAQ

If I deploy the reference cart and checkout bundle, will it change the look and feel of my web site? Will I need to customize my site again?

No. The reference cart and checkout bundle uses the page header and footer from the Web Site Theme you created in NetSuite.

Why do I have to copy template.config when I want to customize a template?

Template.config is a configuration file that tells the server which files should be combined, which file is the output file, and how the template files are referenced by SSP files. For more information, see [Working with Template Combination Files in Version 1.0](#).

Note that the Custom Checkout application folder is an empty folder that is included in the bundle for you to use when you choose to extend the reference solution. You must copy the template.config file (and other files you use to customize the reference solution), because NetSuite does not add any files into that folder or modify its contents in any way. For more information, see

How do I test the reference cart and checkout bundle with only one site license?

You can use the Multiple Web Sites feature, and test the reference cart and checkout bundle in a web store you designate for testing purposes. Or you can use a NetSuite Sandbox account.

You can also test the reference checkout by installing the bundle, but not deploying the touch points. After you install the bundle, you can navigate to the cart and checkout pages using their URLs (for more information, see [Replacing Cart, Login, and Checkout URLs](#)). In this way, you can see pages from the reference cart and checkout solution in the context of your web store. Functionality should be the same, except that the default cart, login, and checkout links will still go to the old pages. This ensures that shoppers on your web store will not see any of the new pages until the touch points are deployed.

Reference Cart & One Page Checkout BETA Template Reference

This section describes the templates included in the reference checkout implementation.

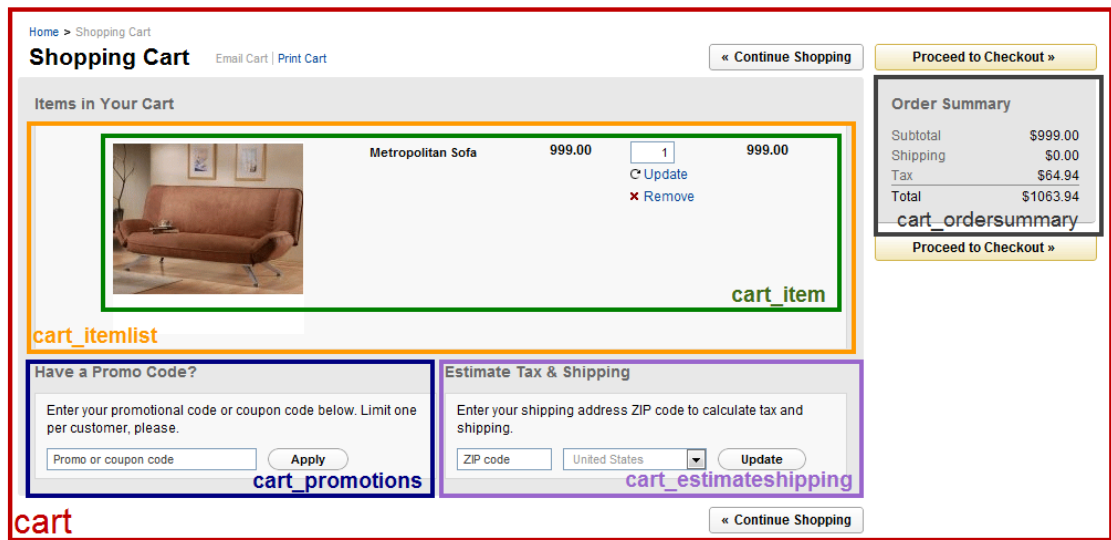
Note that you can only customize copies of the files included in the Reference Checkout application folder. For more information, see [Setting up the Custom Checkout Application Folder](#).

Templates are described below, grouped by the pages in the checkout process to which they apply:

- [Cart Templates](#)
- [Login and Registration Page Templates](#)
- [One Page Checkout Templates](#)

Cart Templates

The templates described below are used to build the reference shopping cart.



| Template | Description |
|----------------------------------|--|
| cart.txt | Calls other templates and macros to display different areas of the shopping cart page. For example, to render the portlet where online shoppers enter coupon codes, cart.txt calls cart_promotions.txt. |
| cart_itemlist.txt | Renders the table that displays the items in the cart. Includes the following columns: Item, Name, Image, Quantity, Options, Subtotal. Note that you must edit this template to change the columns shown on the shopping cart page. |
| cart_item.txt | <p>Renders the item display in the cart:</p> <ul style="list-style-type: none">• Thumbnail image, item name, item price, and quantity in the cart.• UI element for updating and removing items (displayed below the quantity field)• Selected matrix item options for each item in the cart. <p>Also includes logic to calculate the price of the item based on currency, and discounts based on gift certificates or promotion codes.</p> |
| cart_promotions.txt | Renders the portlet where shoppers enter promotion codes. |
| cart_estimateshipping.txt | Renders content in the Estimate Tax & Shipping portlet in the shopping cart. Renders the Zip code input field, country select list, and Update button. |



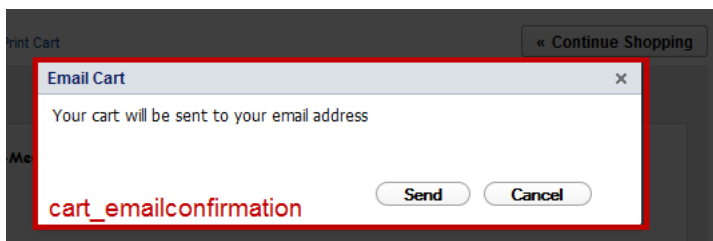
cart_createpromocodeformmacro.txt

Renders an alert for invalid promotion codes and a Remove button. Includes logic for messaging when the promotion code is applied successfully, and when the promotion code is invalid. Includes the input field, apply button, and remove button.

| Order Summary | |
|---------------|----------|
| Subtotal | \$499.00 |
| Shipping | \$0.00 |
| Tax | \$0.00 |
| Total | \$499.00 |

cart_ordersummary.txt

This macro renders the order summary portlet, and is reused on the checkout page. For more information, see [One Page Checkout Templates](#).

**cart_emailconfirmation.txt**

Renders the popup that displays in a lightview when a shopper clicks the link to email the cart.

Login and Registration Page Templates

The templates described below are used to build the login and registration pages.

Home > Sign In

Please Sign In

Returning Customers

Email Address
Password
[Forgot password?](#)

login_returningcustomer

New Customers

Guest Checkout

Create An Account

Name
Email Address
Enter a Password
Re-enter Password
Password Hint

login_createaccountformmacro

login.txt

| Template | Description |
|---|--|
| login.txt | Renders the page where customers can log in, checkout as a guest, or create an account on your web site. Includes the breadcrumb at the top, the structure of the page, the markup code to display the email and password input fields for returning customers, Sign In & Checkout button, and Checkout as Guest button. |
| login_createaccountformmacro.txt | Renders the form where shoppers enter their new account information. Includes input fields for name, email address, password, password verification, and password hint. Also includes the Create Account button. |
| login_returningcustomer.txt | Renders the password hint, if shopper requests it. Renders email input field, password input field, a link to generate a request for password reset, a link back to forgotten password page, and a Sign In & Checkout button. |
| login_createaccountmessage.txt | Displays confirmation message and graphic after a shopper creates a new customer account from your web site. Only available on the confirmation step of guest checkout. |

Note: When shoppers click Checkout as Guest, they are taken directly to the checkout page. Here, they enter shipping and billing information. On the checkout page, shoppers who choose not to create an account are prompted to enter an email address where a receipt for the transaction will be sent.

The screenshot shows a 'New Customers' section with a 'Guest Checkout' button and a 'Create An Account' section. The 'Create An Account' section is highlighted with a red box, and a red arrow points to the label 'login_createaccountsectionmacro'. The form includes fields for Name, Email Address, Enter a Password, Re-enter Password, and Password Hint, followed by a 'Create an Account >' button.

login_createaccountsectionmacro.txt

Renders a message to the web shopper that appears in the New Customer section of the login page under the Create an Account heading. You can use this template to communicate the benefits of creating a customer account on your web site.

The screenshot shows a 'Please Sign In' page with a 'Returning Customers' section. The 'Returning Customers' section is highlighted with a red box, and a red arrow points to the label 'login_passwordhint'. The form includes a message 'Please enter your email to see your password hint.', an 'Email Address' field, and 'Cancel' and 'Submit' buttons.

login_passwordhint.txt

After a customer clicks the Forgot Password link in the Returning Customers section of the reference login page, this template renders the UI where a customer can submit an email address to retrieve the password hint.

