

SuiteCommerce Release Notes



Copyright © 2005, 2018, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

If this document is in public or private pre-General Availability status:

This documentation is in pre-General Availability status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

If this document is in private pre-General Availability status:

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your pre-General Availability trial agreement only. It is not a commitment to deliver any material, code, or functionality,

and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Master Agreement, Oracle License and Services Agreement, Oracle PartnerNetwork Agreement, Oracle distribution agreement, or other license agreement which has been executed by you and Oracle and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

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

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

Sample Code

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

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

No Excessive Use of the Service

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

Beta Features

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

Table of Contents

| | |
|--|-----|
| SuiteCommerce Release Notes | 1 |
| Aconcagua Release of SuiteCommerce | 1 |
| Commerce Applications | 1 |
| Site Management Tools | 7 |
| Commerce Platform | 8 |
| Kilimanjaro Release of SuiteCommerce Advanced | 13 |
| Elbrus Release of SuiteCommerce Advanced | 19 |
| Vinson Release of SuiteCommerce Advanced | 28 |
| Mont Blanc Release of SuiteCommerce Advanced | 34 |
| Denali R2 Release of SuiteCommerce Advanced | 47 |
| Denali Release of SuiteCommerce Advanced | 49 |
| Release Notes Prior to SuiteCommerce Advanced Denali | 56 |
| Reference ShopFlow | 57 |
| Reference Checkout | 86 |
| Reference My Account | 96 |
| Core Improvements for SuiteCommerce Advanced 2015.1 | 108 |
| User Experience Improvements for SuiteCommerce Advanced 2015.1 | 110 |

SuiteCommerce Release Notes

- [Aconcagua Release of SuiteCommerce](#)
- [Kilimanjaro Release of SuiteCommerce Advanced](#)
- [Elbrus Release of SuiteCommerce Advanced](#)
- [Vinson Release of SuiteCommerce Advanced](#)
- [Mont Blanc Release of SuiteCommerce Advanced](#)
- [Denali R2 Release of SuiteCommerce Advanced](#)
- [Denali Release of SuiteCommerce Advanced](#)



Important: All features described in SuiteCommerce releases may not be available in your NetSuite account. Your access to these features is subject to the terms of service in your NetSuite contract. Some features may require extra purchase.

Aconcagua Release of SuiteCommerce

Welcome to the Aconcagua release of SuiteCommerce!

The Aconcagua release of SuiteCommerce introduces a new product line, **SuiteCommerce Standard**, along with a major change to how you can enhance your web store experience using the **Extensions Framework**. Custom functionality is created as an extension which you can then install and activate for your account. Extensions can be developed and shared by our partner community which then provides you with a range of choices for improving your web store to meet your business needs.

These changes are detailed in this document along with related commerce applications, commerce platform, and Site Management tools enhancements.

The following bundles are available:

| Bundle ID | SuiteApp |
|-----------|------------------------------------|
| 222167 | SuiteCommerce Standard |
| 228895 | SuiteCommerce Extension Management |
| 228676 | SuiteCommerce Base Theme |
| 222173 | SuiteCommerce Configuration |
| 53051 | Reference Product Lists Records |

For details on each set of enhancements, see:

- [Commerce Applications](#)
- [Site Management Tools](#)
- [Commerce Platform](#)



Important: You **must** install the Aconcagua release of SuiteCommerce in a **NetSuite 2018.1** account. Many of the features and enhancements in this release have dependencies on NetSuite 2018.1.

Commerce Applications

The Aconcagua release of SuiteCommerce includes the following enhancements:

- SuiteCommerce Standard
- Themes and Extensions
- Extensibility API
- SuiteCommerce Base Theme
- SuiteCommerce Theme Customizer Support
- Documentation Changes
- Copy Configuration
- Intercompany Cross-Subsidiary Fulfillment
- Free Gift Promotion
- Instagram Gallery Extension
- Libraries Update
- Standard Header and Footer in Checkout

SuiteCommerce Standard

SuiteCommerce Standard is the newest addition to SuiteCommerce. This platform lets you create, customize, and configure dynamic and engaging web stores without requiring the in-depth development required of SuiteCommerce Advanced (SCA). With SuiteCommerce Standard, you can quickly activate predefined **themes** to alter the look and feel of a web store or activate **extensions** to introduce functionality and features. This lets site administrators build SuiteCommerce Standard web stores quickly, without requiring development or access to the core functionality or application architecture.

SuiteCommerce Standard also leverages the new Site Management Tools Theme Customizer, which enables Site Management Tools administrators to customize a theme through the SMT user interface without editing Sass or HTML templates.

SCS requires the implementation of a few SuiteApps. SuiteCommerce Standard, SuiteCommerce Extensions Management, SuiteCommerce Base Theme, and SuiteCommerce Configuration SuiteApps are all required.

Note: SuiteCommerce Standard is available as a managed SuiteApp. This means that all updates to the platform are automatic. All upcoming changes to SuiteCommerce Standard will be communicated in advance of any released updates.

Because SuiteCommerce Standard is built on the same architecture as SuiteCommerce Advanced, much of the online documentation applies to both platforms. The documentation highlights what product or product release applies to each topic. Look for the **Applies To** tags within each topic to determine what application or release the topic applies to.

Note: Some procedures in this documentation direct you through NetSuite user interface elements, such as menus and menu items, that state **SuiteCommerce Advanced**. In these cases, follow the procedures as directed.

Themes and Extensions

With this release of SuiteCommerce, users implement themes and extensions to improve their web stores. Themes and extensions are self-contained packages that extend your site in some way. Site

administrators can install themes and extensions as bundled SuiteApps or as custom-built code deployed to an account by in-house theme designers and site developers.

To implement this functionality to your SuiteCommerce Standard or SuiteCommerce Advanced web store, you must install the SuiteCommerce Extensions Management SuiteApp, which adds the Manage Extensions Wizard to your account.

Manage Extensions Wizard

This wizard lets SuiteCommerce Advanced and SuiteCommerce Standard site administrators:

- Activate predefined themes to alter the look and feel of a web store.
- Activate predefined extensions to introduce functionality and features.

Predefined themes and extensions are available as SuiteApps using the NetSuite SuiteBundler. Predefined (published) themes and extensions must be installed into a NetSuite Account before you can activate them. Themes and extensions can also be deployed to an account by site developers using the theme and extension developer tools.

See the help topic [Themes & Extensions](#) for more information on installing and activating themes and extensions.

Theme and Extension Developer Tools

One key aspect of themes and extensions is how things change from previous implementations of SuiteCommerce. You no longer edit distro files and compile code for deployment. Now, you deploy a theme or extension to a NetSuite account, making it available for activation (using the Manage Extensions Wizard). When an administrator activates a theme or extension, NetSuite then compiles the code and applies it to a specified domain.

The Extensions Management SuiteApp includes two different developer tools, one specific to creating themes, the other for extensions. In this way, SuiteCommerce developers are now theme developers, extension developers, or both.

These developer tools let SuiteCommerce Standard and SuiteCommerce Advanced site developers and partners:

- Develop Sass and HTML templates and deploy as themes.
- Expose Sass variables to the Site Management Tools Theme Customizer.
- Develop JavaScript, SuiteScript, or configuration objects that are accessible using the Extensibility API and deploy as extensions.
- Test themes and extensions on a local server.
- Create Custom Content Types (CCTs) as extensions.

Extensibility API

This release of SuiteCommerce introduces a new way to customize and extend your SuiteCommerce web store. SuiteCommerce developers now create extensions using the Extensibility API to introduce new features or functionality. Developers build code within an extension to make API calls to components. These components then interact with the application. This separates custom developer code from the application itself and helps ensure future extensibility of the application.

The best practice is to build extensions using JavaScript, SuiteScript, or configuration objects available within the Extensibility API. See the help topic [Extensibility Component Classes](#) for a list of components available using the Extensibility API.

Note: If you are customizing objects that are not accessible using the Extensibility API, you can use the core SCA developer tools and customization practices to customize these objects.

For more details on the Extensibility API and procedures on how to access components while building extensions, see the help topic [Extensibility API](#).

SuiteCommerce Base Theme

Along with the introduction of themes and extensions, this release includes the SuiteCommerce Base Theme SuiteApp. Best practice for theme developers is to use this theme as a baseline for any theme development. Simply install this SuiteApp and use the Manage Extensions Wizard to activate it for a domain. Then, use the theme developer tools to fetch the theme files into your theme development workspace and start developing your theme.

You must activate a theme before you can develop or test code locally. This theme is built using SuiteCommerce best practices for creating HTML and Sass and using established an [Design Architecture](#) for developing Sass.

- For information on how to install this theme as a SuiteApp, see the help topic [Install Your SuiteCommerce Application](#).
- For information on using the theme developer tools to fetch these files to use a a baseline theme, see the help topic [Theme Developer Tools](#).
- For information on developing your theme, see the help topic [Themes](#).

SuiteCommerce Theme Customizer Support

With this release of SuiteCommerce, theme developers can customize Sass files to expose variables to the Site Management Tools Theme Customizer. Within your Sass structure, you can also establish a schema for displaying each variable in the SMT user interface.

Exposing Sass takes the form of an `editable()` function call wrapped inside a comment in a theme's Sass files. When a site administrator activates a theme with exposed Sass, SMT displays those variables, and SMT administrators can alter the exposed variables to further customize their web store.

This customization can include preset skins. A skin is a selectable set of predefined settings that change the appearance of a theme in SMT. One theme can include any number of skins that SMT administrators select to change that theme's appearance. From a development perspective, a skin is a JSON file that defines each exposed variable and the new values.

See the help topic [Expose Sass Variables for Customization](#) for details.

Documentation Changes

With this release of SuiteCommerce, the online documentation has been reorganized. For the web store topics, we have separated most developer-specific content from user content.

Note: URLs to most topics have not changed, so any local bookmarks to existing topics are maintained.

You will now see two new sections as follows:

Storefront – These topics explain everything you need to know to set up your web store to deliver a full shopping, checkout, and my account experience to your customers. Some of what you can learn about from here includes how to:

- Apply themes and extensions

- Set up and manage transactions
- Prepare items for availability and search in your store
- Optimize your store for SEO and analytics

Developer Documentation – These topics are written for theme and extension developers and web store administrators. Some of what you can learn about from here includes how to:

- Set up domains
- Create themes and extensions
- Use the available APIs to customize your web store experience

Copy Configuration

With this release of SuiteCommerce, you can use the Copy Configuration functionality to copy a SuiteCommerce configuration record from one domain to another.

Copy Configuration is part of the SuiteCommerce Configuration SuiteApp. After installing this SuiteApp, Copy Configuration is available by going to Setup > SuiteCommerce Advanced > Configuration and selecting your website and domain.

If you are implementing an SCA site, the two domains do not have to be on the same release. For example, you can copy a Kilimanjaro configuration record to another domain on a later release of SCA. Since it is common for later versions of SCA to have fields that do not exist in previous releases, the tool copies the values of all fields that are common between the two releases and uses default values for the additional fields.

Common scenarios where this feature is useful include:

- Moving a web store from staging to production
- Moving from one SuiteCommerce Advanced release to another

For additional information see the help topic [Copy a SuiteCommerce Configuration Record](#).

Intercompany Cross-Subsidiary Fulfillment

With this release, SuiteCommerce supports Intercompany Cross-Subsidiary Fulfillment.

This feature only applies to OneWorld accounts with multiple subsidiaries and requires proper setup in NetSuite. See the help topic [Intercompany Cross-Subsidiary Fulfillment](#) for setup information.

After you properly configure Intercompany Cross-Subsidiary Fulfillment in NetSuite, users can order an item from a subsidiary site regardless of inventory level at that subsidiary as long as there is sufficient inventory to fulfill the order at a related subsidiary. This capability applies to all types of orders including ones set to ship or set for pickup in store.

Free Gift Promotion

With this release of SuiteCommerce, the SuitePromotions Free Gift option is available in your webstore. After adding a Free Gift promotion, the free gift is automatically applied to qualifying orders. Users can elect to delete the free gift if they do want to receive the gift item.

Keep the following in mind with Free Gift promotions:

- Free Gift promotions do not work with the Multi Ship To feature.
- Best Offer logic does not apply to Free Gift promotions.
- Parent matrix items cannot be selected as a free gift.

- Items with options cannot be selected as a free gift.

If your site is not already configured to use SuitePromotions, see the help topic [Promotions](#) for information about how to add SuitePromotions to your site.

Instagram Gallery Extension

With this release of SuiteCommerce, you can add Instagram feeds to your web store. This feature is available as an extension for SuiteCommerce Standard and sites implementing the Aconcagua release of SCA only. See the help topic [Themes & Extensions](#) for information on installing and activating extensions.

To use this feature, install the SuiteCommerce Instagram Gallery SuiteApp (Bundle ID: 228696), activate the Instagram extension, and use Site Management Tools (SMT) to add Instagram feeds to your web store.

You can add multiple instances of Instagram on your web store and can add them to all pages, pages of a certain type, or only on specific pages. You can configure each instance with different Instagram feeds based on username, hashtag, or combination of both. Note that you will need to acquire a free Instagram Access Token from Instagram.com to successfully access Instagram feeds (see [Instagram Authentication](#)).

For more information, see the help topic [Instagram Gallery Extension](#).

Libraries Update

With this release of SuiteCommerce, the third-party libraries shown in the table below have been updated. Library version information is also shown in the distro.json file. If you are migrating to the Aconcagua release of SuiteCommerce Advanced, ensure that no deprecated methods exist in your implementation before migrating.

| Library | Old Version | New Version |
|------------------------|-------------|-------------|
| Almond | 0.3.0 | 0.3.3 |
| Backbone | 1.1.2 | 1.3.3 |
| Backbone.Stickit | 0.8.0 | 0.9.2 |
| Bignumber | 1.5.0 | 4.0.4 |
| html5shiv | 3.7.2 | 3.7.3 |
| jQuery-Zoom | 1.7.13 | 1.7.20 |
| json2 | 2.0.1 | 2.0.2 |
| require | 2.1.15 | 2.3.5 |
| twitter-bootstrap-sass | 3.3.1 | 3.3.7 |
| underscore | 1.7.0 | 1.8.3 |

Standard Header and Footer in Checkout

With this release of SuiteCommerce, you can elect to use the standard website header on checkout pages rather than the default checkout header and footer. This property is available on the Configuration record Checkout Tab.

For additional information, see the help topic [Use Standard Header and Footer](#).

Site Management Tools

This version of SuiteCommerce Site Management Tools includes the following enhancements:

- [Content Visibility Dates](#)
- [Landing Page Visibility Dates](#)
- [Theme Customizer](#)

Content Visibility Dates

This release introduces a significant change to the way you publish content and the visibility of content on your site. Previously with SMT you had a published and an unpublished version of your site. When you published content it was immediately visible on your site, and publishing applied all of your unpublished changes to the site. This included edits to existing content, addition of new content, and content deletions.

This release adds content visibility dates, an important new feature to SMT. Now when you add a piece of content to your site, you can set visibility dates for that content. Visibility options include the following:

- Always Visible
- Display at a certain date and time and never expire
- Display at a certain date and time and expire at a certain date and time

Additions and changes to content must still be published, but now you have the ability to selectively publish content and changes. The content is visible based on the visibility options you specify for that content.

Other important and related features include:

- Visibility time line to see upcoming content changes based on the visibility dates
- Preview the site for future date and time
- Include or exclude unpublished content in the site preview

These changes impact users with SMT version 3 and the Kilimanjaro release of SCA. See the help topic [Content Visibility](#).

Landing Page Visibility Dates

Prior to this release of SMT, when you created a landing page, the page was accessible immediately on the site. This release of SMT gives you new options for controlling landing page visibility. Now when you create a landing page, you can specify when that page is visible. Visibility options include the following:

- Always visible
- Visible as of a specific date and time and never expires
- Visible as of a specific date and expire on a specific date and time.

Unlike content, landing pages do not require publishing. They display based solely on the visibility options you specify for the page. See the help topic [Landing Page](#).

Theme Customizer

This release of SMT brings you the Theme Customizer which lets you make style changes to pre-defined elements on your site. For example, you might change the size or style of heading fonts, the

background color, or the color of links. The elements available for customization must be defined in the theme that is deployed to your SCA site. These customizable elements are defined by the theme developer.

Customizable elements can be defined as **skins**, which are groups of elements designed to let you quickly change the look of your site by applying the skin. A theme can have multiple skins or no skins at all. Skins are not required for defining customizable elements.

The theme customizer is available in SMT only when your site is using a supported version of SCA and when the theme deployed to your site contains customizable elements.