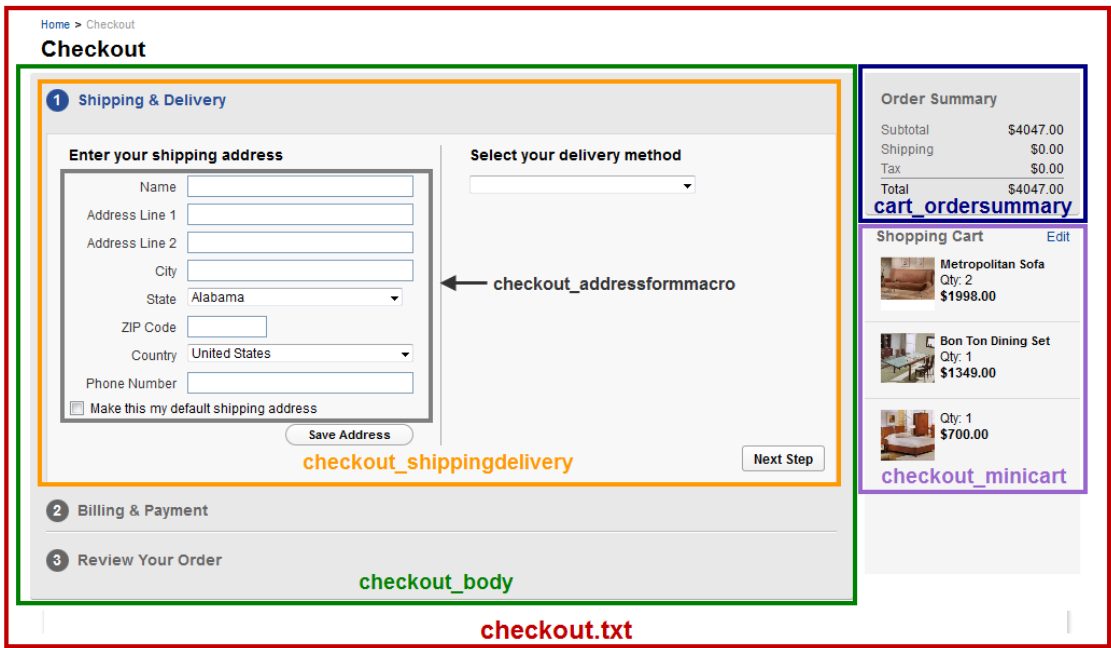
One Page Checkout Templates

The templates described below are used to build one page checkout. The reference implementation consists of three checkout steps presented on one page. The topics below include information about the templates used to create each step:

- Shipping & Delivery Step
- Billing & Payment Step
- Checkout Review Step

Shipping & Delivery Step

In this step of the checkout process customers enter their shipping address and select a shipping method. Customers that already exist in the system have the option of selecting an address they have entered previously, or entering a new shipping address. Customers that choose not to create an account on your site are required to enter an email address where a receipt for the web order is sent.



| Template | Description |
|--------------------------------------|--|
| checkout_shippingdelivery.txt | Renders two target areas: one for entering address information, and one for communicating with address services. Also renders the Next Step button in the first step of the reference implementation checkout. |
| checkout_addressformmacro.txt | Renders the input fields on the address form. Reused anywhere the address input form is displayed on your site. |
| checkout_minicart.txt | Renders a view of the shopping cart on the checkout page. |

Select your shipping address

☒ **S. Wolfe**
123 Main Street
Oakland, CA 94602 US
510-555-1212
[Edit](#) | [Delete](#)

☐ **S. Wolfe**
777 Campus Drive
San Mateo, CA 94403 US
650-5551212
[Edit](#) | [Delete](#)

[Add a New Address](#)

| Template | Description |
|-------------------------------------|--|
| checkout_addresses.txt | Renders the address data associated with checkout. Communicates with JavaScript to get information from related services, and calls macros for receiving and displaying address information. You can reuse this template anywhere in the checkout workflow where customers are required to select or edit an address they entered at a previous time. |
| checkout_addresslistitem.txt | Renders addresses from existing customer records. Displays all address fields as well as the radio button and links for Edit and Delete. |
| checkout_addressitem.txt | Renders name and address information including city, state, and country on the address summary. Displayed after a shopper has progressed to the next step. |

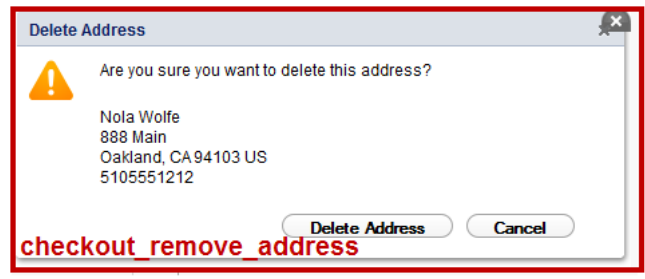
checkout_addressform.txt

Renders the address input form in a popup on the checkout page. Uses checkout_addressformmacro.txt, which contains all the fields.

checkout_guestemailform.txt

Renders a portlet for guest checkout users. Includes content for the portlet title, email address input field. Logic for displaying the form is in site.js.

| Template | Description |
|--|---|
| checkout_addressstatesectionmacro.txt | Renders the states or provinces in the State select list on the address form. |
| checkout_addressformmacro.txt | Renders the input fields on the address form. Reused anywhere the address input form is displayed on your site. |



checkout_remove_address.txt

Displays popup alert when customer tries to delete an address. Includes default text for the alert message and Cancel button.

| Template | Description |
|--|--|
| checkout_shippingmethodcarriers.txt | Renders the list of shipping methods available on the web site, and shipping rates. Also includes the header text for the portlet. |
| checkout_shippingmethoditem.txt | Displays the shipping method item name and rate. |
| checkout_shippingrates.txt | Displays shipping methods and their rates. Draws the radio buttons. |

Checkout

1 Shipping & Delivery

Edit

Shipping Address

S. Wolfe
123 Main
Oakland, CA 94602 US
5105551212

Delivery Method

FedEx - \$34.93

checkout_shippingmethodsummary

checkout_shippingdeliverysummary

↑
checkout_addresssummary

| Template | Description |
|--------------------------------------|--|
| checkout_shippingmethodsummary.txt | Shows the selected shipping method in view-only format. |
| checkout_shippingdeliverysummary.txt | Shows the customer's address and shipping information in view-only format. Includes the Edit button. |
| checkout_addresssummary.txt | Shows the customer's address in view-only format. |

Billing & Payment Step

In this step of the checkout process, customers select a billing address and enter credit card information. Customers have the option to edit or delete existing billing address information. Customers can also enter a gift certificate and promotion codes in this step.

2 Billing & Payment

Select your billing address

sabina 510

123 Main

Oakland, CA 94602 US

5101234567

Edit | Delete

Add a New Address

checkout_creategiftcardformmacro →

Select a Credit Card

VISA

VISA *****5100

6/1/2023

Edit | Delete

Add a New Credit Card

Use a Gift Card or Gift Certificate

Enter your promotional code or coupon code below. Limit one per customer, please.

Gift Card or Gift Certificate

Apply

Have a Promo Code?

Enter your promotional code or coupon code below. Limit one per customer, please.

Promo or coupon code

Apply

Next Step

checkout_billingPayment

| Template | Description |
|--------------------------------------|---|
| checkout_billingPayment.txt | Renders the billing and payment step in checkout. Calls various subtemplates, including macros, for displaying address and credit card information. Includes a <div> for each portlet where the customer inputs data, such as the billing address field, credit card information, gift certificate, and promotion code. Also includes a Next Step button. |
| checkout_creategiftcardformmacro.txt | Renders the input fields where a customer enters gift certificate information. Includes logic for handling multiple gift certificates and invalid codes. Also includes Apply and Remove All buttons. |
| checkout_giftpromo.txt | Displays the promotion code that was entered. |

Have a Promo Code?

Your promo code was applied successfully. Limit one code per order. Remove this discount to apply a different code.

✔

Promo code SPRINGCAT applied

Remove

checkout_promocode ↑

Reference Cart and One Page Checkout BETA

N

checkout_promocode.txt

Displays the promotion code that was entered.

Enter a Credit Card

Name on Card

Card Type

Card Number

Expiration

CCV

checkout_creditcardformmacro

checkout_creditcardform

Save Credit Card Cancel

| Template | Description |
|----------------------------------|--|
| checkout_creditcardform.txt | Generates a popup where customers enter credit card information. Includes Save and Cancel buttons. |
| checkout_creditcardformmacro.txt | Renders the input fields where customers enter credit card information. |

Enter a Credit Card

Name on Card

Card Type

Card Number

Expiration

CCV

checkout_ccvmacro

For security purposes, we require the Card Security Code on your credit card. Enter the 3-digit number that follows your account number on the back of your card

112 456 1120

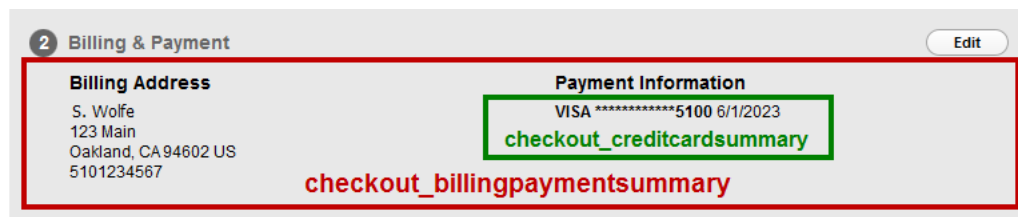
Save Credit Card Cancel

checkout_ccvmacro.txt

Renders a tool tip for entering CCV code for payment information. Includes message with instructions.