The theme customizer is available only in version 3 of SMT and requires the Aconcagua release of SuiteCommerce Advanced or greater. See the help topic [Theme Customizer](#) for more information.

Commerce Platform

Commerce Platform includes platform capabilities that you can use to build and deploy a more interactive and engaging shopping experience for customers. NetSuite 2018.1 includes the following enhancements:

- [Enhanced Search Experience with Search Synonyms](#)
- [Enhancements to Sitemap Generator](#)
- [Automatic Cache Invalidation with Major Release](#)
- [Advance Notice: HTTP2 Compliant Headers](#)
- [Improved Performance with Server-Side SASS Compilation](#)
- [Monitor Domain Health Using Health Status Indicators](#)
- [Shopping Methods for Encrypted Payload Bridging](#)
- [Support for Free Gift Promotions in the Commerce API](#)
- [Maximum Order Quantity on Items Purchased Online](#)
- [Website Specific Customers](#)
- [Website Template for Registration Confirmation Email Messages](#)
- [Link to Web Site Assistant Tool Deprecated](#)
- [CSV Import Supported for Commerce Categories](#)

Enhanced Search Experience with Search Synonyms

Synonyms improve the shopper search experience by enabling them to find items using their own words. Synonyms ensure the search results include the term the shopper searched for and also items associated with the synonymous terms.

For example, you define **trousers** and **pants** as synonyms. Any shopper searching for the term **pants** now gets search results for **pants** and **trousers**.



Note: The Synonyms feature is available to only customers who have been migrated to the new item search service.

Enhancements to Sitemap Generator

Release 2018.1, introduces Sitemap Generator enhancements that let you perform the following tasks:

- Schedule automatic sitemap generation – The Sitemap Generator Scheduler enables you to configure the Sitemap Generator to automatically update your sitemap at scheduled time intervals. Scheduling ensures you always have an up-to-date sitemap available for search engines to index.
- Add Site Management Tools (SMT) landing page URLs to the Sitemap – You can now check a box to add SMT landing page URLs to your sitemap. This is in addition to the Homepage URL, Product URLs, Commerce Category and Subcategory URLs that are added by default when you generate a sitemap.

For more information about these enhancements, see the help topic [Sitemap Generator](#).

Automatic Cache Invalidation with Major Release

After a major release, it is advisable to remove outdated data from the CDN cache. Prior to 2018.1, this had to be done manually.

The 2018.1 release introduces changes that automatically invalidate your CDN cache after every major release. This is applicable to all SuiteCommerce Advanced and SuiteCommerce Standard websites that have domains deployed on CDN.

For more information, see the help topic [Automatic Cache Invalidation](#).

Advance Notice: HTTP2 Compliant Headers



Note: The target date for migration to HTTP/2 is April 2018.

NetSuite will soon be migrating to Hypertext Transfer Protocol version 2 (HTTP/2) because it enables a more efficient use of network resources.

The 2018.1 release includes code changes to convert header field names to lowercase prior to their encoding in HTTP/2. This is necessary to support compliance with HTTP/2 because HTTP/2 supports only lowercase headers.

Depending on how you use the `nlobjRequest#getAllHeaders()` method, you may need to update your customized code to be compatible with HTTP/2.

If you use it for enumerating by keys or for any other programmatic purpose, you should do one of the following:

- Use the `getHeader()` method instead of `getAllHeaders()` in your code. The `getHeader()` method has been verified as being compatible with HTTP/2.
- Check and update your code to ensure that it is compatible with HTTP/2, particularly with the lowercase headers returned by `getAllHeaders()`.

You do not need to take any action if you use `nlobjRequest#getAllHeaders()`:

- As part of a SuiteCommerce Advanced (SCA) bundle and do not use it in your customized code.
- For logging or passing to other HTTP requests.

Improved Performance with Server-Side SASS Compilation

Prior to 2018.1, Syntactically Awesome Style Sheet (SASS) was compiled in the developer's browser, which was inefficient and incompatible with the Theme Customizer.

With the upgrade to NetSuite 2018.1, SASS is compiled on the server-side. This change enables editing of SASS variable values in the browser and previewing the changes without recompiling all the source SASS files.

Monitor Domain Health Using Health Status Indicators

The new Domain Health Status feature provides information about the compliance state and status of your configured domains. It conducts health tests to monitor, troubleshoot, and diagnose the root cause of problems related to your domain setup.

The diagnosed issues are segregated into the following three categories:

- Deployment
- DNS record setup
- Domain Certificate status

The Set Up Domains page, which lists all of your domains, now displays a status icon to report each domain's health. View the domain record to see a detailed health status message that describes the problem.

The new Domain Configuration Alert reminder notifies you whenever any of your domains has a problem. To actively monitor your domain health, add the alert reminder to your reminder portlet. For instructions on how to do this, see the help topic [Setting Up Reminders](#).

Shopping Methods for Encrypted Payload Bridging

The Commerce API has two new Shopping methods associated with encrypted payload bridging:

- `getEffectiveShoppingDomain()` — returns the name of the shopping domain associated with the current session.
- `constructDomainBridgingUrl(link)` — lets you convert the link to a domain bridging URL.

Note: Although these methods are available in the 2018.1 release, the encrypted payload bridging feature will not be available until a future release.

Support for Free Gift Promotions in the Commerce API

The new Free Gift Promotion is a SuitePromotions feature that enables you to add free items to eligible transactions. Though it is similar to the existing Buy X, Get Y Promotions, Free Gift Promotion has the advantage that it can be set up to add coupon codes to eligible transactions automatically. The ability to add coupon codes manually to apply the promotion is also available.

The Commerce API has been enhanced with the following changes to accommodate this feature:

- Added a new Order method `getEligibleFreeGiftItems()`, which returns a list of all Free Gift promotions that are currently applied. It also returns the Item IDs of all the free items.
- Added a new Order method `addFreeGiftItem()`, which adds the eligible item to the cart and marks it as a free item.
- Added a new Order method `rejectFreeGiftItem()`, which permits shoppers to reject the free item provided the item is not already in the cart.
- The existing `promocode` JSON object has been extended by adding a new field `promotion_type`. Consequently, the existing Order methods `getAppliedPromotionCode(promocode, fields)`,

`getAppliedPromotionCodes(fields)`, and `getFieldValues()` return the type of promotion (if applicable) for the item along with other item attributes.

- The existing `item` JSON object has been extended by adding a new field `freegiftpromotionid`. Consequently, the existing `Order` method `getFieldValues()` returns the ID of free gift promotion.
- When the Free Gift Promotion is active, the number of free items in the shopping cart can be changed using the `updateItemQuantity()`, `updateItemQuantities()` or `removeItem()` methods. If this is done, the number of free items rejected is calculated. If the shopper later decides to accept more free items, the `getEligibleFreeGiftItems()` order method can be used to retrieve the number of free items rejected. The shopper can then add more free items to the cart up to the maximum number for which the transaction is eligible.

To use the Free Gift Promotion feature, you must enable the SuitePromotions feature as described in [Configuring Promotions](#). This option enables you to create a new Free Gift Promotion and use the promotion on the SuiteCommerce website by manually entering a coupon code.

To apply the promotion on the SuiteCommerce website automatically, in addition to enabling the SuitePromotions feature, you must enable the Auto-Apply Promotions feature as described in [Configuring Promotions](#). You must also enable the Auto-Apply Promotions feature under Promotions on the Shopping subtab of the Website Setup Record. Enabling these features automatically applies the coupon code to eligible transactions, but it does not add the free item to the shopping cart. The free item is added to the shopping cart when the SuiteCommerce Advanced bundle calls the `getEligibleFreeGiftItems()` and `addFreeGiftItem()` methods.

Note: Although the changes are available in the 2018.1 release for use with the Commerce API, the Free Gift Promotion feature will not be available in the SuiteCommerce Advanced Web Stores until the next release of SuiteCommerce Advanced.

Maximum Order Quantity on Items Purchased Online

Prior to release 2018.1, merchants could set only a minimum quantity limit for items purchased online. Now, you can also set a maximum quantity limit on the item record in NetSuite. After you define a maximum quantity, online customers are unable to submit an order unless the quantity of that item meets the maximum quantity limit. If a customer tries to order a quantity of the item greater than the maximum limit you set, a notice is displayed on the website. The customer must change the quantity before submitting the order. This capability is available for Site Builder and SuiteCommerce websites. For more information, read the help topic [Order Quantity](#).

Website Specific Customers

This feature enables merchants who run multiple websites to restrict customer login access to one site. Retailers with multiple brands, and merchants running both a wholesale site and a retail site can restrict customer access to one website over the others. Prior to release 2018.1, customers who registered on one of your websites, had access to all of your websites.

To use customer website assignment capabilities, go to Setup > SuiteCommerce Advanced > Setup Web Site, and check the box next to **Assign New Customers to This Site**. After the box is checked, all customers that register on that website get login access on that website only.

Note: The check box only affects new customers created after your account has been upgraded to 2018.1.

Two new fields were added to the customer record:

- The **Source Web Site** field shows the website on which the customer registered.

- The **Assigned Web Site** field, on the System Information subtab, enables you to change the customer website assignment on individual customer records. You can update customer records in bulk, by performing a mass update. No value in this field means the customer can login to all sites.

Note: The new fields are available on standard customer forms. If you use custom forms, you must customize the form and add the new fields.

Managing Customers

After your NetSuite account is upgraded to release 2018.1, check the **Assign New Customers to This Site** box on the Web Site Setup page to start using this feature. Existing customers are not automatically assigned to websites. You can assign existing customers to websites by performing a **Customer Web Site Assignment mass update**. To perform the mass update, go to Lists > Mass Update > Mass Updates. Select the website that you want to assign to customers, and then select search criteria to find the customer records you want to change.

You can assign a customer to only one website, not to a subset of your websites. When you enable this feature, customers can establish a customer account only on the website where they register. If a customer signs up on more than one website, then the customer gets access to a customer account for each website, and a separate customer record is created in your NetSuite account. For more information, read the help topic [Assign Customers to Websites](#).

Website Template for Registration Confirmation Email Messages

This feature introduces the capability to send a confirmation email message to online customers after they register on your website. This feature also provides an option for customizing the email message sent to customers. An Administrator or Store Manager can go to Setup > Company > System Email Template. You can select the template category, Web Site Other Notifications, and then customize the Standard Web Site Registration Confirmation email template as needed.

To start sending the Registration Confirmation email message, go to Setup > SuiteCommerce Advanced > Web Site Setup. Click the Email subtab, under the **Other Email** sublist, find the Registration Confirmation section, and then select the system email template you want to send.

Registration Confirmation

☒ SEND EMAIL

EMAIL FROM ADDRESS
info@mycompany.com

☒ BCC REGISTRATION CONFIRMATION EMAIL

EMAIL TO
salesmanager@mycompany.com

SELECT A SYSTEM EMAIL TEMPLATE
Standard Web Site Registration Confirmation

In 2018.1, the default Store Manager role has the following additional permissions and access to the following areas of NetSuite:

- Modify system email templates at Setup > Company > System Email Templates.
- Select a system email template to send as the registration confirmation email at Setup > SuiteCommerce Advanced > Web Site Setup > Email > Other Emails

Note: No changes are applied to custom Store Manager roles created prior to release 2018.1.

For more information, read the help topic [Registration Confirmation Email Messages](#).

Link to Web Site Assistant Tool Deprecated

In release 2018.1, the navigation link to the Web Site Assistant has been removed from the navigation menu under Setup > Site Builder. You can perform all the setup tasks that were performed using the Web Site Assistant in other areas of NetSuite.

When configuring a Site Builder website, go to Setup > Site Builder > Set Up Web Site to perform the following tasks:

- Configure the website domain.
- Select site templates for the appearance of your website and upload the logo.
- Select the website home page.

To set up tabs and categories for your Site Builder website, go to Lists > Web Site > Tabs, and Lists > Web Site > Categories.

CSV Import Supported for Commerce Categories

As of 2018.1, you can import CSV files to create and update commerce categories records in NetSuite.

You can import commerce categories data if the Commerce Categories feature is enabled. For more information, see the help topic [Enable Commerce Categories Feature](#).

To import commerce categories data, in the Import Assistant set the Import Type to Website, and the record type to Commerce Categories.


For more information, see the help topic [Commerce Category Import](#).

Kilimanjaro Release of SuiteCommerce Advanced

 **Applies to:** SuiteCommerce Advanced | Kilimanjaro

Welcome to the Kilimanjaro release of SuiteCommerce Advanced!



Important: You **must** install the Kilimanjaro release of SuiteCommerce Advanced in a **NetSuite 2017.2** account. Many of the features and enhancements in this release have dependencies on NetSuite 2017.2. For details on enhancements and changes in NetSuite 2017.2, see  [Release Notes 2017.2](#).

This version includes the following enhancements:

- [3D Secure Payments](#)
- [Change Email Address](#)
- [Custom Content Types](#)
- [Display of VAT/GST/PST](#)
- [Field Set Setup Script Changes](#)
- [Node.js Requirement Changes](#)
- [Sass Style Guide](#)

- [Style Definition Changes](#)
- [SuiteCommerce Configuration Updates](#)
- [SuitePromotions Enhancements](#)

The following bundles are available:

- SuiteCommerce Advanced Kilimanjaro: Bundle ID – **194217**
- Site Builder Extensions Kilimanjaro : Bundle ID – **194218**
- Site Builder Extensions Premium Kilimanjaro : Bundle ID – **194219**
- SuiteCommerce Configuration: Bundle ID – **194222**
- Reference Product Lists Records: Bundle ID – **53051**: This bundle is required even if Product Lists records are not leveraged for your implementation. If not installed, the following error is returned when shoppers navigate to the cart: The record type [CUSTOMRECORD_NS_PL_PRODUCTLIST] is invalid.



Important: To take advantage of new features in existing implementations, you must migrate the new code to your existing code base. For details, see the help topic [Migration to New Releases](#).

3D Secure Payments

With the Kilimanjaro release of SuiteCommerce Advanced, the use of 3D Secure authentication through payment provider accounts such as Cybersource or Merchant e-Solutions is supported. 3D secure authentication provides for additional fraud protection by allowing users to create and assign a password to their credit and debit cards. The password is then verified when a user processes a transaction through your store.

For more information, see the help topic [3D Secure Payment Authentication](#).

Change Email Address

With the Kilimanjaro release of SuiteCommerce Advanced, users can now change their registered email address in the My Account application. Once a user submits the change request, they receive an email to confirm the change. When the user clicks the confirmation link, a confirmation message is displayed within the login/checkout screen.



Note: This feature is available by default in the Kilimanjaro release of SuiteCommerce Advanced. For instructions on backporting the feature to previous releases of SuiteCommerce Advanced, see the help topic [Change Email Address Patch](#).

For more details, see the help topic [Change Email Address](#).

Custom Content Types

With the Kilimanjaro release of SuiteCommerce Advanced, users can create Custom Content Types.

To implement this feature, you create a custom module within your application source code. This is similar to any customization you implement in SCA. You must also set up the CCT in NetSuite and implement it into your SCA site using Site Management Tools.

 **Note:** You must be using SMT Version 3 to implement CCTs.

For more information on building a custom CCT module for SCA, see the help topic [Create a CCT Module](#).

For more information on implementing a CCT using Site Management Tools, see the help topic [Create a CCT Module](#).

Display of VAT/GST/PST

With the Kilimanjaro release of SuiteCommerce Advanced, the display of VAT, GST, and PST tax amounts for Canadian, UK, and Australian editions of NetSuite is supported. For details, see the help topic [Web Store Taxes — VAT, GST, PST](#).

Field Set Setup Script Changes

With the Kilimanjaro release of SuiteCommerce Advanced, you can now configure a custom matrix child item search field set using the SuiteCommerce Configuration record. This includes a default matrixchilditems_search field set. To account for this new functionality, the field set setup script now includes a default matrix child item field set.

For more information on the configurable properties related to this feature, see the help topic [Shopping Catalog Tab](#).

For more information on how to implement the field set setup script, see the help topic [Import Field Sets](#).

Node.js Requirement Changes

With the Kilimanjaro release of SuiteCommerce Advanced, the developer tools have been modified to support Node.js version 6.11.x (where x is the most current minor release). See the help topic [Install Node.js](#) for more information.

Sass Style Guide

With the Kilimanjaro release of SuiteCommerce Advanced, you can now use the developer tools to create a style guide automatically.

A style guide helps you document and use the various style elements defined for your site. Contributors can refer to a style guide to create new pages and elements or customize existing ones. A style guide can also ensure that your best practices for site design are being met by multiple contributors.