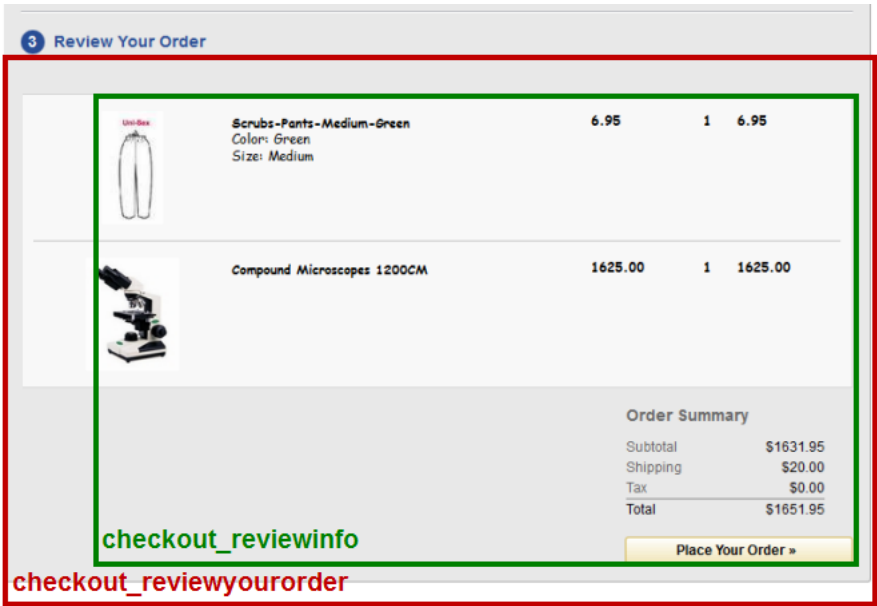
| Template | Description |
|--|---|
| checkout_creditcards.txt | Renders the portlet in the billing and payment step, where customers enter credit card information. Includes logic for displaying default credit card information for existing customers, and displaying input fields for new customers. Also includes links for adding a new credit card or saving credit card information that was entered. |
| checkout_creditcarditem.txt | Renders credit card information in the Billing & Payment step of checkout. Includes the expiration date for the selected credit card. |
| checkout_creditcardlistitem.txt | Renders the radio button, the credit card logo for the selected card, and links for Edit and Delete. |



| Template | Description |
|--|--|
| checkout_billingpaymentssummary.txt | Shows the billing address and selected credit card. |
| checkout_creditcardsummary.txt | Shows the customer's credit card information in view-only format. Note that the credit card number is obfuscated for security. |

Checkout Review Step

As customers move through each step of the checkout process, their information is displayed in view-only format for review. The checkout review step allows the customer to review the complete web order before submitting the order.



| Template | Description |
|------------------------------|---|
| checkout_reviewinfo.txt | Displays all the items in the shopping cart. Includes the order summary portlet and Place Order button. |
| checkout_reviewyourorder.txt | Provides the frame of the page. |

Home > Checkout

Checkout

Order Confirmation

Thank you for shopping with us! We've received your order and will process it promptly. Please print this page for your records.

Order Number: 364424-11

Order Summary

| | |
|--------------|-----------------|
| Subtotal | \$499.00 |
| Shipping | \$10.00 |
| Tax | \$0.00 |
| Total | \$509.00 |

Create an Account

Name
S. Wolfe

Email Address
swoffe@email.com

Enter a Password

Re-enter Password

Password Hint

[Create an Account >](#)

Shipping & Payment Information

Shipping Address
S. Wolfe
123 Main
Oakland, CA 94602 US
5105551212

Delivery Method
Airborne - \$10.00

Payment Information
S. Wolfe VISA **** *\$100 1/31/2014

Billing Address
S. Wolfe
123 Main
Oakland, CA 94602 US
5105551212

checkout_confirmation_info

| | | | |
|------|--------|---|--------|
| iPad | 499.00 | 1 | 499.00 |
|------|--------|---|--------|

Order Summary

| | |
|--------------|-----------------|
| Subtotal | \$499.00 |
| Shipping | \$10.00 |
| Tax | \$0.00 |
| Total | \$509.00 |

[Continue Shopping >](#)

checkout_confirmation

| Template | Description |
|---------------------------------------|---|
| checkout_confirmation.txt | Provides the frame of the confirmation page. |
| checkout_confirmation_info.txt | Displays the order number, shipping and delivery information, payment information, billing address, and items in the order. |

Creating a Custom Cart and Checkout Solution

The topics listed below are intended to help you learn about customizing Reference Cart & One Page Checkout BETA. You can also use the Developer Cheat Sheet, included in the Reference Checkout application folder, for detailed information about the code structure used in the reference solution.

To customize files in the reference solution, you must copy them into the Custom Checkout application folder. For more information, see [Setting up the Custom Checkout Application Folder](#).

Important: The following procedures are intended to provide a starting point for your own customizations. They are not an exact blueprint; you will need to adapt steps to fit the requirements of your site.

- [Modifying CSS in Reference Cart & One Page Checkout BETA](#)
- [Adding a Custom Field in a Template](#)
- [Changing the Order of Checkout Steps](#)
- [Changing the Layout of the Checkout Page](#)
- [Displaying Related Items in the Reference Shopping Cart](#)
- [Using the Validation Library](#)
- [Splitting One Checkout Step into Two Steps](#)
- [Adding Terms and Conditions to Reference Cart & One Page Checkout BETA](#)

You must test any changes thoroughly before releasing them to your production web site. For a testing tip, see [Debugging with SSP Files](#).

Modifying CSS in Reference Cart & One Page Checkout BETA

You can modify the style sheet included in the reference solution to make the look and feel of the shopping cart and checkout pages consistent with the rest of your web site.

To edit the style sheet in a NetSuite Account:

1. Put a copy of **style.css** in the Custom Checkout application folder. Make your changes in the Custom Checkout Application folder.
 1. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout > css.
 2. Click Copy Files.
 3. Check the box next to **style.css**.
 4. In the Copy To field, select Web Site Hosting Files : Live Hosting Files : SSP Applications : NetSuite Inc. : Custom Checkout : css.
 5. Click Copy.

2. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > css.
3. Click Edit next to **style.css**.
4. Click Edit on the file record to open the file content screen, and then make your changes in the Content field.
5. Click Save.

Your changes to the CSS file will affect the login, shopping cart, and checkout pages. After saving your changes to the CSS file, refresh one of these pages in the web site to see your changes.

Adding a Custom Field in a Template

You can edit any template to display custom fields.

If you use a custom field on customer records, you must add this field to the login template to use the reference implementation. Go to Customization > Lists, Records, & Fields > Entity Fields, and copy the ID for the field you plan to add to the template.

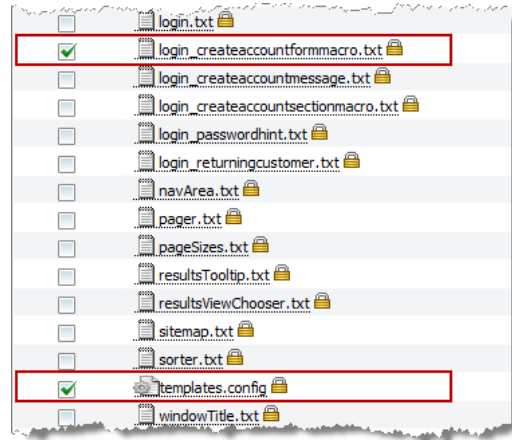
To add a custom field to a reference checkout template:

1. Create a custom field, and remember the ID. You will need to insert the ID in the template.

Go to Customization > Lists, Records, & Fields > [Custom Field] > New, where [Custom Field] is the desired field type.

2. Put a copy of the template file you want to modify and the template.config file in the Custom Checkout application folder. You will modify the template (.txt) file in the Custom Checkout application folder.
 1. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout > templates.
 2. Click Copy Files.
 3. Check the box next to the template file you want to modify, and the template.config file. For example: To add a custom field in the registration form on the login page, copy the **login_createaccountformmacro.txt template** and **templates.config**.

Note: If you use Reference Cart & One Page Checkout BETA Version 1.1, copying the template.config file is not necessary.



4. In the Copy To field, select Web Site Hosting Files : Live Hosting Files : SSP Applications : NetSuite Inc. : Custom Checkout : templates.
5. Click Copy.
3. Edit the template file in the Custom Checkout folder to add the custom field.

For example: To add a custom field in the registration form on the login page, add a new <div> class for the new field:

```
<div class="f-row">
  <label for="">How did you find out about us?</label>
  <div class="f-input"><input type="text" id="custentity1" name="" value="" class="required"/></div>
</div>
```

4. Click Save in the file editor.
5. Click Save on the file record.
6. Refresh the login page on your web site to verify your change.

To make sure that shoppers on your site cannot proceed to checkout without entering information in the custom field, include the **class="required"** parameter in the <div> tag. Also note that if you want to include a custom field with list options, you can write your own code in the template to create the dropdown.

Changing the Order of Checkout Steps

Reference Cart and One Page checkout includes a three-step checkout process displayed on one page. You can modify the JavaScript included in the reference implementation to

change the order in which the steps are presented on the checkout page. For example, you can customize the reference implementation so that the first step is where customers enter billing and payment information.

1. Put a copy of **site.js** and **combiner.config** in the Custom Checkout application folder. Make your changes in the Custom Checkout Application folder.
 - a. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout > js.
 - b. Click Copy Files.
 - c. Check the box next to **site.js**, and the box next to **combiner.config**.

Note: If you use Reference Cart & One Page Checkout BETA Version 1.1, copying the **combiner.config** file is not necessary

- d. In the Copy To field, select Web Site Hosting Files : Live Hosting Files : SSP Applications : NetSuite Inc. : Custom Checkout : js.
 - e. Click Copy.
2. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc > Custom Checkout > js.
3. Click Edit next to **site.js**.
4. Click Edit on the file record to open the file in edit mode.
5. Find the variable, `_checkoutController` in **site.js**.
6. Cut and paste IDs and templates to organize them as you wish. In the code sample below, the billing step was changed to be the first step of checkout.

```
* One page reference checkout
*
* Checkout steps are defined here
*/
var _checkoutController = (function () {
  var _templates, _ctx = null;
  // order of steps here dictates the order of steps of the checkout
  var _steps = [
    {
      id: "billing",
      template: "checkout_tmpl_billing"
    },
    {
      id: "shipping",
      template: "checkout_tmpl_shipping"
    },
    {
      id: "review",
      template: "checkout_tmpl_review"
    },
    {
      id: "confirmation",
      template: "checkout_tmpl_confirmation"
    }
  ];
})();
```

7. Save your changes to the file.

Changing the Layout of the Checkout Page

You can rearrange the views displayed on the checkout page by making changes to the template file. The example below describes how to move the order summary portlet from the right side of the cart to the left side.