See the help topic [Style Guide](#) for details and instructions on how to build a style guide using the developer tools.

As a prerequisite for a style guide, SCA uses Knyle Style Sheet (KSS) notation to document and define elements within source Sass files. As a best practice when customizing SCA Sass files, use KSS notation.

See the help topic [KSS Notation](#) for details.

Style Definition Changes

To help you maintain and develop Sass variables, SuiteCommerce Advanced introduces the following Sass style definitions with the Kilimanjaro release. This provides better optimized Sass and more comprehensible variable definitions. These enhancements include the following:

- Redefined color palette, typography, and spacing definitions
- Human-readable, standardized variable definitions
- More elegant element definition through Sass code nesting

For details, See the help topic [Style Definitions](#).

Colors

Neutral shades and theme colors are handled differently from previous releases. Both neutral shades and theme color palettes now use a shading scale from 1000 (dark) to 0 (light).

Example: Kilimanjaro Release and Later

```
$sc-color-primary: #e23200;
$sc-color-secondary: #15607b;

//...

$sc-neutral-shade-base: #222426;
$sc-neutral-shade-900: $sc-neutral-shade-base;
$sc-neutral-shade-600: lighten($sc-neutral-shade-base, 18);
$sc-neutral-shade-300: lighten($sc-neutral-shade-base, 55);
$sc-neutral-shade-0: lighten($sc-neutral-shade-base, 100);

//...

$sc-color-theme-base: $sc-color-secondary;
$sc-color-theme-900: $sc-color-theme-base;
$sc-color-theme-400: desaturate(lighten($sc-color-theme-base, 45), 18);
$sc-color-theme-300: desaturate(lighten($sc-color-theme-base, 60), 22);
$sc-color-theme-200: desaturate(lighten($sc-color-theme-base, 68), 22);
```

Example: Previous Releases

```
$sc-color-theme:#5b7f8c;
$sc-color-theme-light:#9cb6bf;
$sc-color-theme-background:#e4eff5;
$sc-color-theme-background-light: lighten($sc-color-theme-background, 3.5);
$sc-color-theme-background-lightest: lighten($sc-color-theme-background, 6);
$sc-color-theme-border: desaturate(darken($sc-color-theme-background, 8), 3);
$sc-color-background-list:#f1f8fa;

$sc-color-primary: #f15c28;
$sc-color-secondary: #15607B;
$sc-color-tertiary: #ffffff;

$sc-color-success: #4a7f35;
$sc-color-success-background: #eef7e4;
$sc-color-warning: #8e7728;
$sc-color-warning-background: #f9f5cd;
$sc-color-error: #C33C48;
$sc-color-error-background: #FFE9F1;
$sc-color-info: $sc-color-theme-light;
```

```
$sc-color-info-background: $sc-color-theme-light;
```

Typography

Typography variables now employ a more familiar sizing method of extra-small (xs) through triple-extra-large (xxxl) within the variable definition. Each of these sizes is calculated from a base font size (\$sc-font-size-base) expressed as rem units (relative em).

Example: Kilimanjaro Release and Later

In the following example, the value for \$sc-font-size-m (medium) equals 15px (1 x 15px):

```
$sc-font-size-base: 15px;

$sc-font-size-xxs: 0.75rem;
$sc-font-size-xs: 0.87rem;
$sc-font-size-s: 0.93rem;
$sc-font-size-m: 1rem;
$sc-font-size-l: 1.067rem;
$sc-font-size-xl: 1.2rem;
$sc-font-size-xxl: 1.47rem;
$sc-font-size-xxxl: 1.73rem;
```

Example: Previous Releases

```
$sc-base-font-size: 15px;
$sc-smallest-font-size: 11px;
$sc-small-font-size: 13px;
$sc-h1-font-size: 26px;
$sc-h1-font-size-mobile: 22px;
$sc-h2-font-size: 22px;
$sc-h2-font-size-mobile: 18px;
$sc-h3-font-size: 22px;
$sc-h3-font-size-mobile: 18px;
$sc-h4-font-size: 18px;
$sc-h4-font-size-mobile: 15px;
$sc-h5-font-size: $sc-base-font-size;
$sc-button-large-font-size: $sc-base-font-size * 1.067;
$sc-button-medium-font-size: $sc-base-font-size;
$sc-button-small-font-size: $sc-small-font-size;
```

Spacing

SuiteCommerce Advanced now introduces **levels** to indicate the added or reduced space for a class relative to a base padding or margin style. Each level designation in the named variable equals the multiplier.

In the following example, the value for \$sc-padding-lv2 equals 10px (2 x 5px):

Example: Kilimanjaro Release and Later

```
$sc-padding-base: 5px;
$sc-padding-lv1: $sc-padding-base;
```

```

$sc-padding-lv2: $sc-padding-base * 2;
$sc-padding-lv3: $sc-padding-base * 3;
$sc-padding-lv4: $sc-padding-base * 4;
$sc-padding-lv5: $sc-padding-base * 5;
$sc-padding-lv6: $sc-padding-base * 6;
$sc-padding-lv7: $sc-padding-base * 7;
$sc-padding-lv8: $sc-padding-base * 8;

```

Example: Previous Releases

```

$sc-base-padding: 5px;
$sc-base-margin: 5px;

$sc-small-margin: $sc-base-margin * 2;
$sc-medium-margin: $sc-base-margin * 4;
$sc-large-margin: $sc-base-margin * 8;
$sc-xlarge-margin: $sc-base-margin * 12;

$sc-padding-small: $sc-base-padding * 5 $sc-base-padding * 3;
$sc-padding-medium: $sc-base-padding * 6;

$button-safe-margin: $sc-base-margin * 2;

```

SuiteCommerce Configuration Updates


With the Kilimanjaro release of SuiteCommerce Advanced, the following configuration properties have been added. Refer to the appropriate section for more information.

| Property Label (SuiteCommerce Configuration Record) | Property ID | More Information | Property Location (SuiteCommerce Configuration Record) Tab > Subtab |
|---|----------------------------|---|---|
| Require a Phone Number With Addresses | addresses.isPhoneMandatory | Require a Phone Number With Addresses | My Account > Addresses (new subtab) |
| Enable 3D Secure Payments | isThreeDSecureEnabled | Enable 3D Secure Payments | Checkout |
| If the Information is Available to your Country, shows detailed taxes in the Cart | showTaxDetailsPerLine | If the Information is Available to your Country, shows detailed taxes in the Cart | Shopping Catalog |
| Allow custom matrix child search in the item list | matrixchilditems.enabled | Matrix child items fieldset for search | |
| If the Information is Available to your Country, shows detailed taxes in the Cart | matrixchilditems.fieldset | If the Information is Available to your Country, shows detailed taxes in the Cart | |
| Favicon of the Website | faviconPath | Favicon of the Website | Advanced > Favicon Path (new subtab) |

SuitePromotions Enhancements

With the Kilimanjaro release of SuiteCommerce Advanced, the following promotions enhancements are available:


- **Auto-Apply Promotions** – You can configure promotions to be automatically applied when certain criteria are met.
- **Best Offer Promotions** – Automatically configured for promotions where auto-apply is enabled. For example, if a user has applied multiple promotions to an order including a promotion that is defined as exclusive, the best combination of promotions is automatically determined.
- **Audit and Visibility of Promotions** – Shows if a promotion was added by the user or by best offer. If a promotion is not applied, it also displays the reasons why it was not applied.

 **Note:** Due to dependencies on enhanced Commerce APIs, to use these promotions enhancements you must have the Kilimanjaro release of SuiteCommerce Advanced installed in a 17.2 NetSuite account. For details, see [Support for Auto-Apply Promotions and Best Offer in the Commerce API](#).

To get started with using Auto-Apply promotions, you must do the following:

- Enable the **SuitePromotions** feature.
- Enable **This Promotion Can Be Automatically Applied** on the promotions record and define any promotions requirements.
- Enable **Auto-Apply Promotions** on the shopping tab of the web site setup record.



For detailed information on setting up and configuring SuitePromotions in SuiteCommerce Advanced, see the help topic [Promotions](#).

 **Note:** These features are available by default in the Kilimanjaro release of SuiteCommerce Advanced. For instructions on backporting to the Elbrus release of SuiteCommerce Advanced, see the help topic [Auto-Apply Promotions for Elbrus](#).

Elbrus Release of SuiteCommerce Advanced

 **Applies to:** SuiteCommerce Advanced | Elbrus

Welcome to the Elbrus release of SuiteCommerce Advanced!

 **Important:** You **must** install the Elbrus release of SuiteCommerce Advanced in a **NetSuite 2017.1** account. Many of the features and enhancements in this release have dependencies on NetSuite 2017.1. For details on enhancements and changes in NetSuite 2017.1, see  [Release Notes 2017.1](#).

This version includes the following enhancements:

- [Custom Field Support](#)
- [Facets as URL Parameters](#)
- [Handlebars.js Helper Additions](#)
- [Internet Explorer 8 No Longer Supported](#)

- Mastercard 2-Series BIN Range Support
- Multi-Image Option Enhancements
- Pickup In Store
- Product Details Page Architecture Changes
- Quantity Pricing
- Quick Order
- Script Debugger Change for Field Set Setup Script
- Single Secure Domain for Shopping and Checkout
- Secure Shopping Domain
- SuiteCommerce Configuration Updates
- SuitePromotions
- View Architecture Changes

The following bundles are available:

- SuiteCommerce Advanced Elbrus: Bundle ID — **165410**
- Site Builder Extensions Elbrus: Bundle ID — **165411**
- Site Builder Extensions Premium Elbrus: Bundle ID — **165412**
- SuiteCommerce Configuration: Bundle ID — **135010**
- Reference Product Lists Records: Bundle ID — **53051**: This bundle is required even if Product Lists records are not leveraged for your implementation. If not installed, the following error is returned when shoppers navigate to the cart: The record type [CUSTOMRECORD_NS_PL_PRODUCTLIST] is invalid.



Important: To take advantage of new features in existing implementations, you must migrate the new code to your existing code base. For details, see the help topic [Migration to New Releases](#).

Custom Field Support

With the Elbrus release of SuiteCommerce Advanced, you can access core and custom field values through the Commerce API, Items API, and SuiteScript. You can configure SuiteCommerce Advanced and customize template files to render information or ask for information stored in these fields on your web store's Shopping, Checkout, and My Account pages.

SuiteCommerce Advanced supports the following kinds of custom field records:

- **Custom Item Fields** – adds custom item field data associated with an item. These fields can render in the Product Details Page (PDP), Checkout, and My Account.
- **Custom Transaction Body Fields** – adds custom transaction fields to the body of a web store transaction (order as a whole). You can also include some field metadata (such as the field label or item options). These fields can render in Checkout and My Account.
- **Custom Transaction Column Fields** – adds a custom transaction field to the column of a transaction record (across multiple transaction lines in an order). These fields can render in the PDP, Cart, Checkout, and My Account.
- **Custom Transaction Item Options** – adds a custom transaction field that applies to a transaction line (one line item within an order). These fields can render in the PDP, Cart, Checkout, and My Account.

Setting up Custom Fields for your SCA site requires NetSuite field set up, web site configuration, and template customization. For detailed information, see the help topic [Commerce Custom Fields](#).

Facets as URL Parameters

When using Faceted Navigation with the Elbrus release of SuiteCommerce Advanced, you have the option to include facets in the URL as either part of the path or as URL parameters. SCA gives you the following configuration options:

- Configure all facets as part of the URL path (default)
- Configure all facets to act as URL parameters
- Configure individual facets as either URL parameters or part of the path.

For more information, see the help topic [Facets as Parameters](#).

Handlebars.js Helper Additions

With the Elbrus release of SuiteCommerce Advanced, the HandlebarsExtras module defines the following additional Handlebars.js helpers:

- `breaklines`
- `ifEquals`

For more information, see *HandlebarsExtras*.

Internet Explorer 8 No Longer Supported

As of the Elbrus release of SuiteCommerce Advanced, implementations no longer support the use of Internet Explorer 8 (IE8) and older. SuiteCommerce Advanced implementations and features prior to the Elbrus release of SuiteCommerce Advanced are not affected by this change.

Discontinued support of IE8 does not mean that your SuiteCommerce Advanced website will crash on IE8 users. However, it does mean that users may experience unexpected behaviors. Test all development for browser compatibility. For more information, see the help topic [Supported Browsers for SuiteCommerce Web Stores](#).

Mastercard 2-Series BIN Range Support

SuiteCommerce Advanced now supports the latest Mastercard 2-Series BIN range. The `PaymentMethods.json` file contains the latest regex string as the default value for the `paymentmethods.regex` configuration property.

For more information, see the help topic [Payment Methods Subtab](#).

Multi-Image Option Enhancements

With the Elbrus release of SuiteCommerce Advanced, you can configure more than one item option ID to determine the images that display in your web store. This capability can more accurately depict items in the product details page (PDP) based on multiple option selections. In previous versions of SuiteCommerce Advanced, you could only configure one item option ID to trigger an image change


in the PDP, regardless of other option selections. With Elbrus release, your web store can now depict images based on multiple selections.

For more information, see the help topic [Setting Up Multiple Images for an Item](#).

Pickup In Store

With the Elbrus release of SuiteCommerce Advanced, the Pickup In Store feature is available. Pickup In Store offers a new order fulfillment choice for NetSuite customers.

When you enable this feature, your customers have the option to pick up their orders at nearby retail locations instead of paying to ship the items.

 **Note:** Added permissions for roles and scripts will be included with the bundle.

Some of the enhancements and options included are as follows:

- Configure your retail locations to allow Pickup In Store.
- Create or modify a customized sales order form to fit Pickup In Store.
- Enable the Store Locator feature to show the nearest location to the customer.
- Set up inventory items to indicate whether they are eligible for pickup at retail locations.
- Specify hours for pickup and give customers a cutoff time for pickup on the same day.
- Set a stock buffer so that an item only appears as available for pickup if you have a specific amount of items in stock at the given location.

For more information [Pickup In Store](#).

Product Details Page Architecture Changes

The product details page provides users with detailed information about a product and lets shoppers add items to their cart. To ensure a positive user experience, the PDP needs to create intuitive interactions and return information from the server quickly and efficiently. The architecture of the PDP is designed to accommodate these challenges.

With the Elbrus release of SCA, changes in the architecture of the PDP provide the following enhancements to the user experience:

- Performance enhancements — This new architecture aligns code with existing SCA architecture, reducing maintenance cost and enhancing extensibility
- Clear distinction between Quick View and Full PDP view
- Structured architecture with the PDP as a form, using the same automatic logic validation as other forms used by SCA
- Faster image loading
- Quicker message feedback for user errors
- Configuration enhancements
- Consistent product images throughout the shopping and checkout applications
- Multiple-image selection enhancements
- Enhanced mobile experience
- Readable URLs

For more information on the PDP architecture changes, see the help topic [Product Details Page Architecture](#).

Modules Affected

The following table explains the changes to the different modules that affect the PDP.



Important: The architectural changes introduced in this release will affect how your site displays and processes information in the Product Details Page, the Cart, and throughout transactions. Make note of the following changes. Some modules include code changes. Others are added or removed.

* = This symbol indicates that the module is new with Elbrus release of SCA.

| Modules Removed | Transformed Into |
|-------------------|--|
| Item Details | Product * Item * ProductDetails * ProductLine * |
| ItemOptionsHelper | Product * |
| ItemsKeyMapping | Item * |
| ItemViews | Transaction.Line.Views * |
| Order | Transaction |



Note: To enhance extensibility, SCA uses two different item option templates, one for the PDP and another for search result (Facets).

Item.KeyMapping.js


As part of the architecture changes to the PDP, the ItemsKeyMapping module has been removed with the Elbrus release of SCA. The logic performs the same function, but now exists in the Item module as item.KeyMapping.js. This file contains mapping definitions of what is returned by the Search API. For example, if you want to set the name of items for data stored in a custom field instead of the default **Display Name** field, you extend the mapping in the Item.KeyMapping file, as opposed to customizing each instance across the entire code base.

Quantity Pricing

With the Elbrus release of SuiteCommerce Advanced, the Quantity Pricing feature is available for NetSuite customers who offer quantity-based price breaks. This feature uses the Search API to return an item's quantity pricing details from NetSuite (in JSON format) and display results in a chart that describes the pricing discounts available based on the quantity requested. This information is collapsed by default and available at the user's request by clicking on a link at various locations throughout your website.