1. Copy the cart template and the templates.config file in the Custom Checkout application folder. You will modify the template file in the Custom Checkout application folder.
 - a. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout > templates.
 - b. Click Copy Files.
 - c. Check the box next to **cart.txt**, and **templates.config**.

Note: If you use Reference Cart & One Page Checkout BETA Version 1.1, copying the template.config file is not necessary

- d. In the Copy To field, select Web Site Hosting Files : Live Hosting Files : SSP Applications : NetSuite Inc. : Custom Checkout : templates.
 - e. Click Copy.
2. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc > Custom Checkout > templates.
3. Click Edit next to cart.txt.
4. Click Edit on the file record to open the file in edit mode.
5. Use the code samples below as guidelines for changing the template:

- Move the content area to the right.

```
<!-- main-container -->
<div class="main-container clearfix">
  <!-- BEGIN content-part -->
  <div class="content-part fl-r clearfix">
```

- Move the order summary portlet to the left.

```
<!-- BEGIN aside -->
<div class="aside fl-l">
  <!-- btn place -->
  <div class="btn-place clearfix">
```

6. Save your changes.

Your changes to the cart.txt file will only affect the shopping cart page. After saving your changes, refresh the cart page to see your changes.

Using the Validation Library

Reference cart and one page checkout uses a third party library for validation rules. You can use this library when you customize the reference solution.

To access the validation library, go to `/js/libs/validation.js`.

For more information about the validation rules included, visit the web site, [really-easy-field-validation](#). On the site, read the *Options* section for the CSS classes you can use on entry forms such as the login page where customers enter email and password information.

Read the CSS Hooks section if you want to extend the rules included in the reference solution.

For example, you can apply a custom validation rule to the state field on the address form, indicating that your web store only ships to a certain state or province. To do this, add the validation rule to `site.js`, and then modify the template that draws the address form.

1. Put copies of the files you will modify in the Custom Checkout application folder.
 - a. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout.
 - b. Copy the files listed below from the Reference Checkout application folder to their corresponding folders in the Custom Checkout application folder:
 - `/js/site.js`
 - `/js/combiner.config`
 - `/templates/checkout_addressstatesectionmacro.txt`
 - `/templates/templates.config`

Note: If you use Reference Cart & One Page Checkout BETA Version 1.1, it is not necessary to copy the `combiner.config` and the `template.config` files.

2. Create a new validation rule.
 - a. Click Edit next to **site.js**, and click the Edit link on the file record. Go to the end of the file.
 - b. Add new code. Use the sample below as a guide:

```
Validation.add("ca-only", "Only ship to California", {  
  is : "CA"  
});
```

- c. Save.
3. Edit the template, **checkout_addressstatesectionmacro.txt** to use this rule.

- a. Add the new rule to the login-name field.

```
<select name="state" id="state" class="validate-select ca-only" title="State">
```

- b. Save.

Displaying Related Items in the Reference Shopping Cart

You can extend the reference shopping cart to display related items associated with an item selected in the cart. To do this, you need to modify the services for retrieving the items in the cart, create a new macro for displaying related items, and edit the cart template to include the macro for displaying related items.

To display related items in the reference shopping cart:

1. Set up related items in NetSuite. For more information about creating related items, see the help topic Related Items.
 2. Put copies of the files you will modify in the Custom Checkout application folder.
 1. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout.
 2. Copy the files listed below from the Reference Checkout application folder to their corresponding folders in the Custom Checkout application folder.
 - /services/cart.ss
 - /templates/cart.txt
 - /templates/templates.config
- Note:** If you use Reference Cart & One Page Checkout BETA Version 1.1, copying the template.config file is not necessary
3. Modify the cart service to return related items.
 1. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > services.
 2. Click Edit next to cart.ss, and click the Edit link in the file record.
 3. Extend the service provided in the reference solution to return related items on the shopping cart page.
 - Modify the variable that returns data for use in the shopping cart, **retobj**, to include related items.

```
var retobj = {"header": {"status": {"code": "SUCCESS", "message": "success"}},
  "result": {"totalfound": 0,
    "items": [],
    "relateditems": [], //Add related items member.
    "promocode": {},
    "summary": {}}};
```

- Add a block of custom code to return all related items for each item in the cart.

```
}
retobj.result.items[j] = item;

//Getting the related items for each item in my cart.
var itemSource = {"internalid": items[i].internalid, "itemtype": "invtpart"};
var related = nlapiGetWebContainer().getShoppingSession().getRelatedItems(itemSource);
if (related) {
  for(var j=0; j<related.length; j++)
  {
    retobj.result.relateditems[j] = { 'id': related[j].internalid, 'name': related[j].storedisplayname2,
      'price': related[j].onlinecustomerprice, 'image': related[j].storedisplayimage }
  }
}
// END Getting Related Items
```

4. Modify the template that draws the shopping cart to display related items.

You need to create a new macro based on **cart_item.txt**, and edit **cart.txt** to include the new macro you created.

1. Create a new file on your computer. You can use the sample code below, as a guideline. Name the file *cartRelatedItem.txt*.

```
{macro cartRelatedItem(item)}
  <tr>
    <td class="it-pict">

      {var imgethumb = item.image}
      

    </td>
    <td class="it-title">
      <h3><a>{item.name}</a></h3>
    </td>
    <td class="it-price">
      <strong class="price">{item.price}</strong>
    </td>
  </tr>
{/macro}
```

2. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > templates.
3. Upload *cartRelatedItem.txt*. (This is the file you created in Step 4a.)
4. Click Edit next to **cart.txt**, and then click the Edit link in the file record.

5. Add a new section to the cart template to display related items. You can choose where on the cart page you want to display related items, and you can add HTML markup to format the look and feel. The code sample below shows the basic syntax:

```
<!-- Related items section heading -->
<div class="heading clearfix">
  <table>
    {for related in obj.cart.relateditems}
      ${cartRelatedItem(related)}
    {/for}
  </table>
</div>
<!-- END heading -->
```

Put an item that has related items in the shopping cart to see your changes. The list of related items should display under the cart.

Splitting One Checkout Step into Two Steps

In the reference implementation for one page checkout, customers enter two types of data in each step. For example, shipping and delivery, billing and payment. You can split one checkout step, such as billing and payment, into two steps, so that customers enter credit card information in a separate step.

To extend the reference implementation in this way, you need to create new templates for the additional step and its summary, and modify the JavaScript in site.js to recognize the new templates and validate the credit card information being entered.

The instructions below include code samples for adding a payment step to the checkout process, so the web customer sees four steps in one page checkout, rather than the three steps included in the reference implementation. The fourth step is where the customer can enter credit card information.

Step One: Prepare your files

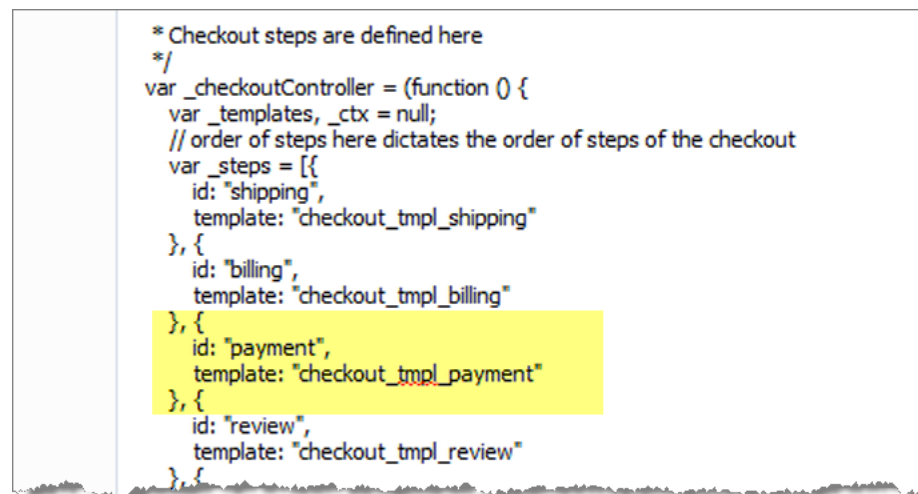
1. Put copies of the files you will modify in the Custom Checkout application folder.
 1. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout.
 2. Copy the files listed below from the Reference Checkout application folder to their corresponding folders in the Custom Checkout application folder.
 - /js/site.js
 - /js/combiner.config
 - /templates/checkout_billingpayment.txt
 - /templates/checkout_billingpaymentsummary.txt

- /templates/templates.config

Note: If you use Reference Cart & One Page Checkout BETA Version 1.1, it is not necessary to copy the combiner.config and the template.config files.

2. Create a new step in the checkout process. Modify **site.js** in the Custom Checkout application folder, by reusing a definition for a step that already exists. You need to create an **ID** and a **tag** to identify a new template for the new step.
 1. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > js.
 2. Click Edit next to site.js, and click the Edit link on the file record.
 3. Find the variable **_checkoutController**. This is the area of code where you need to insert a new step.

For the example described here, the ID for the new step is **payment**, and the tag for the template is **checkout_tmpl_payment**.



```
* Checkout steps are defined here
*/
var _checkoutController = (function () {
  var _templates, _ctx = null;
  // order of steps here dictates the order of steps of the checkout
  var _steps = [{
    id: "shipping",
    template: "checkout_tmpl_shipping"
  }, {
    id: "billing",
    template: "checkout_tmpl_billing"
  }, {
    id: "payment",
    template: "checkout_tmpl_payment"
  }, {
    id: "review",
    template: "checkout_tmpl_review"
  }, {
  }];
}
```

4. Save your changes to site.js.
3. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > templates.

Step Two: Creating a new template

1. Create a new template to show an additional step on the checkout page. Use the steps below to create a new template file for the payment step:
 1. Click Download next to *checkout_billingpayment.txt*.

2. Copy the contents of *checkout_billingpayment.txt* into a new file on your computer, and give the file a new name, for example, *checkout_payment.txt*.
3. Upload the file to the templates folder of the Custom Checkout application.
4. Click edit next to the new template file you uploaded.
5. Enter a value in the Tag field. Use a naming convention similar to what exists in the reference implementation, for example: **checkout_tmpl_payment**. Note that this tag associates the template with the JavaScript in *site.js*.
6. Click Save on the file record.

Note: Because the additional step is included in *site.js*, and a new template has been created to show the new step, at this point you can verify your changes by visiting the checkout page and clicking ctrl + refresh. Four steps display in the checkout process. The new step is empty. Follow the instructions below to add content.

2. Edit the contents of the new *checkout_payment.txt* template. Remove the information that is no longer relevant for the new step you are creating.
 1. The new payment step should only include fields and services related to credit card input. Remove the content listed below:
 - `<div id="addresses"></div>`
 - `<div id="guestCheckoutEmailSection"></div>`
 - `<div id="giftPromoSection"></div>`
 2. Save your changes to the file.
3. Create a new template file for the view-only summary information displayed when the customer clicks Next Step.
 1. Click Download next to *checkout_billingpaymentssummary.txt*.
 2. Copy the contents of *checkout_billingpaymentssummary.txt* into a new file on your computer, and give the file a new name, for example, *checkout_paymentssummary.txt*.
 3. Click edit next to the file you uploaded.
 4. Enter a value in the Tag field, for example, **checkout_tmpl_paymentssummary**.
 5. Click Save.
4. Edit the original *checkout_billingpaymentssummary.txt* file to update the content.
 1. Remove the div that includes payment information and credit card information.

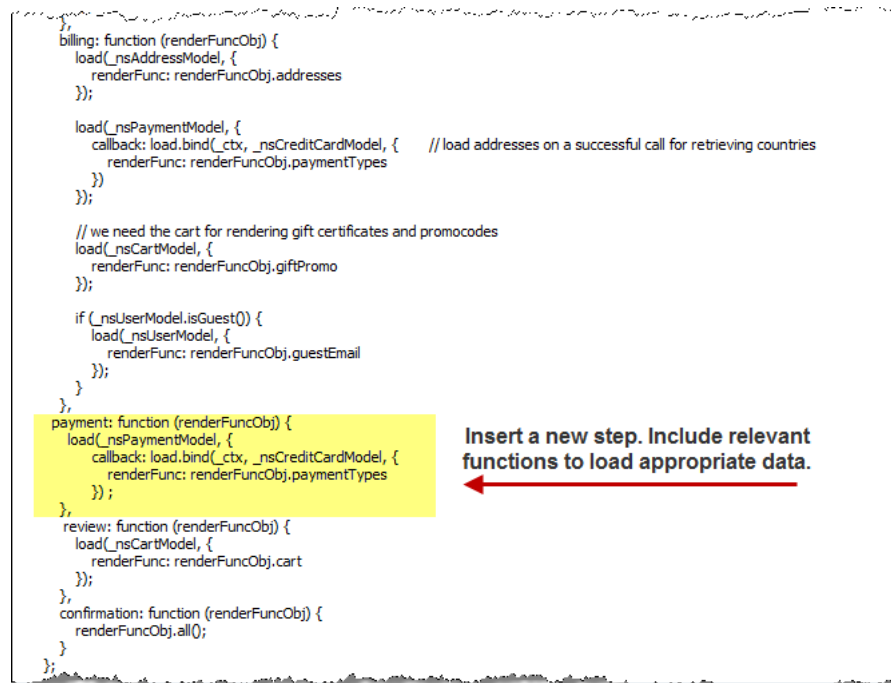
```
<div class="col fl-r">
  <h3>Payment Information</h3>
  <div id="creditCardSummary"></div>
</div>
```

2. Save the file.

Use the steps below as guidelines for changing functions in site.js. Typically, you need to make changes to two types of functions in the JavaScript: functions for loading data, and functions for rendering data.

Step Three: Modifying functions for loading data

- Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > js.
 1. Click Edit next to site.js, and then click the Edit link on the file record.
 2. Find the variable **_dataLoadingFuncs**.
 3. Copy and paste the review function, and rename it **payment**.



```
    billing: function (renderFuncObj) {
      load(_nsAddressModel, {
        renderFunc: renderFuncObj.addresses
      });

      load(_nsPaymentModel, {
        callback: load.bind(_ctx, _nsCreditCardModel, { //load addresses on a successful call for retrieving countries
          renderFunc: renderFuncObj.paymentTypes
        })
      });

      // we need the cart for rendering gift certificates and promocode
      load(_nsCartModel, {
        renderFunc: renderFuncObj.giftPromo
      });

      if (_nsUserModel.isGuest()) {
        load(_nsUserModel, {
          renderFunc: renderFuncObj.guestEmail
        });
      }
    },
    payment: function (renderFuncObj) {
      load(_nsPaymentModel, {
        callback: load.bind(_ctx, _nsCreditCardModel, {
          renderFunc: renderFuncObj.paymentTypes
        })
      });
    },
    review: function (renderFuncObj) {
      load(_nsCartModel, {
        renderFunc: renderFuncObj.cart
      });
    },
    confirmation: function (renderFuncObj) {
      renderFuncObj.all();
    }
  }
};
```

4. Delete **load_nsCartModel** from the new **payment** step.
5. Cut **load_nsPaymentModel** from the billing step, and paste it in your new payment step.

Step Four: Modifying functions for rendering data

1. Create a new function for rendering. Find **_renderFuncs** in site.js.

```

    $(billingAddressSummary).update(_templates["addressSummaryTpl"].render({
      selected: _nsAddressModel.getSelected("billing")
    }));
  },
  paymentTypes: function () {
    $(creditCardSummary).update(_templates["creditCardsSummaryTpl"].render({
      selected: _nsCreditCardModel.find(_nsPaymentModel.getSelected())
    }));
  }
},
payment: {
  step: {
    // init: function () {
    //   $(_konst.sideMiniCart).hide();
    // },
    // cart: function () {
    //   $("reviewInfo").update(_templates["reviewInfoTpl"].render({cart: _nsCartModel.cart, checkout: true}));
    //   paymentTypes: function () {
    //     _templates.creditCardsTpl.render({
    //       hasCreditCredits: !_nsCreditCardModel.isEmpty(),
    //       defaultCreditCard: _nsCreditCardModel.getDefault(),
    //       noneDefaults: _nsCreditCardModel.getNoneDefaults(),
    //       paymentTypes: _nsPaymentModel.paymentTypes
    //     });
    //   }
    // }
  },
  review: {
    step: {
      init: function () {
        $(_konst.sideMiniCart).hide();
      },
      cart: function () {
        $("reviewInfo").update(_templates["reviewInfoTpl"].render({cart: _nsCartModel.cart, checkout: true}));
      }
    }
  }
}

```

- a. Copy and paste the review step to create a new step.
- b. Remove code that no longer applies.
- c. Add relevant functions to render appropriate data.

1. Copy and paste the block of code in the review step, and then rename it **payment**.
 2. Delete cart and minicart.
 3. Cut the payment types function out of billing, and then paste it into the new payment block.
2. Finalize the payment summary step.

```
summary: {
  addresses: function () {
    $("billingAddressSummary").update(_templates["addressSummaryTpl"].render({
      selected: _nsAddressModel.getSelected("billing")
    }));
  },
  paymentTypes: function () {
    $("creditCardSummary").update(_templates["creditCardsSummaryTpl"].render({
      selected: _nsCreditCardModel.find(_nsPaymentModel.getSelected())
    }));
  }
},
payment: {
  step: {
    paymentTypes: function () {
      _templates.creditCardsTpl.render({
        hasCreditCredits: !_nsCreditCardModel.isEmpty(),
        defaultCreditCard: _nsCreditCardModel.getDefault(),
        noneDefaults: _nsCreditCardModel.getNoneDefaults(),
        paymentTypes: _nsPaymentModel.paymentTypes
      })
    }
  },
  summary: {
    paymentTypes: function () {
      $("creditCardSummary").update(_templates["creditCardsSummaryTpl"].render({
        selected: _nsCreditCardModel.find(_nsPaymentModel.getSelected())
      }));
    }
  }
},
review: {
  step: {
    init: function () {
      $(_konst.sideMiniCart).hide();
    },
    cart: function () {
      $("reviewInfo").update(_templates["reviewInfoTpl"].render({cart: _nsCartModel.cart, checkout: true}));
    }
  }
}
```

1. Cut the payment types function from the original billing summary block and paste it into the new payment summary block
2. Click Save to save your changes to site.js.

Step Five: Updating events in site.js

- Create a validation event to fire after leaving the payment step. The code sample below shows the code block. The steps below explain each modification made to the code.

```

/*
 * Payment Step
 */
function savePaymentStepBridge(evt) {
    try {
        var isValid = true;
        //var bWaitForGuestEmailBeforeProceed = false;
        var errorMessage = "Please ";
        //if (!_nsAddressModel.selectedBillingAddressId) {
        //    errorMessage += "select a billing address ";
        //    isValid = false;
        //}

        if (!_nsPaymentModel.getSelected() || (_nsPaymentModel.isCCVRequired() &&
            !_nsPaymentModel.ccvs[_nsPaymentModel.getSelected()])) {
            errorMessage += "_nsCreditCardModel.isEmpty() ? "<br>add a credit card" : "<br>select a credit card ";
            isValid = false;
        }

        //if ("F" === _nsCartModel.cart.promocode.isValid) {
        //    errorMessage += "<br>remove the invalid coupon code";
        //    isValid = false;
        //}

        //if (_nsUserModel.isGuest()) {
        //    if (updateGuestEmail()) {
        //        bWaitForGuestEmailBeforeProceed = true;
        //    }
        //    else {
        //        errorMessage += "<br>fill in your email address";
        //        isValid = false;
        //    }
        //}

        if (isValid) {
            // if (!bWaitForGuestEmailBeforeProceed) {
            //     proceedToNextStep();
            // }
        }
        else {
            toggleValidationTip(true, errorMessage);
        }
    }
    catch (e) {
        handleException(e);
    }
}

/*
 * Goto the next step in checkout.