The module for this feature is available by default with the Elbrus release bundle, but you must configure NetSuite to enable quantity pricing capabilities. For instructions on how to enable this feature in NetSuite, see the help topic [Using Quantity Pricing](#). You can disable this feature by editing the `distro.json` file.

For more information on this feature, see the help topic [Quantity Pricing](#).

 **Note:** The Quantity Pricing feature relies on the Search API and is not available with SiteBuilder Extensions.


Quick Order

With the Elbrus release of SuiteCommerce Advanced, the Quick Order feature is available for shoppers who prefer to quickly search for and add multiple items to their cart. This feature is designed for merchants who allow catalog ordering or provide wholesale products on their site. Quick Order assists shoppers who are familiar with a merchant's catalog and wish to shop for items by Stock Keeping Unit (SKU), Part Number, or other known identifier.

Users can click a **Quick Order** link in the header or view their Cart and expanding the **Quick Add** accordion. Users can then quickly search for, add, or remove multiple items to an order and adjust quantities in a list format. This feature offers an alternative to browsing through the web store.

This feature is enabled by default with the Elbrus release bundle, but you must set up Search Fields in NetSuite to ensure proper operation. You can disable this feature by editing the `distro.json` file.

For more information on this feature, see the help topic [Quick Order](#).

 **Note:** The Quick Order feature relies on the Search API and is not available with SiteBuilder Extensions.

Script Debugger Change for Field Set Setup Script

In the SuiteScript debugger tool, there is now an option to select the API Version to run scripts against. When running the Field Set Setup script, you must explicitly select the Version 1.0 API. For more information, see the help topic [Import Field Sets](#).

Single Secure Domain for Shopping and Checkout


SuiteCommerce Advanced now supports a single secure (HTTPS) domain for the checkout and shopping applications. A single domain provides consumers with a seamless buying experience. It removes the delays associated with shopping in one domain and moving to another for checkout.

For more information, see the help topic [Single Secure Domain for Shopping and Checkout](#).

Patch files are available for backporting to the Denali, Mont Blanc, and Vinson implementations of SuiteCommerce Advanced. For more information, see the help topic [Issue Patches](#). Detailed instructions are also available for backporting to pre-Denali implementations of SuiteCommerce Advanced. See the help topic [Secure Shopping Domain \(pre-Denali\)](#).

Secure Shopping Domain

SuiteCommerce Advanced now supports a secure shopping (HTTPS) domain using Secure Sockets Layer (SSL) certificates. Including HTTPS technology in your shopping area assures your customers that their activities on your site are secure. Using HTTPS can also increase site traffic since search engines tend to rank secure sites higher than non-secure sites.

 **Important:** When using HTTPS, you **must** install the Elbrus release of SuiteCommerce Advanced in a **NetSuite 2017.1** account. If you are working with an older release of SCA, you must also install the patch files for using HTTPS. See the help topic [Issue Patches](#).

You can either set up a new secure shopping domain or directly switch from an existing non-secure shopping domain to a secure shopping domain. For more details on this feature, see the help topic [Secure Shopping Domains](#).

SuiteCommerce Configuration Updates

With the Elbrus release of SuiteCommerce Advanced, the following configuration properties have been added or updated. Refer to the appropriate section for more information.

| Property Label (SuiteCommerce Configuration Record) | Property ID | More Information | Property Location (SuiteCommerce Configuration Record) Tab > Subtab |
|---|---|---|---|
| Facets as URL Parameters | facetsAsUrlParameters | Facets as URL Parameters | Shopping Catalog > Facets |
| Facets > Is URL Parameter? | facets.isParameter | Facets | |
| Facets > Max | facets.max | | |
| Allow Adding More Than One Promo Code | promocodes.allowMultiples | Allow Adding More Than One Promo Code | Shopping Catalog |
| Product Details Information | productDetailsInformation | Product Details Information Subtab | Shopping Catalog > Product Details Information (New Subtab) |
| Product Details Information > Name | productDetailsInformation.name | | |
| Product Details Information > ID | productDetailsInformation.contentFromKey | | |
| Product Details Information > ItemProp | productDetailsInformation.itemprop | | |
| Multi-Image Option | productline.multiImageOption | Multi-Image Option Subtab | Shopping Catalog > Multi-Image Option (Relocated Subtab) |
| Show Only the Items Listed in : Item Options and Custom Transaction Fields | ItemOptions.showOnlyTheListedOptions | Item Options Subtab | Shopping Catalog > Item Options (New Location) |
| Item Options and Custom Transaction Column Fields > URL Parameter Name | ItemOptions.optionsConfiguration.urlParameterName | | |
| Item Options and Custom Transaction Column Fields > Sort Index | ItemOptions.optionsConfiguration.index | | |
| Item Options and Custom Transaction Column Fields > Show Option in Item Lists | ItemOptions.optionsConfiguration.showSelectorInList | | |
| Item Options and Custom Transaction Column Fields > Facet Cell Template | ItemOptions.optionsConfiguration.templateFacetCell | | |
| Default Selected Templates by Item Option Type | ItemOptions.defaultTemplates.selectedByType | | |

| Property Label (SuiteCommerce Configuration Record) | Property ID | More Information | Property Location (SuiteCommerce Configuration Record) Tab > Subtab |
|--|---|--|---|
| Option Type | ItemOptions.defaultTemplates.selectedByType.type | Type.type | |
| Template Name | ItemOptions.defaultTemplates.selectedByType.template | Type.template | |
| Default Selector Templates by Item Option Type | ItemOptions.defaultTemplates.selectorByType | Default Selector Templates by Item Option Type | |
| Option Type | ItemOptions.defaultTemplates.selectorByType.type | Type.type | |
| Template Name | ItemOptions.defaultTemplates.selectorByType.template | Type.template | |
| Default Facet Cell Templates by Item Option Type | ItemOptions.defaultTemplates.facetCellByType | Default Facet Cell Templates by Item Option Type | |
| Option Type | ItemOptions.defaultTemplates.facetCellByType.type | Type.type | |
| Template Name | ItemOptions.defaultTemplates.facetCellByType.template | Type.template | |
| Color Palettes (Was Facet Color Palettes) | layout.ColorPalette | Color Palettes Subtab | Layout Tab > Color Palettes (New Location) |
| Color Palettes > Palette ID | layout.ColorPalette.paletteld | | |
| Color Palettes > Color Name | layout.ColorPalette.colorName | | |
| Color Palettes > > Value | layout.ColorPalette.colorValue | | |
| Color Palettes > Source | layout.ColorPalette.imgsrc | | |
| Color Palettes > Height | layout.ColorPalette.imgheight | | |
| Color Palettes > Width | layout.ColorPalette.imgwidth | | |
| Light Colors | layout.lightColors | Light Colors Subtab | Shopping Catalog > Light Colors (New Location) |
| Enable Pickup in Store | checkoutApp.isPickupInStoreEnabled | Enable Pickup In Store | Checkout Tab |
| Pickup In Store Sales Order Custom Form ID | checkoutApp.pickupInStoreSalesOrderCustomFormID | Enable Pickup In Store Sales Order Custom Form ID | |
| Remove PayPal Address | removePaypalAddress | Remove PayPal Addresses | |
| Sales Order | customFields.salesorder | Custom Fields Subtab | Advanced > Custom Fields (New Subtab) |
| Disable ESC Key to Login | cms.escToLoginDisabled | Disable ESC Key to Login | Integrations > Site Management Tools |

SuitePromotions

With the Elbrus release of SuiteCommerce Advanced, the SuitePromotions feature is available. With SuitePromotions, your customers can add multiple eligible promo codes to their order on your

SuiteCommerce Advanced website. Customers can see the applied promotions in their cart and remove them if needed.

SuitePromotions include item promotions, order promotions, and shipping promotions. You can configure each promotion to be completely exclusive (not stackable) or you can prevent stacking only for the same promotion types. For example, you allow stacking of a shipping promotion and an item promotion, and you prevent customers from stacking two different shipping promotions.

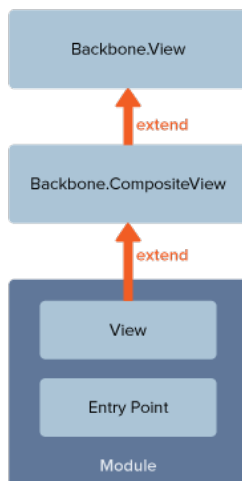
See the help topic [Promotions](#) for more details.

View Architecture Changes

With the Elbrus release of SuiteCommerce Advanced, all Backbone.Views are composite views by default.

New Architecture

With the Elbrus release of SuiteCommerce Advanced, all Backbone.Views are now Backbone.CompositeViews by default. Composite views are simply Views that contain Child Views. These Child Views further modularize the application, so views can be used in multiple contexts. For example, some modules define Child Views that are extensions of Backbone.View. As a consequence, those Child Views are also now Backbone.CompositeViews. This ensures that data displays in a consistent way across the application.



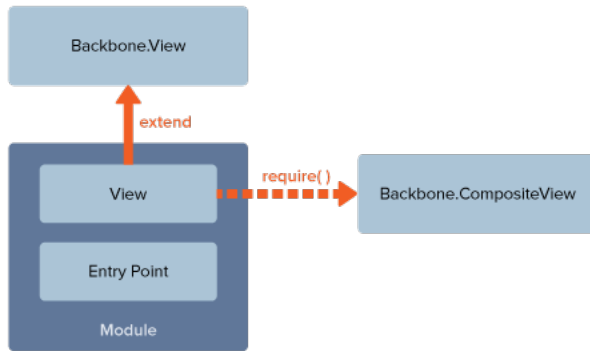
With the Elbrus release of SCA, Views no longer need to declare themselves as composite views using the `require()` method.

See the help topic [View Architecture](#) for detailed information.

To make this change backward compatible with earlier releases of SCA, Backbone.CompositeView.js includes a `CompositeView.add()` method as a no operation (noop) method. This prevents any errors when calling it from any custom code in any implementations prior to Elbrus release.

Previous Architecture

In previous releases of SCA, all views extended Backbone.View and declared themselves as a composite views using the `require()` method.



Vinson Release of SuiteCommerce Advanced

i Applies to: SuiteCommerce Advanced | Vinson

Welcome to the Vinson release of SuiteCommerce Advanced! This version includes the following enhancements. Many of the features and enhancements in this release have dependencies on the 2016.2 release of NetSuite. For details, see the [NetSuite 2018.1 Release Notes](#).

- Alternative Payment Methods
- Application Performance Management (APM) Changes
- Category Integration
- Checkout Usability Improvements
- Configuration Changes
- Device Fingerprinting for Transactions
- Field Set Setup Script
- Header Improvements
- Newsletter
- Quotes Enhancements
- Save Credit Card Information
- SCIS Integration
- Services Architecture Changes
- Site Access Restriction
- Store Locator

The following bundles are available:

- SuiteCommerce Advanced Vinson: Bundle ID — **134179**
- Site Builder Extensions Vinson: Bundle ID — **134180**
- Site Builder Extensions Premium Vinson: Bundle ID — **134181**
- SuiteCommerce Configuration: Bundle ID — **135010**



Important: To take advantage of new features in existing implementations, you must migrate the new code to your existing code base. For details, see the help topic [Migration to New Releases](#).

Alternative Payment Methods

This release of SuiteCommerce Advanced includes an update to Alternative Payment Methods. Previously, Alternative Payment Methods were available to users only when making a payment during a standard checkout. Now you can use alternative payment methods in the following situations:

- When paying for sales orders converted from a Request for Quote
- When making payments to an invoice

These improvements do not require any configuration or setup beyond that required for the overall feature. For more information on this feature, see the help topic [Alternative Payment Methods](#).

Application Performance Management (APM) Changes

With this release of SuiteCommerce Advanced the data stored by the Application Performance Management (previously known as Real User Management — or RUM) has been updated. For detailed information on what data is stored, see the help topic [Data Stored by APM](#). For more information on the overall APM feature, see the help topic [Application Performance Management](#).

A diff in the form of a .patch file for backporting APM enhancements to Mont Blanc is also available. For details on implementing this patch, see the help topic [Issue Patches](#).

Category Integration

With this release of SuiteCommerce Advanced you can integrate NetSuite Categories into your SCA website.

With Categories enabled, you can create a hierarchical structure of product categories, subcategories, and products to organize your store catalog. Each site can have one catalog, with categories assigned to it. This structure lets you present items to shoppers in a structured and organized fashion.

To enable Categories for your SCA site, some setup and configuration in NetSuite is required. For more information on implementing Categories into your SCA site, see the help topic [Commerce Categories](#). For information on configuration properties for this feature, see the help topic [Categories Subtab](#).

Checkout Usability Improvements

This release of SuiteCommerce Advanced includes an update to the Checkout experience and includes the following enhancements:

- [Search by Purchase Order](#)
- [Guest User Checkout Fields](#)
- [Save Credit Card Information](#)

Search by Purchase Order

You can now set your SuiteCommerce Advanced site preferences (Setup > SuiteCommerce Advanced > Set Up Web Site) to allow users to add Purchase Order (PO) numbers to the Payment Information page of your web store. The **Display Purchase Order Field on Payment Info Page** setting displays or hides a text field where customers can enter PO numbers. This PO number automatically appears on sales orders generated from your web store. POs are also searchable by PO number.

This setting has been available for Site Builder and is now available for SCA sites. For information on these preferences, see the help topic [Payments Page](#).

Guest User Checkout Fields

You can now configure your site to provide **Name** and **Email** input fields for guest users. The new `forms.loginAsGuest.showName` property displays or hides first and last name input fields when a user registers as a guest. Likewise, the new `forms.loginAsGuest.showEmail` property provides an email address input field. If enabled, these fields become required for guest users visiting your site.

You can also choose to make these fields automatically populate throughout the guest's checkout experience. The new `autoPopulateNameAndEmail` property enables or disables auto-population of the guest's name and email in any forms throughout checkout. If this property is enabled, `forms.loginAsGuest.showName` and `forms.loginAsGuest.showEmail` properties must be enabled (as applicable) to auto-populate.

Configure these properties using the new SuiteCommerce Configuration tool. For descriptions of configuration properties, see the help topic [Forms Subtab](#).

For details on the new Configuration Tool, see the help topic [Configure Properties](#).

Save Credit Card Information

You can now set your SuiteCommerce Advanced site preferences (Setup > SuiteCommerce Advanced > Set Up Web Site) to:

- **Display 'Save My Credit Card Info' Field** – enables users to choose whether or not to save credit card information.
- **Save Credit Card Info by Default** – saves credit card information in a Customer record by default.

If you set **Display 'Save My Credit Card Info' Field** to true, your site displays an option during checkout, and users can choose to save their card information for their future use. If you enable both preferences, users can still clear this option and choose not to save their card information. These settings have been available for Site Builder and are now available for SCA sites.

For information on these preferences, see the help topic [Payments Page](#).

Configuration Changes

This release of SuiteCommerce Advanced includes significant changes to how you configure your SuiteCommerce Advanced website. Configuration is no longer contained in JavaScript configuration files as part of the application source code. These properties are now stored in JSON configuration files and set using a new SuiteCommerce Configuration record in NetSuite.

Pre-Vinson

In previous versions of SuiteCommerce Advanced, configuration properties were stored in the following locations:

- Configuration.js (backend)
- SC.Configuration (frontend)
- SC.Checkout.Configuration (frontend)
- SC.MyAccount.Configuration (frontend)
- SC.Shopping.Configuration (frontend)

Vinson

With the Vinson release of SuiteCommerce Advanced, configuration properties are stored in JSON files as part of the SuiteCommerce Advanced source code. Any module that defines configuration

properties contains a subdirectory called Configuration. This directory contains the JSON configuration files associated with the module. The JSON files contain the schemas used to manage configuration settings plus default property values for the associated domain. Any default configuration settings come with the Vinson source code, but you must create new JSON configuration files for your custom modules.

With the Vinson release of SuiteCommerce Advanced there is a new SuiteCommerce Configuration bundle (bundle ID: 135010). After installation, you can edit existing properties using the SuiteCommerce Configuration record at Setup > SuiteCommerce Advanced > Configuration.

For a description of Configuration for both pre-Vinson and Vinson implementations, see the help topic [Configure Properties](#).

For detailed descriptions of configurable properties (regardless of SCA implementation), see the help topic [Configuration Properties Reference](#).

Device Fingerprinting for Transactions

For this release of SuiteCommerce Advanced, device fingerprinting for online orders as part of the advanced fraud management features available with CyberSource Decision Manager is supported. For more information, see the help topic [Device Fingerprinting](#).