```

1. In site.js, find StepBridge(evt).
2. Copy and paste the billing step, and rename it to Payment step.
3. Modify the custom save payment event and bridge event.
4. In the billing step, copy and paste the stepBridge function, and rename it to paymentBridge.
5. Save your changes

Step Six: Finalizing the new template

- Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > templates.
 1. Edit *checkout_payment.txt*.
 2. Add a <div> tag for the payment summary step.

3. Save.

Now when your customers go through the checkout process, they will click Next Step after entering their billing information, and enter credit card information in a separate step.

Adding Terms and Conditions to Reference Cart & One Page Checkout BETA

You can extend the reference solution so that customers must agree to a set of terms and conditions before completing web orders.

To extend the reference solution to include terms and conditions:

1. Put copies of the files you will modify in the Custom Checkout application folder.
 1. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout.
 2. Copy the files listed below from the Reference Checkout application folder to the corresponding folders in the Custom Checkout application folder:
 - /services/order.ss
 - /js/site.js
 - /js/combiner.config
 - /templates/checkout_reviewinfo.txt
 - /templates/templates.config
2. Go to Setup > Site Builder > Setup Web Site.
 1. Click the Shopping subtab.
 2. Find the **Review & Submit Page** section.
 3. Check the **Require Terms and Conditions** box.
 4. Click Save.
3. Add service methods to order.ss for setting terms and conditions.
 1. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > services.
 2. Click Edit next to **order.ss**, and then click the Edit link in the file record.
 3. Add code similar to the highlighted lines in the sample below:


```

    }
    else if (method == 'setTermsAndConditions') {
        var bChecked = params.bChecked == 'T';
        nlapiGetWebContainer().getShoppingSession().getOrder().setTermsAndConditions(bChecked);
    }
    else if (method == 'get' || method == 'getAll') {
    }
    else {
        returnVal = JSON.stringify({"header": {"status": {"code": "ERR_NO_METHOD_FOUND", "message": "No method found."}} });
        response.writeLine(returnVal);
        return;
    }

    // Add promocodes, giftcertificates and terms and conditions.
    var order = nlapiGetWebContainer().getShoppingSession().getOrder().getFieldValues(["promocodes", "giftcertificates", "agreetermcondition"]);
    var promocodes = order.promocodes;
    var giftcertificates = order.giftcertificates;

    if (promocodes && promocodes.length > 0) {
        retobj.result.promocode = promocodes[0];
    }

    if (giftcertificates && giftcertificates.length > 0) {
        retobj.result.giftcertificates = giftcertificates;
    }

    retobj.result.agreetermcondition = (order.agreetermcondition == 'T');

    returnVal = JSON.stringify(retobj);
}

```

4. Save your changes to order.ss.
4. Modify site.js, to set terms and conditions during the checkout process.
 1. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > js.
 2. Click Edit next to site.js, and then click the Edit link in the file record.
 3. Add code similar to the sample below:

```
var NSOrderModel = Class.create(NSDataModel, {
  initialize: function ($super, service) {
    $super(service);
    this.order = null;
  },
  refresh: function ($super, xParams) {
    if (this.order === null || $super(xParams)) {
      this.order = null;
      return true;
    }
    return false;
  },
  load: function (mp) {
    var model = this;
    var serviceParams = {
      method: "getAll"
    };
    this.comm(serviceParams, mp, {
      success: function (result) {
        model.order = result;
      }
    });
  },
  setTermsAndConditions: function(mp) {
    var model = this;
    var serviceParams = {
      method: "setTermsAndConditions",
      params: {
        bChecked: mp.args.bChecked
      }
    };
    this.comm(serviceParams, mp);
  },
  placeOrder: function (mp) {
    var model = this;
    var serviceParams = {
      method: "placeOrder"
    };
    this.comm(serviceParams, mp, {
      success: function (result) {
        model.order = result;
      }
    });
  }
});
```

5. Modify site.js to load order data. You can use the code sample below:

```
});
},
review: function (renderFuncObj) {
  load(_nsOrderModel, {
    callback: load.bind(_ctx, _nsCartModel, {
      renderFunc: renderFuncObj.cart
    })
  });
},
confirmation: function (renderFuncObj) {
  renderFuncObj.all();
}
};
```

6. Modify site.js to include the order object in rendering the review step.

```

    }
  },
  review: {
    step: {
      init: function () {
        $_konst.sideMiniCart().hide();
      },
      cart: function () {
        $("reviewInfo").update(_templates["reviewInfoTpl"].render({cart: _nsCartModel.cart, order: _nsOrderModel.order, checkout: true}));
      }
    },
    summary: {
  
```

7. Modify site.js to include an event handler for setting terms and conditions.

```

    }

    function placeOrderCallback(result, args) {
      _nsStepMgr.stepToNextStep();

      _analytics.trackOrder(result);

      load(_nsCartModel, {refresh:true}); // get the new empty cart -- no need to draw it.
      BDK.fire("checkoutComplete");
    }

    function setTermsAndConditionsBridge(evt) {
      setTermsAndConditions(evt.memo.bChecked);
    }

    function setTermsAndConditions(bChecked) {
      var checked = bChecked ? "T" : "F";
      _nsOrderModel.setTermsAndConditions({
        args: {bChecked: checked},
        callbacks: { success: clearError }
      });
    }

    function showTermsAndConditionsBridge(evt) {
      var content = BDK.tpl("checkout_tmpl_terms_and_conditions").render();

      showFormInLightView({
        content: content,
        title: "Terms And Conditions"
      });
    }

    /*
     * Guest checkout -- ability to create a new account
     */
    function createAccountBridge(evt) {
      validateForm("newCustomerLoginForm", _ctx, createAccount);
    }
  }

```

8. Modify site.js to handle new events for terms and conditions.

```

/* Placing the order */
BDK.observe("customPlaceOrder", placeOrderBridge, this, false, "customPlaceOrder");

BDK.observe("createAccount", createAccountBridge, this, false, this.name + ": create account");

BDK.observe("customCountryChange", countryChangeBridge, this, false, "customCountryChange");

BDK.observe("customSetTermsAndConditions", setTermsAndConditionsBridge, this, false, "customSetTermsAndConditions");
BDK.observe("customShowTermsAndConditions", showTermsAndConditionsBridge, this, false, "customShowTermsAndConditions");

$(document.body).insert(div = new Element("div", {
  id: "_konst.formContainer"
})).hide();

document.observe("lightview:hidden", function (event) {
  $(_konst.formContainer).scrollTop = 0;
});

```

9. Modify the template, **checkout_reviewinfo.txt** to include terms and conditions.

```

    ${orderSummary(obj)}
  </div>

  <div style='clear:both;'>
    {var checked = obj.order.agreetermcondition ? "checked='checked'" : ""}
    <input type='checkbox' onclick='BDK.fire("customSetTermsAndConditions", {bChecked: this.checked});'
    ${checked}> I confirm I have read and agree to the <a onclick='BDK.fire("customShowTermsAndConditions"); return false;'>terms and
    conditions</a>
  </div>

  <div class="btn-place clearfix">
    <a onclick='BDK.fire("customPlaceOrder");return false;' class="button3 fl-r">Place Your Order <em>&raquo;</em></a>
  </div>

</div>

```

10. Create a new template, **checkout_termsandconditions.txt**. This file should include the terms and conditions you want customers on your site to agree to. You must also include a tag on the file record.

1. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > templates.
2. Click Add File to upload your new template file to the file cabinet. Note the code sample below:

```

<div class="main-wrapper popup" style="left:auto; right:0;">
  <div class="popup-top"></div>
  <div class="popup-frame clearfix">
    <div class="popup-hold clearfix">
      <div class="popup-heading clearfix">
        <h2>Terms And Conditions</h2>
      </div>

      THE LEGAL AGREEMENTS SET OUT BELOW GOVERN YOUR USE OF MY WEB STORE. TO AGREE TO THESE TERMS, CLICK "AGREE."

    </div>
  </div>
  <div class="popup-btm"></div>
</div>

```

3. Click Edit next to the file.
4. Enter **checkout_tmpl_terms_and_conditions** in the Tag field.

5. Save.

Reference Cart & One Page Checkout BETA Version 1.1

Reference Cart & One Page Checkout BETA Version 1.1 includes updates to the layout of the customized cart, checkout, and login pages, along with the source files and assets to support them. Version 1.1 also includes additional files to support enhanced functionality.

You are not required to install Version 1.1. However, if you choose to install the new version of the reference implementation, read [Installing Reference Cart & One Page Checkout Beta Version 1.1](#).

Reference Cart & One Page Checkout BETA Version 1.1 includes the following enhancements:

- Support for Google Analytics
- Project File for SuiteCloud IDE Compatibility
- New JavaScript File for Site Settings
- Resolution for Insecure Content Browser Notifications
- Combination Files Copied to the Custom Checkout Folder by Default
- Template for Terms and Conditions
- Option for Shoppers to Save Credit Card Information
- Displaying Relevant Delivery Options
- Support for Multiple Languages
- Layout Enhancements

Installing Reference Cart & One Page Checkout Beta Version 1.1

You have a choice whether or not to install this new version of the bundle. An account administrator or another user with the SuiteBundler permission can perform an update or installation.

Note: NetSuite does not guarantee that the files in the new Reference Checkout SSP application folder will be compatible with your Custom Checkout SSP application.

If you already installed Version 1.0, you can update the bundle:

1. Go to Customization > SuiteBundler > Search & Install Bundles > List.

2. Select the Update option in the Action dropdown menu. For more information, see the help topic Update Available Icon on Installed Bundles Page.

If you choose to install Version 1.1 without having a previous installation, follow the steps below:

1. Go to Customization > SuiteBundler > Search & Install Bundles.
2. On the Search & Install Bundles page:
 - a. In the Location field, select Production Account.
 - b. In the Account ID field, enter **3519604**.
 - c. Click Search.
3. Click the link for Reference Cart & One Page Checkout BETA.
4. On the Bundle Details page, click Install.
5. Review the Preview Bundle Install page and click Install Bundle. (Also click OK in the popup that displays.)
6. Click Refresh to update the installation progress display.
7. After installation is complete, go to Setup > Site Builder > SSP Applications.
8. Click View next to Reference Checkout.
9. Select the site where you want to deploy reference checkout.
10. Click Deploy to Site. Here, you can choose whether to deploy all touch points. You can clear the box next to the touch points you do not want deployed to your site.

To see the reference cart in your web store, go to Setup > Site Builder > Preview Site, and then click the shopping cart link.

If you choose to uninstall Reference Cart & One Page BETA Version 1.0, and then you choose to install Version 1.1, you must remove or rename the Custom Checkout Folder prior to installing Version 1.1.

Support for Google Analytics

Support for Google Analytics tracking has been built in to Reference Cart & One Page Checkout BETA Version 1.1. Note that the Order Tracking Script HTML entered on the Analytics subtab on the Web Site Setup page does not apply when the checkout touch point has been deployed.

To use Google Analytics with Reference Cart & One Page Checkout BETA Version 1.1:

1. Go to Setup > Site Builder > Set Up Web Site.

2. Click the Analytics subtab.
3. Paste the tracking code snippet from Google Analytics in the **Addition to <head>** field. For example:

```
<script type="text/javascript">
  var _gaq = _gaq || [];
  _gaq.push(['_setAccount', 'UA-XXXXXXX-X']);
  _gaq.push(['_setDomainName', 'none']);
  _gaq.push(['_setAllowLinker', true]);
  _gaq.push(['_trackPageview']);
  (function() {
    var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
    ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') + '.google
-analytics.com/ga.js';
    var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
  })();
</script>
```

4. Click Save.

When customers submit orders on your web site, you can go to Google Analytics account and view the transactions.

Analytics data is automatically captured from web orders using the trackOrder function in site.js:

```
* Control data passed to Google Analytics
*/
var _analytics = {
  trackUrl: function( url ){
    if ( typeof _gaq != 'undefined' ) {
      _gaq.push(['_trackPageview', url]);
    }
  },
  trackOrder: function( order ) {
    if ( typeof _gaq != 'undefined' ) {
      var shippingAddress = _nsAddressModel.getSelected('shipping');

      _gaq.push([
        '_addTrans',
        order.internalid,
        "",
        _nsCartModel.cart.summary.total,
        _nsCartModel.cart.summary.tax,
        _nsCartModel.cart.summary.shippingcost,
        shippingAddress.city,
        shippingAddress.state,
        shippingAddress.country
      ]);
    }
  }
};
```

You can modify the trackOrder function to capture the information you want from web orders. Visit the [Google Developers](#) web site to understand the parameters you can pass to Google Analytics.

Additional Notes on Using Google Analytics with Reference Checkout

- You must use the latest version of the Google Analytics Code to track web site activity successfully.

- Attribute tags for tracking order confirmation data are not valid when the reference checkout touch points are deployed. Modify the trackOrder function in *site.js*.
- Checkout URLs captured by Google Analytics only include the active step in the one page checkout process. Be sure you are targeting the appropriate URLs in your Google Analytics account.

The URLs listed below are tracked in Reference Cart & One Page Checkout BETA Version 1.1:

- /cart
- /checkout/shipping
- /checkout/billing
- /checkout/review
- /checkout/confirmation
- /checkout/login-register
- /login-register

Note: When shoppers on your site use the Proceed to Checkout button to access the checkout, page the URL is **/checkout/login-register**. When shoppers click the header link or any other link to login or register, the URL is **/login-register**.

Project File for SuiteCloud IDE Compatibility

Prior to Version 1.1, an error dialog displayed when customizing SSP files using the SuiteCloud IDE. This error is associated with a reported bug in the JavaScript Development Toolkit (JSDT) plug-in for Eclipse. For more information, visit [The Eclipse bug-tracking web site](#). To resolve the error, you must disable the JavaScript validator included in the JSDT plug-in.

Reference Cart & One Page Checkout BETA Version 1.1 includes a configuration file for you to download that disables the JavaScript validator in the JSDT plug-in. Add the configuration file in the NetSuite File Cabinet, in the root folder of your SSP project.

To disable JavaScript validation from the Eclipse JSDT plug-in:

1. Click [here](#) for access to the configuration file. Note that you must be logged in to NetSuite to access the file.
2. Download the file to your computer.
3. Extract the .project file inside the ZIP file.
4. Log into NetSuite.

5. Go to Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc.

Note: This procedure expects that you are working with an SSP Application created by NetSuite. If you have created your own SSP App, install the configuration file in the root folder of your own SSP App.

6. Click Add File.
7. Upload the .project file.

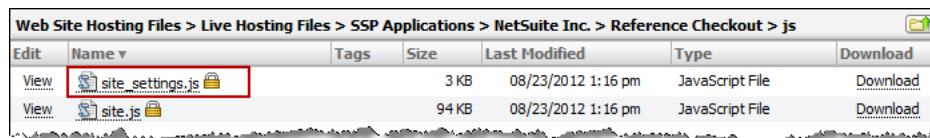
The path to the configuration file in the NetSuite file cabinet should look like this:

Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > project.

To verify the configuration file is working, you should be able to open a JavaScript file, using NetSuite SuiteCloud IDE, edit the file, and then click Save. An error dialog should not display.

New JavaScript File for Site Settings

A new file has been added to the js directory, *site_setting.js*.



| Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout > js | | | | | | |
|--|------------------|------|-------|--------------------|-----------------|----------|
| Edit | Name ▾ | Tags | Size | Last Modified | Type | Download |
| View | site_settings.js | | 3 KB | 08/23/2012 1:16 pm | JavaScript File | Download |
| View | site.js | | 94 KB | 08/23/2012 1:16 pm | JavaScript File | Download |

You can now modify the location of your sitesettings.ss service using the JavaScript file directly. Note that the variable declaration in this file points to the sitesetting.ss file. So you can change the name and the location of the file.

```
var _service = 'sitesettings.ss'
```

Use this new site settings file for making changes to the naming conventions in the reference implementation. You can also change the structure of objects used in the site settings services file, and then make corresponding changes in the *site_settings.js* file.

Resolution for Insecure Content Browser Notifications

Previously, in Reference Cart & One Page Checkout Version 1.0, the SSP file associated with the checkout touch point prompted an insecure content notification when shoppers clicked on the checkout page. Now, in Version 1.1, the insecure content notification has been resolved.

You can also use the new *redirect* API parameter in *site.js* to redirect shoppers after login.

```

function redirectUserAfterLoginRegister(result) {
  if (result.redirecturl) {
    global.location = result.redirecturl;
  }
  else {
    global.location = BDK.siteSettings.getSiteSettings().touchpoints.checkout;
  }
}

function loginUserBridge (evt) {
  validateForm("returningCustomerLoginForm", _loginCtx, loginUser);
}

function loginUser(input) {
  // get origin parameter. It indicates where we came from and will redirect accordingly. [checkout|customercenter].
  // empty origin means it's from shopping.
  if (N.cart.items.items && N.origin != 'checkout') {
    input.redirect = 'checkout';
  } else {
    input.redirect = N.touchpoints.customercenter;
  }
  _nsUserModel.login({args:input, callbacks:{success:loginUserCallback, failure:loginUserFailedCallback}});
}

function loginUserCallback(result, args) {
  redirectUserAfterLoginRegister(result);
  BDK.fire("userLoggedIn");
}

```

The redirect parameter can be used in the following calls:

- `nlapiGetWebContainer().getShoppingSession().registerCustomer(params);`
- `nlapiGetWebContainer().getShoppingSession().login(params)`

For example, *params.redirect* can specify where to redirect the shopper after login or registration. Valid redirect parameters include: *checkout*, *customercenter*, or any of the shopping URLs on your web site.

The code sample below takes a customer to the checkout page after she logs in:

```

params = { email : 'jane@email.com',
           password : 'bar',
           redirect : 'checkout' }
nlapiGetWebContainer().getShoppingSession().login(params);

```

Important: For security reasons do not pass the redirect URL as a parameter in the address bar. This value should be either hard coded in site.js or loaded through one of the services included in the Reference Checkout folder.

Combination Files Copied to the Custom Checkout Folder by Default

Previously, when you extended the reference cart and checkout solution to create a custom solution, you had to copy configuration files along with the JavaScript and template files you wanted to customize.