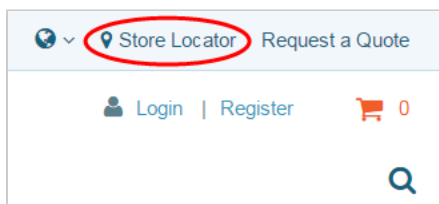
Field Set Setup Script

This release of SuiteCommerce Advanced includes a new field set named Items Searcher. This field set is required for use with [Quotes](#). See the help topic [Import Field Sets](#) for details.

Header Improvements

This release of SuiteCommerce Advanced includes an update to the code and design of the website header. These changes alter the appearance and behavior of the website header.

These changes include the addition of the Store Locator feature. The header now loads with a Store Locator link and pin icon (mobile devices display only the pin icon). When a user loads the Store Locator page, the site prompts them to enable geolocation. Other capabilities within the header function in the same manner as the previous release.



Newsletter

This release of SuiteCommerce Advanced includes a Newsletter feature to manage email subscriptions to an established email marketing campaign. With this feature:

- Your site displays a newsletter sign up link in the site's footer.
- The email address entered is validated against any existing NetSuite Customer or Lead records with a matching address.
- A Soft Opt-In global Subscription Status to new subscribers is created.
- Your site displays a confirmation message after the shopper signs up.

- For each newsletter subscriber, the NetSuite Customer record for existing customers is updated or a new Lead record for new shoppers is created.

Note: This is an email subscription opt-in feature only. It is not a marketing campaign or content management tool. After a user opts in, it is the merchant's responsibility to manage subscriptions for each Lead or Customer record. For detailed information on creating and managing an email marketing campaign using NetSuite, see the help topic [Email Marketing Campaigns](#).

To enable the Newsletter feature to work, do the following in your NetSuite account:

- Enable subscriptions for your account
- Create a new Employee Center role
- Set up Elevated Permissions
- Configure feature properties

For detailed information about this feature, see the help topic [Newsletter](#).

Quotes Enhancements

With this release of SuiteCommerce Advanced, users with permissions to create quotes now see an **Add to Quote** button on product details pages. After pressing this button, SCA sites display a message informing the user that the item and specified quantities are added to their quote. These items appear in the user's request for quote. If a user does not have permissions to create quote requests, this button does not appear.

Note: If a user adds an item using this button and requests a quantity lower than the minimum required, the item is added to the quote request, but at the minimum quantity required for purchase.

For detailed information, see the help topic [Quotes](#).

This enhancement requires a new field set called Items Searcher. The Items Searcher field set is included in the field set setup script. For details, see the help topic [Import Field Sets](#).

SCIS Integration

This release of SuiteCommerce Advanced includes an update to the SCIS Integration feature.

Using the previous release of this feature, shoppers could view all of their purchases in one list on their Purchase History page, differentiating between the Online or In Store purchases using the **Origin** column.

With the Vinson release of SuiteCommerce Advanced, shoppers can also:

- View and print a PDF of an individual purchase from the Purchase Details page.
- Determine if a purchase is associated with an existing sales quote.

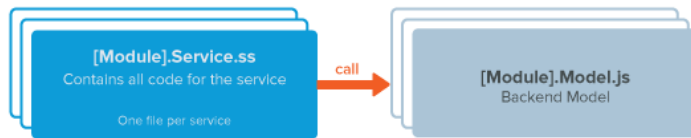
These improvements do not require any configuration or setup beyond that required for the overall feature. For more information on this feature, see the help topic [SuiteCommerce InStore Integration](#).

Services Architecture Changes

This release of SuiteCommerce Advanced includes a significant change to how SCA organizes and manages services.

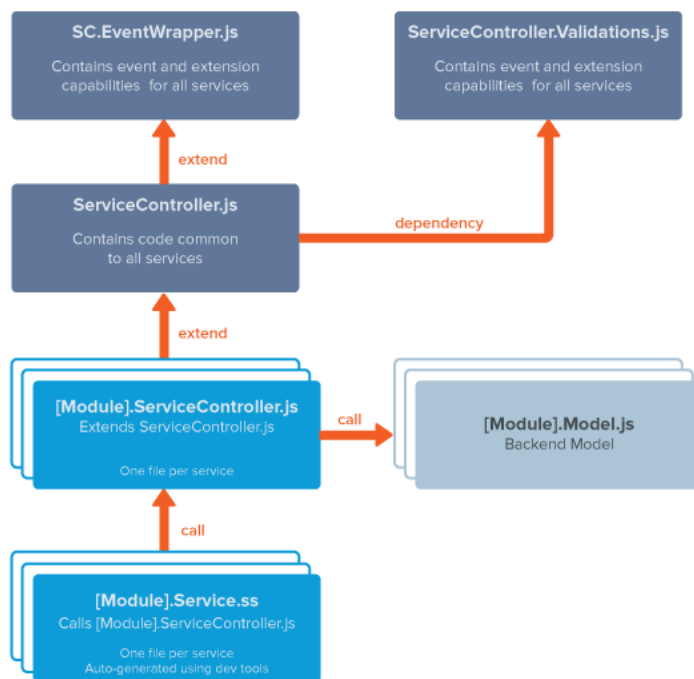
Previous Releases of SCA

In previous releases of SCA, each service required a .ss file to handle data transactions with NetSuite. Each .ss file handled HTTP requests sent by the frontend application to access data in NetSuite. Each .ss file maintained its own logic, but also included code common to all services. One .ss file existed per service.



Vinson Release and Later

The Vinson release of SuiteCommerce Advanced relies on more files to achieve the same result. This new architecture may appear more complex, but it helps maintain services in a more structured way than previous versions of SCA.



SC.EventWrapper.js – contains centralized, extensible Before and After Event listening capabilities plus extension capabilities for use by all service methods.

ServiceController.js – maintains extensible code common to all services. The code that used to be maintained across all .ss files is now maintained here. This new ServiceController module extends SC.EventWrapper.js and references validation methods declared in ServiceController.Validations.js.

[Module].ServiceController.js – processes requests for a specific service. It does this by extending ServiceController.js. Code is maintained in one location and made available to all services. Each service-specific controller can call a backend model, which communicates with NetSuite as in previous releases.

[Module].Service.ss – handles communication between the frontend SuiteCommerce application and the associated service's [Module].ServiceController. Each service still requires one service (.ss file) to communicate with the frontend, but this file is now automatically generated using the dev tools and simply calls its related, service-specific controller.

ServiceController.Validations.js – contains the validation methods for requests common to all services.

Note: The ns.package.json file of the module associated with the service signals the dev tools to auto-generate your [Module].Service.ss file and associates it with a specific service controller.

For detailed information on how to build custom services, see the help topic [Create a Service to Handle HTTP Requests](#).

Site Access Restriction

This release of SuiteCommerce Advanced includes changes to how you set up website access restrictions. If implementing Vinson release or later, you enable these features in the Web Site Setup in NetSuite. In the previous release of SCA, this required changing the backend Configuration.js file.

For more information on setting up restrictions to your entire website or restricting access to just pricing information, see the help topic [Website Restriction](#). For information on setting up

Store Locator

This release of SuiteCommerce Advanced includes a Store Locator feature. Store Locator lets shoppers search for physical store locations on your site. Any search returns results within proximity of the shopper's location or specified address. A link to the Store Locator page appears in the Header, along with Pin icon.

With this feature enabled, shoppers can:

- View store locations as a list and on an interactive map
- View stores near their current location
- Search for stores near a specific address, city, or zip code
- Choose from a list of results to view information about a location, including address and store hours
- View a complete list of all stores
- Get directions to a particular store

To implement the Store Locator feature, you must use NetSuite to create a custom Customer Center Role and set up Elevated Permissions for the associated backend service (.ss file). You must also configure feature properties. For detailed information on setting up this feature, see the help topic [Store Locator](#).

Note: The Vinson release bundle of SuiteCommerce Advanced uses Google Maps as the default map engine for Store Locator.

Mont Blanc Release of SuiteCommerce Advanced

Applies to: SuiteCommerce Advanced | Mont Blanc

Welcome to the Mont Blanc release of SuiteCommerce Advanced! This version includes the following enhancements:

- [Change in Supported Version of Node.js](#)
- [Quotes](#)
- [Site Access Restriction](#)
- [SCIS Integration](#)
- [Alternative Payment Methods](#)
- [Cancel Order](#)
- [Real User Monitoring \(RUM\) Integration](#)
- [Google Tag Management](#)
- [My Account Changes](#)
- [URLs for Major Payment Methods](#)
- [Issue Fixes](#)
- [Updated and New Modules](#)

The following bundles are available:

- SuiteCommerce Advanced MontBlanc: Bundle ID — **107016**
- Site Builder Extensions MontBlanc: Bundle ID — **107017**
- Site Builder Extensions Premium MontBlanc: Bundle ID — **107018**



Important: To take advantage of new features in existing implementations, you must migrate the new code to your existing code base. For details, see the help topic [Migration to New Releases](#).

Click the following link to download a .zip file containing .patch files for the Mont Blanc release:

[Mont_Blanc_2.0_Patch_Files.zip](#)

These .patch files contain the complete diffs between this release and the Denali R2 release.

Use a tool such as Crucible to open these files and graphically visualize all additions, deletions, and modifications. You can also view .patch file in a text editor. They are displayed with appropriate syntax highlighting when the syntax is set to diff.

Change in Supported Version of Node.js

With this release of SuiteCommerce Advanced, the developer tools were modified to support Node.js version 4.2.6 in addition to support for version 0.12.x. See *Using the Developer Tools* for more information.

Quotes

With this release of SuiteCommerce Advanced, users can now create quotes from your website. With this feature enabled, a **Request a Quote** link appears by default to all users (logged in or not) in your site's Shopping and My Account header. These links direct users to a page where they can search for and add items, enter shipping information, and add optional user comments to build a quote request.

Users can eventually make purchases from a quote, but only after the quote reaches a certain status in the corresponding Estimate record. You configure the status that triggers this functionality by editing code in the Configuration.js file. You can also configure the time (in days) that all quotes expire after initial submittal.

To enable this feature you must perform the following tasks:

- Install the latest SCA and Reference Product List Records bundles.
- Edit the Customer Center Role permissions.
- Set up Search Fields.
- Customize the Standard Quote form to include shipping information.
- Configure the Configuration.js to set the Customer Status Internal ID that will allow a quote to be purchased.
- Configure the Configuration.js to define the invoice form used to create a purchase from a quote.

Note: The Quotes feature will function without the Reference Product List Records bundle installed. However, items built in a product list will be non-persistent and will be lost if the user leaves the site before submitting the quote. Installing the Reference Product List Records bundle ensures that product list items are persistent. Shipping information and user comments are non-persistent, regardless of bundle installation.

Optional customizations include:

- Disabling the Create a Quote functionality, allowing users to view quotes and make purchases, without directly creating quotes
- Customizing sales representative information

For detailed information on enabling the Quotes feature, see the help topic [Quotes](#).

Purchases From Quotes

After a user submits a quote request, any web site links to Review or Purchasing pages are disabled. This prevents users from making a purchase from that quote until the merchant allows it.

In NetSuite, the Estimate record's **Status** field lists all available Customer Statuses for the quote. When the merchant manually changes this field to the Customer Status whose Internal ID matches the ID you set in the Configuration.js file, all web site links to Review or Purchasing pages are enabled. The user can then make a purchase.

The Customer Status List at Setup > Sales > Customer Statuses displays the Customer Statuses configured for your account and their associated Internal IDs. You declare this Internal ID using the `purchase_ready_status_id` property in the Configuration.js file.

All quotes use the same configurable expiration, in days. After this time expires, the quote can no longer become a purchase. You set this expiration using the `days_to_expire` property in the Configuration.js file.

In the following example, quotes created on your site expire 14 days after the initial RFQ. Assuming a quote has not expired, it becomes available for purchase when its associated Estimate record reaches the **Qualified** Customer Status (Internal ID = 7).

```
, quote: {
  days_to_expire: 14
, purchase_ready_status_id: '7'
}
```

You must also configure what transaction form is used to create the purchase. You set the Internal ID of the chosen transaction form using the `invoice_form_id` property in the Configuration.js file.

```
, quote_to_salesorder_wizard: {
```



```
invoice_form_id: '89'
}
```

Disable Create a Quote Functionality

After setting up your account for quotes, you can configure SuiteCommerce Advanced to disable a user's ability to create a quote from your site. Users can view quotes and make purchases, but they can not create a quote directly using your site.

To do this, delete all lines within the `distro.json` file that contain either of the following references:

```
RequestQuoteAccessPoints
RequestQuoteWizard
```

See the help topic [Disable Quotes](#) for detailed information.

Customize Sales Representative Information

The Quotes feature displays Sales Representative information on your site's Quote Details page. You can customize your web site to display the Sales Representative as configured in NetSuite. If your NetSuite account does not have Sales Representative information set up, or you do not want to display the information, you can add a customized message instead.

What displays on your site depends on the **SalesRep** field on the quote's Estimate record in NetSuite.

- If the **SalesRep** field for the quote lists a sales representative whose contact information (phone number and e-mail address) is set up in NetSuite, that information displays on the Quote Details page.
- If the **SalesRep** field lists a sales representative whose contact information is not set up in NetSuite, your site defaults to display customizable contact information found in the `SC.MyAccount.Configuration.js` file.
- If the **SalesRep** field is blank, your site displays two disclaimer messages, one under the Order Summary and one at the bottom of the page. You can customize the text of these messages in the `SC.MyAccount.Configuration.js` file.

See the help topic [Customize Sales Representative Information](#) for instructions.

More Information

Before setting up the Quotes feature on your SuiteCommerce Advanced web site, understand the following information:

- This feature does not function in Site Builder Basic. Site Builder Premium supports creating quotes and making purchases.
- This feature does not support subsidiaries.
- Users can add items to a quote, regardless of the item's out of stock behavior.
- Users cannot change the delivery method of a quote after the quote becomes a sales order. The site's default delivery method is used.
- Handling costs are not displayed in the summary total of the quote, but they are added to the order total after the quote becomes a purchase.
- Parent matrix items cannot be added to a quote, only specific child items.
- The Multiple Ship To feature is not available when making a purchase from a quote.

- Payment methods through external gateways are not available when making purchase from a quote.
- Gift Certificates cannot be added to quotes.
- When a user submits a quote, SuiteCommerce Advanced does not automatically send an email notification of the submission by default. This action can be set up in NetSuite as a Workflow Action (see the help topic [Workflow Elements](#)) or as a User Event Script (see the help topic [User Event Scripts](#)).

Site Access Restriction

With this release of SuiteCommerce Advanced, you can now restrict access to your web site in one of two ways:

- Protect your entire website so that only registered customers can log in to view your web site.
- Protect pricing information only. Visitors can view your site, but they must register and log in to view pricing information.

These features are disabled by default and require set up in NetSuite and some minor customization of the Configuration.js file to function.



Warning: The Search API is public and contains all the information exposed by the fieldset being used. Enabling password protection restricts users from accessing some or all of your site. However, a user who knows a URL that calls the Search API could access some of this information.

Restrict Access to Your Entire Site

You can now restrict your entire site to only logged in users. If this feature is enabled, visitors must register and log in to see the contents of your site. The site redirects unauthenticated visitors to a Login page with a link to a Registration form where they must register on your site to gain access. After a logged in user does not interact with the site for 30 minutes, their session ends. The first action they perform that calls a service redirects them to a Login page.

To enable this feature, you must perform some minor NetSuite set up and set the `passwordProtectedSite` property to true in the Configuration.js file. For detailed information on restricting all access to your site, see the help topic [Restrict Access to Your Entire Site](#)

Restrict Access to Pricing Information

You can now restrict pricing information to only registered users. If enabled, visitors have access to your site but they must log in to view prices and purchase products or services. This feature lets you hide prices from competitors or wholesalers and encourages users to register on your site.

With this feature enabled and a user is not logged in, SuiteCommerce Advanced replaces prices with a log in message and link. The user can follow this link to access a Login/Registration page where they must register on your site to view prices or make purchases. After a logged in user does not interact with the site for 30 minutes, their session ends. The first action they perform that calls a service redirects them to a Login page.



Note: Restricting access to your entire site overrides this feature.

To enable this feature, you must set the `loginToSeePrices` property to true in the Configuration.js file. For detailed information on restricting all access to your site, see the help topic [Restrict Access to Pricing Information](#)

SCIS Integration

With this release of SuiteCommerce Advanced, integration with the SuiteCommerce InStore platform is now possible. SuiteCommerce InStore is a web-based point-of-sale application that uses an in-store, tablet-delivered commerce portal connected to your NetSuite backend.

With this feature enabled, you can incorporate your SuiteCommerce InStore transactions into your SuiteCommerce Advanced web site order management and billing components. With this feature, your shoppers can now:

- View details of all online (SCA) and in-store (SCIS) purchases and returns in their My Account section of your web site.
- Distinguish between online and in-store purchases.
- Archive in-store purchase information (receipts) as a PDF.
- Initiate an online return of a product purchased in store.
- Reorder items online for products purchased in store.
- View cash sales from all channels in their Transaction History page.
- Make online payments for in store purchases that were settled on terms (invoices).

SuiteCommerce purchases originate from either online (SCA) or in-store (SCIS) sales. In the backend, NetSuite manages these different purchase types using different types of Transaction records. For example, an order placed online through your SCA site creates a Sales Order transaction record. Another purchase placed in your store using SCIS may require the Cash Sale or the Invoice record, depending on the transaction type.

The SuiteCommerce InStore Integration feature displays all of a user's transactions in several My Account pages, marking each as Online or In Store, regardless of the Transaction record used. This creates a seamless omni-channel experience for your customers.



Important: Users will see an **Origin** column in their Purchase History and Transaction History pages where the Status column used to be. This **Origin** column designates the purchases's origin as Online or In Store.

The following table lists the purchase origin (as displayed on your web site) and its associated SuiteCommerce application, transaction type, and NetSuite record.

| Origin | Application | Transaction Type | Transaction Record in NetSuite |
|----------|---|---|--|
| Online | SuiteCommerce Advanced | Online purchase Shipped to a single address | Sales Order |
| | SuiteCommerce Advanced | Online purchase Shipped to multiple addresses | Sales Order Multi-Ship To |
| In Store | NetSuite | Added directly into NetSuite | Sales Order |
| | SuiteCommerce InStore NetSuite Point of Sale | In-store purchase Single payment method | Cash Sale |
| | SuiteCommerce InStore NetSuite Point of Sale | In-store purchase Multiple payment methods | Invoice |
| | SuiteCommerce InStore NetSuite Point of Sale | In-store purchase Mixed delivery Single payment method | Sales Order Multi-Ship To Cash Sale |
| | SuiteCommerce InStore NetSuite Point of Sale | In-store purchase Mixed delivery Multiple payment methods | Sales Order Multi-Ship To Invoice |

To implement SCIS Integration, you must have the latest SuiteCommerce InStore and SuiteCommerce Advanced bundles installed in your account. You must also set the `isSCISIntegrationEnabled` property to true in the Configuration.js file. For detailed information on this feature, see the help topic [SuiteCommerce InStore Integration](#).

Alternative Payment Methods

With this release of SuiteCommerce Advanced, setting up a third-party payment option is now available. With this feature enabled, users have the option to check out using third party companies offering payment processing services, such as Google Wallet. This selection redirects users to a third-party web site for payment. Users are then prompted to leave your site and enter the third-party payment site for authentication.

This feature does not require any code customization. However, you must set up your NetSuite account for external payments. See the help topic [Alternative Payment Methods](#) for detailed information.



Important: To create a payment gateway to a third-party payment service provider, an administrator must install the correct SuiteApp that integrates the payment method into your account. You must install a SuiteApp developed and distributed by a SuiteCloud Developer Network (SDN) Partner. See *Partner SuiteApps* for more information about SuiteApps developed and supported by SDN Partners.

If no payment service provider offers a SuiteApp to meet your needs, you can create your own Payment Processing implementation. To build, test, and bundle a Payment Processing plug-in implementation, you must have the appropriate authorization from NetSuite. You must also be a member of the SuitePayments program. To obtain this authorization and to be considered for the program, see [SuiteApp.com](#).



Note: With Mont Blanc release of SuiteCommerce Advanced, the Order Details page no longer provides information on the shipping method, shipping address, billing information, or payment type.

Cancel Order

With this release of SuiteCommerce Advanced, users can now cancel a purchase that has been submitted but not processed. Users have the option of clicking a **Cancel Purchase** button in the Purchase Details page.

Visibility of this button is dependent on the purchase fulfillment status. If the Sales Order record's **Status** field lists anything other than **Pending Approval**, the purchase cannot be canceled, and the **Cancel Purchase** button does not appear. This button appears when the Sales Order record lists any other status.

The feature does not require any setup or code customization.

Real User Monitoring (RUM) Integration

This release of SuiteCommerce Advanced provides integration for the Real User Monitoring (RUM) feature. Real User Monitoring enables you to extract and store metrics about how users interact with your website. This information is stored in a backend database in NetSuite.

You can analyze this data using tools available in an upcoming release of the Application Performance Management (APM) SuiteApp.

To implement this feature, SuiteCommerce Advanced uses the Sensor utility module. When Real User Monitoring is enabled, this module is automatically loaded when the application starts.

For detailed information on this feature, see the help topic [Application Performance Management](#).

Google Tag Management

This release of SuiteCommerce Advanced includes the new Google Tag Manager module. GTM enables you to manage site tracking and analysis tags and code snippets on your site. This eliminates the requirement to maintain individual pieces of code for things such as Google Universal Analytics or Adwords because everything is managed through GTM. GTM is vendor neutral and enables you to manage not only Google tags, but also tags from third parties such as Bing.

The [GTM Container Generator](#) aids you in implementation of GTM by using your tracking account information to generate a JSON file. This JSON file is imported into your GTM account. For more information on implementing and using Google Tag Manager, see the help topic [Google Tag Manager](#).

My Account Changes

With this release of SuiteCommerce Advanced, new labels on your web site's My Account pages are displayed. In the My Account overview page, the label, **Orders**, in the My Account accordion list is replaced with **Purchases**. Also, the **My Orders** page is now titled **My Purchases**.

This is a global change and is not configurable.

URLs for Major Payment Methods

With this release, SuiteCommerce Advanced changes the way it locates Payment Method logos to display during checkout. Therefore, you must make minor changes to your NetSuite Payment Method records to display the bundled logos that were used in the previous release. These changes are described in [Previous Approach](#).

Previous Approach

Previously, SuiteCommerce Advanced used URLs configured in the SC.Configuration.js file to locate Payment Method logos. The `creditCardIcons` object superseded any Flags set up in the Payment Method record in NetSuite and linked to logos that were included in the SuiteCommerce Advanced bundle.

Previous Code

```
creditCardIcons: {
  'VISA': 'img/visa.png'
  , 'Discover': 'img/discover.png'
  , 'Master Card': 'img/master.png'
  , 'Maestro': 'img/maestro.png'
  , 'American Express': 'img/american.png'
}
```

Updated Approach

After migrating to the Mont Blanc release of SuiteCommerce Advanced, the SC.Configuration.js file no longer contains the `creditCardIcons` object. The SC.Configuration.js file now uses the `paymentMethods` array to point to the Payment Method record in NetSuite using the Payment Method ID.

To link to the bundled logos, enter the following links in the URL column on the **Payment Visuals** subtab of the associated Payment Method record in NetSuite. See the help topic [Creating a Payment Method](#) for setup instructions.

Note: To maintain design continuity on your SuiteCommerce Advanced site, we recommend using the following URLs and the logos bundled with this and previous releases. These differ from the defaults as listed in [Creating a Payment Method](#). Ensure that each image is uploaded to your NetSuite File Cabinet in the appropriate locations.

| Payment Method | URL |
|------------------|------------------|
| Visa | img/visa.png |
| Discover | img/discover.png |
| Master Card | img/master.png |
| Maestro | img/maestro.png |
| American Express | img/american.png |

Updated Code

```
, paymentmethods: [
  {
    key: '5,5,1555641112' //'VISA'
    , regex: /^4[0-9]{12}(?:[0-9]{3})?$/
  }
  , {
    key: '4,5,1555641112' //'Master Card'
    , regex: /^5[1-5][0-9]{14}$/
  }
  , {
    key: '6,5,1555641112' //'American Express'
    , regex: /^3[47][0-9]{13}$/
  }
  , {
    key: '3,5,1555641112' // 'Discover'
    , regex: /^6(?:011|5[0-9]{2})[0-9]{12}$/
  }
  , {
    key: '16,5,1555641112' // 'Maestro'
    , regex: /^(?:5[0678]\d\d|6304|6390|67\d\d)\d{8,15}$/
  }
  , {
    key: '17,3,1555641112' // External
    , description: 'This company allows both private individuals and businesses to accept payments over the Internet'
  }
]
```

Issue Fixes

This release includes many issue fixes including:

Product Review Records Imported via CSV are Not Displaying on Product Page

Previously, imported product review records from a comma-separated variable (CSV) file did not display on the Product Page.

To fix this problem, NetSuite updated code in the backend and released a Reference Product Review Records bundle update.

Bundle Update

NetSuite replaced the following code in the `NS_PR_SS_User_Event.js` file within the Reference Product Review Records bundle.

Previous Code:

```
function beforeSubmit (action)
{
    'use strict';

    var user = nlapiGetUser()
    ,   new_review = nlapiGetNewRecord();
    // false when deleting
    if (new_review)
    {
        var item_id = new_review.getFieldValue('custrecord_ns_prr_item_id');
        // if the record is beeing created
        // and the user is logged in
        if (action.toString() === 'create' && user)
        {
            setReviewUser(new_review,user);
            verifyItemPurchase(new_review, user, item_id);
        }

        // if the review is not assigned to
        // the item record
        if (!new_review.getFieldValue('custrecord_ns_prr_item') && item_id)
        {
            setReviewItem(new_review, item_id);
        }
    }
}
```

Updated Code:

```
function beforeSubmit (action)
{
    'use strict';

    var new_review = nlapiGetNewRecord();
    // false when deleting
    if (new_review)
    {
        var item_id = new_review.getFieldValue('custrecord_ns_prr_item_id')
        ,   user = new_review.getFieldValue('custrecord_ns_prr_writer');
        // if the record is beeing created
```

```

    // and the user is logged in
    if (action.toString() === 'create' && user)
    {
        verifyItemPurchase(new_review, user, item_id);
    }

    // if the review is not assigned to
    // the item record
    if (!new_review.getFieldValue('custrecord_ns_prr_item') && item_id)
    {
        setReviewItem(new_review, item_id);
    }
}
}

```

Backend Code Update

NetSuite replaced the following code in the **ProductReviews.Model.js**, found in the **ProductReviews@2.1.0/SuiteScript** folder.

Previous Code:

```

data.writer && data.writer.id && review.setFieldValue('custrecord_ns_prr_writer', data.writer.i
d);

```

Updated Code:

```

if(session.isLoggedIn2())
{
    review.setFieldValue('custrecord_ns_prr_writer', nlapiGetUser() + '');
}

```

Rejected Product Review Star Ratings Still Appearing

Previously, when a NetSuite user set a Item record's Product Review **Status** field to **Rejected**, the associated product review still displayed on the web site's Star Ratings area.

Reference Product Review Records bundle fixes this problem.

With this bundle update, you can navigate to an item record's **SuiteCommerce Extensions** tab (**Product Reviews** subtab) and change any Product Review's **Status** field to Rejected. The rejected review will no longer appear on your site.

NetSuite replaced the following code in the **NS_PR_SS_User_Event.js** file within the Reference Product Review Records bundle in this release.

Previous Code:

```

function getRating (id)
{
    'use strict';

    var results = nlapiSearchRecord(
        'customrecord_ns_pr_review', null
    , [
        new nlobjSearchFilter('isinactive', null, 'is', 'F')
    ]
    );
}

```



```

        , new nlobjSearchFilter('custrecord_ns_prr_item', null, 'is', id)
        , new nlobjSearchFilter('custrecord_ns_prr_status', null, 'is', approved_status)
    ]
    , [
        new nlobjSearchColumn('internalid', null, 'count')
        , new nlobjSearchColumn('custrecord_ns_prr_rating', null, 'avg')
    ]
);

if (results && results.length)
{
    return {
        count: results[0].getValue('internalid', null, 'count')
        , average: parseFloat(results[0].getValue('custrecord_ns_prr_rating', null, 'avg')).toFixed(1)
    };
}

return null;
}

```

Updated Code:

```

function getRating (id)
{
    'use strict';

    var results = nlapiSearchRecord(
        'customrecord_ns_pr_review', null
    , [
        new nlobjSearchFilter('isinactive', null, 'is', 'F')
        , new nlobjSearchFilter('custrecord_ns_prr_item', null, 'is', id)
        , new nlobjSearchFilter('custrecord_ns_prr_status', null, 'is', approved_status)
    ]
    , [
        new nlobjSearchColumn('internalid', null, 'count')
        , new nlobjSearchColumn('custrecord_ns_prr_rating', null, 'avg')
    ]
    );

    if (results && results.length)
    {
        var average = parseFloat(results[0].getValue('custrecord_ns_prr_rating', null, 'avg'));

        if (isNaN(average))
        {
            average = 0;
        }

        return {
            count: results[0].getValue('internalid', null, 'count')
            , average: average.toFixed(1)
        };
    }
}

```

```

    return null;
}

```

Facet Priority Ignored When Using the URL Component as a FacetID

Previously, the faceted navigation view ignored the `priority` property configured in the `SC.Shopping.Configuration` module. Due to an error in the code, facet fields did not display as expected. The error has been corrected, and the `priority` property now sets the display order of the facet.

The `Facets.FacetedNavigationItems` function was modified in the following file:

Facets.FacetedNavigation.View.js

Previous Code:

```

'Facets.FacetedNavigationItems': function()
{
    var translator = FacetsHelper.parseUrl(this.options.translatorUrl, this.options.translatorConfig);
    , ordered_facets = this.options.facets && this.options.facets.sort(function (a, b) {
        // Default Priority is 0
        return (translator.getFacetConfig(b.id).priority || 0) - (translator.getFacetConfig(a.id).priority || 0);
    });
}

```

Updated Code:

```

'Facets.FacetedNavigationItems': function()
{
    var translator = FacetsHelper.parseUrl(this.options.translatorUrl, this.options.translatorConfig);
    , ordered_facets = this.options.facets && this.options.facets.sort(function (a, b) {
        // Default Priority is 0
        return (translator.getFacetConfig(b.url || b.id).priority || 0) - (translator.getFacetConfig(a.url || a.id).priority || 0);
    });
}

```

Updated and New Modules

Many of the SuiteCommerce modules have been updated for this release. For a complete comparison of before and after versions, view the [diff-denaliR2-montblanc-distro.patch](#) file.

New modules for this release include the following:

- cartItems@1.0.0
- promocodeForm@1.0.0
- quickAdd@1.0.0
- QuoteToSalesOrder@1.0.0
- QuoteToSalesOrderWizard@1.0.0
- QuoteToSalesOrderValidator@1.0.0
- RequestQuoteAccessPoints@1.0.0
- RequestQuoteWizard@1.0.0

- Sensors@1.0.0
- Transaction@1.0.0

Denali R2 Release of SuiteCommerce Advanced

 **Applies to:** SuiteCommerce Advanced | Denali

Welcome to Denali R2 release of SuiteCommerce Advanced! This version includes the following enhancements:

- Site Management Tools
- Bronto Integration
- Improved Gulp Deploy
- Updated Modules



Important: This release is available as an **update** to the existing Denali release of SuiteCommerce Advanced. If you are working with the existing release of SuiteCommerce Advanced, update the bundle from the NetSuite Installed Bundles record, and download the new source code. Migrate any customizations you may already have in place to the new source code directory. For implementations prior to the Denali release of SuiteCommerce Advanced, migration to this latest version requires a re-implementation.

Site Management Tools

With this release of SuiteCommerce Advanced, Site Management Tools are integrated with the Shopping application. A new CMSadapter module provides for the integration and several template files are configured with site management tool areas.



Note: Currently, Site Management Tools are available for the Shopping application only. Secure domains are unsupported (My Account and Checkout).

The new SuiteCommerce Advanced Site Management Tools enable you to manage content on your SuiteCommerce Advanced websites. Content management is facilitated by the Site Management Tools user interface on the website and the areas in your website page templates. With Site Management Tools you can add new content, edit or remove existing content, and rearrange content by dragging it from one location to another. The types of content you can manage include:

- Images and image attributes such as links and alternate text
- Text with traditional WYSIWYG formatting that utilizes the site's style sheets
- Product lists based on pre-defined merchandising rules
- HTML code snippets for embedded videos, social widgets, partner code, and so forth

Content displays site-wide on every page, only on a specific page, or on specific page types depending on the area where the content resides. For example, the areas in the page header and footer are defined as global, which means that any content you place in those areas will display on every page on the site.