Now, in Version 1.1, combination files are automatically installed in the Custom Checkout application folder when you install the bundle. Also, when you copy a template or JavaScript file to the Custom Checkout application folder, the combiner file is automatically updated.

| Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > js | | | | | | |
|---|----------------------|------|-------|--------------------|--------------------|----------|
| Edit | Name ▲ | Tags | Size | Last Modified | Type | Download |
| Edit | combiner.config | | 1 KB | 08/23/2012 1:16 pm | Configuration File | Download |
| Edit | current.js | | 1 KB | 08/23/2012 1:16 pm | JavaScript File | Download |
| Edit | site-0139552132e4.js | | 49 KB | 08/23/2012 1:16 pm | JavaScript File | Download |
| Edit | site.js | | 94 KB | 08/23/2012 1:16 pm | JavaScript File | Download |

| Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Custom Checkout > templates | | | | | | |
|--|--------------------------|------|-------|--------------------|--------------------|----------|
| Edit | Name ▲ | Tags | Size | Last Modified | Type | Download |
| Edit | combined-01394b26fda0.js | | 67 KB | 08/21/2012 2:46 pm | JavaScript File | Download |
| Edit | combined-013955213de0.js | | 67 KB | 08/23/2012 1:16 pm | JavaScript File | Download |
| Edit | current.js | | 1 KB | 08/23/2012 1:16 pm | JavaScript File | Download |
| Edit | templates.config | | 1 KB | 08/23/2012 1:16 pm | Configuration File | Download |

Template for Terms and Conditions

A new template for Terms and Conditions is included in Reference Cart and One Page Checkout BETA Version 1.1.

After you deploy the checkout touch point, set the preference on the Web Site Setup page to display Terms and Conditions. A check box will automatically display in the Billing & Payment step. Use the Terms and Conditions HTML field (on the Web Site Setup page) to create the content you want to display.

Customers can click the terms & conditions link in the Billing & Payment step to display the content in a light box overlay.

Terms & Conditions

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

+ Apply Gift Card or Gift Certificate

+ Apply Promo Code

☒ Subscribe to Reporting - Static Data Email Newsletter.

☒ I AGREE to the terms & conditions

If you choose to display terms and conditions on your site, then customers must check the box, indicating that they agree, in order to successfully complete the checkout process.

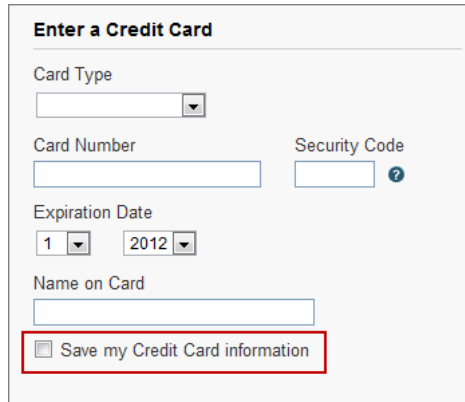
To set up terms and conditions in Reference Cart & One Page Checkout BETA Version 1.1:

1. Go to Setup > Site Builder > Set Up Web Site.
2. Click the Shopping Subtab.
3. Find the Review & Submit Page section.
 1. Check the Require Terms and Conditions box.
 2. Enter your content in the Terms and Conditions HTML field.
4. Click Save.

You can expand the billing step, and then click terms & conditions to view the content on your site. Note that you also have access to the template that renders the light box on the checkout page. You can copy *checkout_termsandconditions.txt* from the Reference Checkout folder to the Custom Checkout folder to make further customizations.

Option for Shoppers to Save Credit Card Information

In Version 1.1, based on the preferences you set on the Web Site Setup page, shoppers on your site have the option to save their credit card information when they check out as guests, or register for customer accounts.



Enter a Credit Card

Card Type

Card Number

Security Code
 ?

Expiration Date
1 2012

Name on Card

☐ Save my Credit Card information

If shoppers choose not to check the box, then they will need to enter credit card information each time they submit orders on your site.

Displaying Relevant Delivery Options

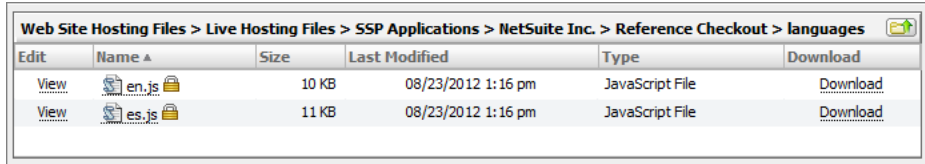
Prior to Reference Cart and One Page BETA Version 1.1, when customers arrived at checkout, shipping methods that were not relevant to the shopper's address were displayed.



Now, when the shopper arrives at the Shipping & Delivery step, the Delivery section is empty until the customer enters a valid address.

After the address is entered, irrelevant delivery options fade, to reveal only the appropriate options and values.

Support for Multiple Languages

New functions have been created in *site.js* and a new folder is installed by the bundle.

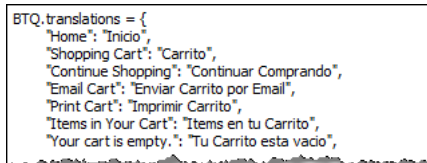


| Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. > Reference Checkout > languages | | | | | |
|---|---|-------|--------------------|-----------------|--------------------------|
| Edit | Name ▲ | Size | Last Modified | Type | Download |
| View |  en.js | 10 KB | 08/23/2012 1:16 pm | JavaScript File | Download |
| View |  es.js | 11 KB | 08/23/2012 1:16 pm | JavaScript File | Download |

If you support multiple languages in your NetSuite account, you can make those languages available when you use Reference Cart & One Page Checkout BETA Version 1.1. Shoppers on your web site can use the language selector to choose a language before beginning the checkout process.

Each file in the languages folder contains a JavaScript object, where the keys are the original English text strings and the values are the translations. You can use one of the JavaScript files in the languages folder as a starting point for supporting multiple languages on your web site.

For example, the Spanish text file (*es.js*) includes a translation for each string of text on the web site:



```
BTQ.translations = {  
  "Home": "Inicio",  
  "Shopping Cart": "Carrito",  
  "Continue Shopping": "Continuar Comprando",  
  "Email Cart": "Enviar Carrito por Email",  
  "Print Cart": "Imprimir Carrito",  
  "Items in Your Cart": "Items en tu Carrito",  
  "Your cart is empty.": "Tu Carrito esta vacio",  
}
```

To add a language to your web site:

1. Copy the languages folder into the Custom Checkout application folder.
2. Copy one of the JavaScript files provided, and rename it using the language code as the file name.

For example, if you want to create French translations for your site, copy the file, *es.js*, and then rename it *fr.js*.

3. Modify the file by adding translations for each text string.

Note that when you use the standard language code (ISO 639-1) as the file name, the language is automatically referred to by the SSP file. To add languages to the reference implementation, call the function that supports multiple languages from a template file or a JavaScript file.

Use this syntax in a template:

```
{{"The text you want to translate"|translate}}
```

Use this syntax in a JavaScript file:

```
BDK.translate("The text you want to translate")
```

You can also include variables in the text. For example, you might want to create a message that includes a variable to show the promotion code entered by a customer on your site: "Your promo code XYZ is not valid."

Use this syntax in a template:

```
{{"Your promo code $(0) is not valid"|translate: promocode_variable}}
```

Use this syntax in a JavaScript file:

```
BDK.translate("Your promo code $(0) is not valid", promocode_variable)
```

Note that \$(0) will be replaced by the value stored in *promocode_variable* in the JavaScript file. If they are using a template, where is the variable declaration, in *site_settings.js*?

Layout Enhancements

Reference Cart & One Page Checkout BETA Version 1.1 includes the following enhancements to the layout of the customized login, cart, and checkout pages:

- The login/registration has been revised to be context aware. The display and functionality will update based on how the customer arrives on the page.

SIGN IN

Returning Customers

Sign in below to checkout with an existing account.

Email Address

Password

[Sign In](#) [Forgot password?](#)

New Customers

Enter your information below to create an account.

Name

Email Address

Password

Re-enter Password

Password Hint

[Create an Account](#)

Benefits of creating an account:

- Expedite future checkouts.
- Manage your orders.

If a customer clicks the global sign in or register link, the word “Checkout” is removed from the labels and buttons, and the guest checkout button is hidden. Registering or signing in takes the customer to the My Account Page.

CHECKOUT

Returning Customers

Sign in below to checkout with an existing account

Email Address

Password

Sign In & Checkout

Forgot password?

New Customers

Checkout without an account

You will have an opportunity to create an account at a later time.

Checkout

- or -

Create an account & checkout

Enter your information below to create an account.

Name

Email Address

Password

Re-enter Password

Password Hint

Create an Account

Benefits of creating an account:

- Expedite future checkouts.
- Manage your orders.

Customers who click a checkout touchpoint are offered three different options: to sign in and checkout, checkout as a guest, or create an account. After customers log in or create an account, they are taken to the first step of the checkout process.

- The Billing & Payment step has been rearranged for better user experience.

2 Billing & Payment

Select your billing address

Add a New Address

Jane Smith

123 main

oakland, CA

94602 US

1234567891

Edit | Delete

Select a Credit Card

Add a New Credit Card

VISA

Jane Smith

VISA *****5100

06/2023

Edit | Delete

+ Apply Gift Card or Gift Certificate

+ Apply Promo Code

☒ Subscribe to Reporting - Static Data Email Newsletter.

☐ I AGREE to the [terms & conditions](#)

Next: Review >

- New address and new credit card links are top aligned.
- After shoppers enter a new billing or shipping address in the web store, this new address is automatically selected from the list of addresses.
- Gift certificate and promotion code entry fields are now expandable.
- A check box for Newsletter signup is marked by default
- The Next Step button has been relabeled to include the title of the next step.
- Products displayed in the order summary are top aligned.
- When more than three products are listed in the order summary, the list is scrollable.