Content is managed on an unpublished version of your site. Content changes are visible to visitors on the live site only after you publish the site. These changes include the addition of new content, deletion of content, rearrangement of content, and so on. This enables you to spend several days or weeks developing a new version of your site. You can then publish the site when you are ready for those changes to go live.

Site Management Tools also includes an edit live site feature that lets you make changes to content on the live site. These changes are available immediately and do not require publishing.

With SuiteCommerce Advanced Site Management Tools you can also create and manage landing pages and enhanced landing pages. Included in this is the management of important SEO considerations for the pages such as page titles, meta description, and meta keywords.

Note: With this release, Site Management Tools are enabled by default and Content Delivery is disabled. We recommend using Site Management Tools but for existing implementations you can continue to use Content Delivery if needed.

For additional information, see the help topic [SMT Overview](#).

Bronto Integration

With this release of SuiteCommerce Advanced, Bronto integration with the Shopping, My Account, and Checkout applications is available. To complete the integration, configure your Bronto `accountId` in the `SC.Configuration.js` file. For details, see the help topic [Bronto Integration](#).

Improved Gulp Deploy

Previously, some customers encountered an `SSS_TIME_LIMIT_EXCEEDED` error when deploying to NetSuite. This issue has been fixed by changing our implementation to use SuiteTalk instead of Suitelets for the file upload process. With Suitelets, we made a single request with all file contents. This resulted in some uploads exceeding Suitelet execution time restrictions. With the move to using SuiteTalk for file uploads, this issue has been resolved.

Note: The gulp command uses SuiteTalk to upload files when deploying to your NetSuite account. To use SuiteTalk, your account must have administrator access and you must enable the Web Services feature. See the help topic [Enabling the Web Services Feature](#) for more information.

We have also significantly improved deployment speeds by uploading only files that have changed. After the initial deployment of all files to NetSuite, a manifest file is created within the NetSuite file cabinet. This file contains the list of files uploaded and a hash of its content. On subsequent deployments, the manifest file is used to determine new or changed files and then only those files are updated during deployment to the NetSuite file cabinet.

For this enhancement, files available in the gulp deploy folder at `gulp/ns-deploy` have been updated.

Updated Modules

The following modules have been updated for this release.

| Module | New/Update | Reason for Update |
|----------------------|------------|-----------------------|
| BrontoIntegration | New | Bronto Integration |
| Cart | Update | Bronto Integration |
| CheckoutApplication | Update | Site Management Tools |
| CMSadapter | New | Site Management Tools |
| Content | Update | Site Management Tools |
| MyAccountApplication | Update | Site Management Tools |


| Module | New/Update | Reason for Update |
|---------------------|------------|-----------------------|
| NavigationHelper | Update | General Denali R2 |
| Profile | Update | Bronto |
| SCA | Update | General Denali R2 |
| SCASB | Update | Bronto Integration |
| SCASBP | Update | Bronto Integration |
| ShoppingApplication | Update | Site Management Tools |
| SspLibraries | Update | Site Management Tools |

Denali Release of SuiteCommerce Advanced

 **Applies to:** SuiteCommerce Advanced | Denali


Welcome to Denali release of SuiteCommerce Advanced! With this release of SCA, NetSuite is bringing you a revamped SuiteCommerce platform that leverages the latest modern web technologies such as Gulp.js, Handlebars.js, and Node.js. We are also releasing new developer tools to enable efficient local development and deployment processes. The user experience takes a Mobile First approach with a redesigned user interface. Modules are versioned to provide simplified upgrades and customizations. And, we continue to provide advanced SEO solutions for site tracking and analysis.

This document details many of the changes included in the release. For complete documentation, blog posts, and other information visit the new [SuiteCommerce Developer's Portal](#).

 **Note:** For release notes on versions prior to Denali release of SuiteCommerce Advanced, see [Release Notes Prior to SuiteCommerce Advanced Denali](#).

Previously, SuiteCommerce Advanced applications were separately bundled (ShopFlow, My Account, and Checkout). Now, SuiteCommerce Applications are bundled as a single SuiteApp to provide simplified customizations and migrations to future releases. The following bundles are available:

- SuiteCommerce Advanced Denali: Bundle ID — **90208**
- Site Builder Extensions Denali: Bundle ID — **90205**
- Site Builder Extensions Premium Denali: Bundle ID — **90206**

 **Important:** This release is a complete re-architecture of the SuiteCommerce Advanced platform. For existing implementations, migration to this latest version requires a re-implementation.

Some of the highlights of this release include:

- [New Developer Tools](#)
- [Logic-less Templating Using Handlebars](#)
- [Mobile First Design](#)
- [Restructure of Application Modules](#)
- [Converted from LESS to Sass CSS Pre-Processor](#)
- [Use of Composite Views Instead of Macros](#)
- [Updated Configuration Files](#)

- Updated Content Delivery and Product Merchandising Bundles
- Translation Support
- Other Architectural Changes

New Developer Tools

With the Denali release of SuiteCommerce Advanced, new developer tools are supported to enable you to deploy SuiteCommerce Advanced applications directly to NetSuite. Using these tools you can edit and customize the application from your local development environment. You can then run tasks to compile, combine, and minify application files locally before deploying your customized application to NetSuite or a local environment for testing. You are no longer required to go into the NetSuite file cabinet to modify application files.

New Tools and Technologies include:

- **Node.js:** This is the JavaScript based application server. Node.js must be downloaded and installed in your local environment.
- **Gulp.js:** Gulp is a JavaScript task runner that uses Node.js. Gulp.js must also be downloaded and installed in your local environment. After your environment is set up with Node.js and Gulp.js, you can use a set of gulp tasks available from the SCA bundle to compile JavaScript and SASS files, deploy SCA code to NetSuite, and so forth. See
- **SASS Compiler:** compiles Sass style sheets used by template files.

For complete details on how to use the new developer tools, see the help topic [Core SuiteCommerce Advanced Developer Tools](#).

Logic-less Templating Using Handlebars

Previously, we used Underscore.js for front-end templating. Now we use Handlebars.js. Handlebars.js provides logic-less semantic templating. Templates contain the HTML used for each module view with placeholder areas defined by double-curly braces where JSON data is injected at run-time.

Note: Although we no longer use Underscore.js for templating, it is still used for many other purposes.

Previously, SCA templates had a `.txt` extension. All templates now have a `.tpl` extension and have been moved to a `Templates` folder within each `Module` folder. Template files are also now combined locally when you run the Gulp deploy commands available with the new developer tools.

For example, the following section is from the `address_list.tpl` template located at `suite_commerce/Address@x.x.x/Templates`. Note the placeholder variables within the double-curly braces.

```
<section class="address-list">
  <h2>{{pageHeader}}</h2>
  <div class="address-list-button-container">
    <a href="/addressbook/new" class="address-list-button-info-cards-new" data-toggle="show-in-modal"> {{translate 'Add New Address'}}</a>
  </div>

  <div class="address-list-default-addresses">
    <div data-view="Addresses.Collection"></div>
  </div>
</section>
```

For more details on Handlebar templating, see the help topic [Logic-less Templates and Handlebars.js](#)

Mobile First Design

SCA is now designed with a Mobile First approach. Mobile First design results in faster and higher performing sites by prioritizing the most important elements for smaller devices first and then scaling the design for larger device sizes. This enhances the user experience by providing a more focused design that allows users to quickly accomplish their core task whether it be finding a product to purchase in a B2C environment or reordering in bulk in a B2B environment.

Using the Mobile First approach, CSS element styles are defined for mobile devices and then scaled progressively for tablet and desktop display.

To support this Mobile First design, SASS styles are now organized in a hierarchy of reusable components. There are Base styles used across applications, template specific styles, and application specific styles. See the help topic [Design Architecture](#) for details on how to work with design elements in your code base. Visit the [SuiteCommerce Developer's Portal](#) design pages for detailed descriptions of all design elements and page flows.

Restructure of Application Modules

Source files for SCA are organized in multiple versioned modules. Each module directory now contains all of the files specific for that module. Previously, template and style sheet files were separated from the modules directories. When customizing a module, you can now create a copy of the module directory, rename with an applicable version, and then package the required versions for distribution using Node and Gulp.

When migrating to the next version of SCA, you can now migrate on a module by module basis. This will enable you to manage your customizations much more efficiently.

For complete details on the architecture of Modules, see the help topic [Module Architecture](#).

Converted from LESS to Sass CSS Pre-Processor

Sites are now designed using the Sass CSS extension language. With Sass, CSS syntax is fully supported in addition to features such as variables and imports. Sass styles are organized within a hierarchy that defines base styles, template specific styles, and application specific styles. Styles are define as reusable components and can be combined to create increasingly complex structures.

For more information on styling in SuiteCommerce Advanced, see the help topic [Design Architecture](#).

Use of Composite Views Instead of Macros

In this release of SuiteCommerce Advanced, all macros have been moved to Backbone Views. Parameters previously defined in macros are now within Views for each module and passed to a context object. Some Views in SuiteCommerce Advanced are Composite Views. Composite Views are views that contain child views or subviews.

For more details, see the help topic [Views](#).

Updated Configuration Files

Previously, the same front-end configuration file was used for Shopping, Checkout, and My Account. That configuration file contained all of the same parameters, although some of those parameters were used only by certain applications. Now there is a `BaseConfiguration` object defined within

the SCA.Configuration model that has global attributes shared across applications and a unique configuration file for each application. Each of the application specific configuration files has a dependency on the base configuration model.

The Backend.Configuration.js file has been renamed to Configuration.js and is now included in the SspLibraries module. This file can be used to set many of the same objects server-side as in prior releases as well as some additional objects specific to this release.

Updated Content Delivery and Product Merchandising Bundles

The Content Delivery and Product Merchandising bundles have been updated with new library files. If you have previously installed these bundles, you must upgrade them to the latest version to use them with this version of SuiteCommerce Advanced.

Multi-Currency Support

Multi-currency is supported for the Shopping and Checkout applications in SuiteCommerce Advanced. Multi-currency is not supported in the My Account application.

Translation Support

Language JavaScript files updated for changes in this release of SuiteCommerce Advanced are coming soon after the initial release. The language files currently available in the application modules are those used in prior releases and are not comprehensive.

Other Architectural Changes

The architectural changes in this release are extensive. Following are some key highlights:

- [Update to Bootstrap 3](#)
- [All Models Inherit from SCModel](#)
- [Filter by Site ID](#)
- [ItemsKeyMapping File Converted to an Application Module](#)
- [Passing Fieldset Names as a Parameter](#)
- [Remove ItemDetails Module from Checkout](#)
- [Minimum Field Set Requirement Changes](#)
- [Facebook appId Configuration](#)
- [Updated 3rd Party JavaScript Libraries](#)

Update to Bootstrap 3

SuiteCommerce Advanced has been upgraded from Bootstrap 2 to use Bootstrap 3 for its front-end framework.

All Models Inherit from SCModel

With this release, there is a new base class, `SCModel`, for all backend models. All models inherit from the `SCModel` and must define a name attribute to uniquely identify the model. Several common base functions for validation, authentication, and error handling are defined within the `SC.Model.js` file. In general, this model should not require customization.

For example, in the following code sample from `ProductList.Model.js`, you can see that first the `SCModel` dependency is defined and then the `ProductList.Model` extends from `SCModel`. Also, the `name` attribute is set as `ProductList`.

```
define(
  'ProductList.Model'
,  ['SC.Model', 'Application', 'ProductList.Item.Search', 'Utils']
,  function(SCModel, Application, ProductListItemSearch, Utils)
{
  'use strict';

  return SCModel.extend({
    name: 'ProductList'
  })
})
```

Filter by Site ID

When you have multiple sites set up for a single account, you can now configure Reference My Account to display orders from specified sites only. This is done by editing the `filter_site` setting in the backend `Configuration.js` file at `SspLibraries/SuiteScript/Configuration.js`.

```
SC.Configuration = {
  ...
  , filter_site: 'current'
  ...
}
```

The following settings are available:

- `filter_site: 'current'`: all orders created for the current site, plus orders created in NetSuite or not assigned to any site are returned
- `filter_site: all`: all orders from all sites in the current account plus orders created in NetSuite or not assigned to any site are returned
- `filter_site: [1,2,3]`: all orders from sites specified by site ID in the array plus orders created in NetSuite or not assigned to any site are returned

Note: Currently, when your account is configured to display orders from multiple sites, those orders are not differentiated by site on the front end.

ItemsKeyMapping File Converted to an Application Module

Previously, the `ItemsKeyMapping` class containing the logic that provided mapping of what is returned from the search API and Commerce API was instantiated in the application object. Now, the `ItemKeyMapping` logic has been made available as an independent `ItemsKeyMapping` module. This improves performance by allowing the class to be instantiated only when a method uses it. For example, the `ItemDetails.Model` now has a dependency on the `ItemsKeyMapping` module and the `ItemsKeyMapping` class is called within the `get` method.

```
define('ItemDetails.Model', ['Backbone.CachedModel', 'ItemOptionsHelper', 'Session', 'ItemsKeyMapping', 'SCA.Configuration', 'underscore', 'Utils']
, function (BackboneCachedModel, ItemOptionsHelper, Session, ItemsKeyMapping, Configuration, _)
{
  ...
})
```

```
{
  'use strict';
  .
  .
  .
  , get: function (attr, dont_cache)
  {
    var configuration = (this.application && this.application.Configuration) || {}
    , keyMapping = _.defaults(configuration.itemKeyMapping || {}, ItemsKeyMapping.getKeyMapping(configuration));
```

Passing Fieldset Names as a Parameter

Previously, only fieldset names predefined in the site configuration could be returned in functions used to retrieve and store item information. Now a `fieldset_name` parameter has been added to these functions. You can leverage this logic to create backend services that specify a specific fieldset name, instead of relying on the site fieldset configuration.

Note: Passing a `fieldset_name` parameter to item functions is not required. If a `fieldset_name` parameter is not passed, the fieldset names as defined in the configuration are used.

For example, note the `fieldset_name` parameter in the `preloadItems` function:

Previously:

```
preloadItems: function (items)
{
  'use strict';

  var items_by_id = {}
  , parents_by_id = {}
  , self = this
  , is_advanced = session.getSiteSettings(['sitetype']).sitetype === 'ADVANCED';
  .
  .
  .
}
```

Updated Code:

```
, preloadItems: function (items, fieldset_name)
{
  var self = this
  , items_by_id = {}
  , parents_by_id = {};

  items = items || [];

  this.preloadedItems = this.preloadedItems || {};
  .
  .
  .
}
```

Remove ItemDetails Module from Checkout

Previously, the entire `ItemDetails` module was included in the Reference Checkout application when only the `ItemDetails` model is required. There is no need for the Router or the entire module definition..

Minimum Field Set Requirement Changes

The `isfulfillable` field must now be defined for the Matrix Child Items field set. Definition is no longer required for the Details field set.

Facebook appId Configuration

With this release, the Facebook sharing feature requires you to set the `appId` in the `SC.Shopping.Configuration.js` file in the `ShoppingApplication` module. Configuration is no longer set in the `Configuration.js` file. To set the `appId`, customize the module or create a custom file override. For more information on the SuiteCommerce Advanced Facebook Share and Like features, see the help topic [Facebook Share](#).

For more information on customizing a module, see the help topic [Customize and Extend Core SuiteCommerce Advanced Modules](#).

Updated 3rd Party JavaScript Libraries

In this release, JavaScript libraries have been migrated to newer releases as detailed in the following table.

| Library | Prior Version | Current Version |
|----------------------|---------------|-----------------|
| almond | 0.3.0 | 0.3.0 |
| Backbone | 1.0.0 | 1.1.2 |
| Backbone.stickit | — | 0.8.0 |
| bignumber | 1.4.1 | 1.5.0 |
| bootstrap | 2.3.1 | 3.3.1 |
| bootstrap-datepicker | - | 1.3.1 |
| font-awesome | - | 4.3.0 |
| handlebars | — | 2.0.0 |
| html5shiv | 3.6.1 | 3.7.2 |
| jquery.bxslider | 4.1.1 | 4.1.2 |
| jquery.cookie | 1.3.0 | 1.4.1 |
| jquery.zoom | 1.7.1 | 1.7.13 |
| jquery | 1.10.2 | 1.11.1 |
| json2 | 2.0.0 | 2.0.0 |
| require | 2.1.1 | 2.1.15 |
| twitter-bootstrap | - | 3.3.1 |
| twitter-typeahead | - | 0.10.5 |
| underscore | 1.6.0 | 1.7.0 |
| Respond.js | - | 1.4.2 |

| Library | Prior Version | Current Version |
|---------|---------------|-----------------|
| jasmine | 1.2.0 | 2.1.3 |

Release Notes Prior to SuiteCommerce Advanced Denali

With the release of SuiteCommerce Advanced Denali, the core architecture and bundling of SuiteCommerce Advanced Reference Implementations was changed.

If you are upgrading an existing Reference Implementation to a newer release, refer to the following for details on changes between releases.



Important: For SuiteCommerce Advanced Denali release notes, see [Denali Release of SuiteCommerce Advanced](#).

Reference ShopFlow:

- ShopFlow 1.07.0
- ShopFlow 1.06.1
- ShopFlow 1.06.0
- ShopFlow 1.05.0
- ShopFlow 1.04.0
- ShopFlow 1.03.0
- ShopFlow 1.02.0

Reference Checkout:

- Checkout 2.05.0
- Checkout 2.04.1
- Checkout 2.04.0
- Checkout 2.03.0
- Checkout 2.02.0
- Checkout 2.0

Reference My Account

- My Account 1.06.0
- My Account 1.05.1
- My Account 1.05.0
- My Account 1.04.0
- My Account 1.03.0
- My Account 1.02.0

SuiteCommerce Advanced 2015.1


- ShopFlow 1.06.0
- Checkout 2.04.0

- [My Account 1.05.0](#)
- [Core Improvements for SuiteCommerce Advanced 2015.1](#)
- [User Experience Improvements for SuiteCommerce Advanced 2015.1](#)

Reference ShopFlow


- [ShopFlow 1.07.0](#)
- [ShopFlow 1.06.1](#)
- [ShopFlow 1.06.0](#)
- [ShopFlow 1.05.0](#)
- [ShopFlow 1.04.0](#)
- [ShopFlow 1.03.0](#)
- [ShopFlow 1.02.0](#)

ShopFlow 1.07.0

 **Note:** The Bundle ID for Reference ShopFlow 1.07 is **96113**.

With this release of ShopFlow, you can now integrate to the new Site Management tools and to Bronto.

- [Bronto Integration](#)
- [Site Management Tools Integration](#)

 **Note:** Bronto integration is also available in Checkout 2.05 or later and My Account 1.06 or later.

Bronto Integration

With this release of Reference ShopFlow, Bronto integration is available. Bronto is a NetSuite company that provides an advanced marketing automation engine and solutions for shopping cart abandonment, post-purchase campaigns and so forth. Bronto can be easily integrated with your Reference ShopFlow, My Account, and Checkout implementations.

The following Bronto Applications are supported:

- [Cart Recovery](#)
- [Conversion Tracking](#)
- [Coupon Manager](#)
- [Pop-up Manager](#)

To implement Bronto integration, you need to modify the Configuration.js file to include your `accountId`.

```
, bronto: {
  accountId: ''
}
});
```

With the `accountId` configured, the BrontoIntegration module causes a configuration.js file to load after the SuiteCommerce Application has initialized. This configuration.js file is cached in the Bronto CDN and uses the `accountId` to pull down the appropriate settings to enable and configure the various integrations setup in the Bronto Connector.

Note: The BrontoIntegration module essentially just provides an interface to the Bronto CDN. Therefore, any updates or enhancements to Bronto applications are completely separate from SuiteCommerce Advanced releases and will be made available through the Bronto CDN. There is no additional customization needed in the SuiteCommerce source code to take advantage of Bronto enhancements.

For detailed information on configuring and managing your Bronto applications, refer to the Bronto documentation.

Site Management Tools Integration

With this release of Reference ShopFlow, Site Management Tools are integrated. A new CMSadapter module provides for the integration and several template files are configured with site management tool areas.

For detailed information on the default areas defined within the template files for Reference ShopFlow, see the Site Management Tools section in the [SuiteCommerce Advanced & Reference Implementations \(pre-Denali\)](#) PDF guide.

For complete information on how to work with Site Management Tools, see the help topic [SMT Overview](#).

Before you begin, you must enable the Site Management Tools feature on the Web Presence subtab at Setup > Company > Enable Features.

By default, in this version of ShopFlow Site Management Tools are configured to be used as the default content management tool. If you prefer to use Content Delivery, the `useCMS` object must be set to `false` in the Configuration.js file.

```
, useCMS: true
,
```

Important: Although this configuration setting is also available in the My Account and Checkout Reference implementations, Site Management Tools are not supported for those applications. The setting in the Configuration.js file is overridden in the Application.js file for secure domains: `SC.Configuration.useCMS = !isSecure && SC.Configuration.useCMS;`

Integrating to ShopFlow 1.06 Implementations

Site Management Tools can be manually integrated into an existing implementation using Reference ShopFlow 1.06. To do this you'll need to copy a number of files from the new release and manually merge some changes into existing files.

Important: Use care when merging changes to ensure that existing customizations for your site are not overwritten.

To integrate Site Management Tools to Reference ShopFlow 1.06:

1. Enable the Site Management Tools feature on the Web Presence subtab at Setup > Company > Enable Features.
2. Install Reference ShopFlow 1.07.
3. Download the source files from ShopFlow 1.07

4. In your ShopFlow 1.06 source, rename the file 'js/src/app/modules/CMSadapter/CMSadapter.js' to 'adapter.js'.
5. Copy the following files into your ShopFlow 1.06 source in the file cabinet:
 - 'js/src/app/modules/CMSadapter/*.*)
 - 'ssp_libraries/models/CMSadapter.js'
 - 'templates/cmsadapter/cms_landing_page.txt'
6. Migrate the code snippets outlined for the following files to your implementation.
 - 'index.ssp'

```

@@ -143,6 +143,10 @@
    }
  </script>

+<% if (SC.Configuration.useCMS) { %>
+  <script src="/cms/2/assets/js/postframe.js"></script>
+<% } %>
+
  <script>
    if (!SC.isCrossOrigin())
    {
@@ -169,8 +173,10 @@
      loadScript([
        '<%= session.getAbsoluteUrl("shopping", 'js/libs/Libraries-0146da3524b6.js') %>'
        , '<%= session.getAbsoluteUrl("shopping", 'templates/Templates-0146da42942c.js') %>'
+<% if (SC.Configuration.useCMS) { %>
+  , '<%= session.getAbsoluteUrl("shopping", 'js/Application-0146db0f5f0f.js') %>'
-  // , '<%= session.getAbsoluteUrl("shopping", 'js/Application-0146db0f5f0f.js') %>'
      ]);

    if (SC.ENVIRONMENT.jsEnvironment == 'browser')

```

- 'index-local.ssp'

```

@@ -67,6 +67,10 @@
  </head>
  <body>

+  <% if (SC.Configuration.useCMS) { %>
+    <script src="/cms/2/assets/js/postframe.js"></script>
+  <% } %>
+
  <script>

    // Do we have SEO Support
@@ -105,6 +109,20 @@

  </script>

+  <% if (SC.Configuration.useCMS) { %>
+    <script>
+      if (SC.ENVIRONMENT.jsEnvironment === 'browser')

```

```

+      {
+          setTimeout(function()
+          {
+              jQuery.getScript('/cms/2/cms.js')
+                  .done(function() {
+                      CMS.trigger('cms:load');
+                  });
+              }, 5000);
+          }
+      </script>
+      <% } %>

<noscript>
    <div class="container">

```

■ 'sc.environment.ssp'

```

@@ -7,7 +7,8 @@
    , Content
    , DefaultPage
    , Merchandising
-   , Error;
+   , Error
+   , CMS;

    try {

@@ -17,6 +18,20 @@
        Language = Environment.currentLanguage && Environment.currentLanguage.locale || '';
        Currency = Environment.currentCurrency && Environment.currentCurrency.code || '';

+       // The use of CDS and CMS are mutually exclusive, if you use CMS you can't use CDS, or
+       if you use CDS you can't use CMS
+       if (SC.Configuration.useCMS)
+       {
+           try
+           {
+               CMS = Application.getModel('CMSadapter').getPages();
+           }
+           catch(e)
+           {
+               console.warn('CMS could not be loaded, reason: ' + JSON.stringify(e));
+           }
+       }
+       else
+       {
+           // Content depends on the installation and inclusion of the
+           // ContentDeliverService provided as a separated bundle
+           // If you need to add more tags to the listURL function please consider
@@ -48,6 +63,7 @@
        {
            console.warn('Merchandising Module not present in ShopFlow SSP');
        }
+    }

```



```

        response.setCDNCacheable(response.CACHE_DURATION_MEDIUM);
    }
@@ -102,6 +118,11 @@
    SC.ENVIRONMENT.MERCHANDISING = <%= JSON.stringify(Merchandising, {}) %>;
    <% } %>

    // CMS configuration
    +<% if (SC.Configuration.useCMS) { %>
    +    SC.ENVIRONMENT.CMS = <%= JSON.stringify(CMS || {}) %>;
    +<% } %>
    +
    // Touch Support
    // Checks if this is a touch enabled device
    SC.ENVIRONMENT.isTouchEnabled = 'ontouchstart' in window || window.DocumentTouch && document i
instanceof DocumentTouch;

```

■ 'ssp_libraries/Models.js'

```

@@ -2373,3 +2373,25 @@
    }
    });

    //Model.js
    // @module CMSadapter
    +Application.defineModel('CMSadapter', {
    +
    +    // @method getPages @return {data:Array<CMSPages>}
    +    getPages: function()
    +    {
    +        var siteSettings = Application.getModel('SiteSettings').get();
    +        var cmsRequestT0 = new Date().getTime();
    +        var cmsPagesHeader = {'Accept': 'application/json' };
    +        var cmsPagesUrl = 'https://system.netsuite.com/api/cms/pages?site_id=' + siteSettings.s
iteid + '&c=' + nlapiGetContext().getCompany() + '&{}';
    +        var cmsPagesResponse = nlapiRequestURL(cmsPagesUrl, null, cmsPagesHeader);
    +        var cmsPagesResponseBody = cmsPagesResponse.getBody();
    +        var data = {
    +            _debug_requestTime: (new Date().getTime()) - cmsRequestT0
    +            , pages: JSON.parse(cmsPagesResponseBody)
    +        };
    +        return data;
    +    }
    +});
    +
    +

```

■ 'ssp_libraries/commons.js'

```

@@ -220,9 +220,16 @@
    , getEnvironment: function (session, request)
    {
        'use strict';

    +
    +        var isSecure = request.getURL().indexOf('https:') === 0;
    +
    +

```

```

+      // HEADS UP!! This hack is because currently CMS doesn't support Secure Domain yet
+      // When CMS does support it, delete this
+      SC.Configuration.useCMS = !isSecure && SC.Configuration.useCMS;
+
+      // Sets Default environment variables
+      var context = nlapiGetContext()
-      , isSecure = request.getURL().indexOf('https:') === 0
+      //, isSecure = request.getURL().indexOf('https:') === 0
+      , siteSettings = session.getSiteSettings(['currencies', 'languages'])
+      , result = {
+          baseUrl: session.getAbsoluteUrl(isSecure ? 'checkout' : 'shopping', '{{file}}')
+      )
@@ -233,6 +240,7 @@
+      , companyId: context.getCompany()
+      , casesManagementEnabled: context.getSetting('FEATURE', 'SUPPORT') === 'T'
+      , giftCertificatesEnabled: context.getSetting('FEATURE', 'GIFTCERTIFICATES') ===
'T'
+      , useCMS: SC.Configuration.useCMS
+      };

+      // If there are hosts associated in the site we iterate them to check which we are in

```

■ 'ssp_libraries/backend.configuration.js'

```

@@ -269,4 +269,7 @@
+      }
+      */
+    ]
+
+    // @property {Boolean} The use of CDS and CMS are mutually exclusive, if you use CMS yo
u can't use CDS, or if you use CDS you can't use CMS
+    // By default we use CMS, so if you are going to use CDS, set 'useCMS' to 'false'
+    , useCMS: true
+  };

```

■ 'js/Starter.js'

```

@@ -6,6 +6,12 @@

+    application.getConfig().siteSettings = SC.ENVIRONMENT.siteSettings || {};

+    // The page generator needs to run in sync in order to work properly
+    if (SC.ENVIRONMENT.jsEnvironment !== 'browser')
+    {
+      jQuery.ajaxSetup({ async: false });
+    }
+
+    SC.compileMacros(SC.templates.macros);

+    // Requires all dependencies so they are bootstrapped
@@ -78,9 +84,6 @@
+    jQuery('.seo-remove').remove();
+  }
+};
-
-

```


ShopFlow 1.06.1

Note: The Bundle ID for Reference ShopFlow 1.06.1 is **69016**.

In this release, a bug with Internet Explorer 8 has been fixed. The issue is a result of an unsupported regular expression character ("^") that causes SuiteCommerce pages to improperly render. This issue affects only customers using Internet Explorer 8. All other browsers are not affected by this issue.

To implement the fix, the **Content.EnhancedViews.js** file at Reference js/src/app/modules/Content/Content.EnhancedViews.js was modified with the following change:

Note: This file was also modified in Checkout and My Account. If you are using those implementations, you must also make the change or upgrade to the latest bundle for each.

Previous Code:

```
, enhanceMetaTags: function (view)
{
  var enhanced_meta_regex;
  this.$head
    // then we add the description
    .find('meta[name="description"]').attr('content', view.metaDescription || view.getMetaDe
scription() || '').end()
    // and keywords meta tags
    .find('meta[name="keywords"]').attr('content', view.metaKeywords || view.getMetaKeyword
s() || '').end()
    // then we append the tags specific to this view
    // excluding any extra descriptions and keywords meta tags
    .append(view.getMetaTags().not('[name="description"]').not('[name="keywords"]'));

  /* jshint ignore:start */
  // related to issue https://github.com/jshint/jshint/issues/823
  enhanced_meta_regex = /\<!\- \- EnhanceMetaTags:STARTS[^\]+EnhanceMetaTags:ENDS \- \- \>/;
  /* jshint ignore:end */
  if (document.head.innerHTML.match(enhanced_meta_regex))
  {
    document.head.innerHTML = document.head.innerHTML.replace(enhanced_meta_regex, '');
  }

  if (view.metaextra)
  {
    endTo(this.$head);
  }

  return this;
}
```

Updated Code:

```
, enhanceMetaTags: function (view)
{
  this.$head
    // then we add the description
```

```

        .find('meta[name="description"]').attr('content', view.metaDescription || view.getMetaDe
scription() || '').end()
        // and keywords meta tags
        .find('meta[name="keywords"]').attr('content', view.metaKeywords || view.getMetaKeyword
s() || '').end()
        // then we append the tags specific to this view
        // excluding any extra descriptions and keywords meta tags
        .append(view.getMetaTags().not('[name="description"]').not('[name="keywords"]'));

// remove head's elements that are between enhanceMetaTagsStart and enhanceMetaTagsEnd
var remove_element = false; //flag used to determine when remove elements.

jQuery('head').children().each(function ()
{
    var $element = jQuery(this)
        , element_id = $element.attr('id');

    if (element_id === 'enhanceMetaTagsStart')
    {
        remove_element = true;
    }

    if (remove_element)
    {
        $element.remove();
    }

    if (remove_element && element_id === 'enhanceMetaTagsEnd')
    {
        remove_element = false;
    }
});

//add meta extra elements between enhanceMetaTagsStart and enhanceMetaTagsEnd
if (view.metaextra)
{
    jQuery('<script id="enhanceMetaTagsStart"></script>' + view.metaextra + '<script id="enh
anceMetaTagsEnd"></script>').appendTo(this.$head)
}

return this;
}


```

ShopFlow 1.06.0

This document details changes to Reference ShopFlow since the previous release. For a complete **DIFF** of files between this release and Reference ShopFlow 1.05.0, click the following link to download the .patch file:


[.patch](#).

Use a tool such as Crucible to open this .patch file and graphically visualize all additions, deletions, and modifications. The .patch file can also be viewed in any text editor and will display appropriate syntax highlighting when syntax is set to Diff.

 **Note:** The Bundle ID for Reference ShopFlow 1.06.0 is **69016**.

Support for URL Component Aliases

With the release of ShopFlow 1.06, URL component aliases are now supported. You can now define aliases for both items and facets. When a user enters a URL alias, they are now redirected to the canonical URL component. If the alias is not defined, a 404 page not found error is thrown. For detailed information on defining URL components in NetSuite, see the help topic [URL Components](#).

 **Important:** Redirection occurs in both the client (browser) and server (SEO) environments.

To implement support for URL component aliases the following modules were modified:

- **Facets.js:** The `prepareRouter` function was modified to include facet aliases as a possible route. Possible facet aliases are retrieved from the `siteSettings` object in the `backend.configuration.js` file.

The `siteSettings` object is cached for two hours by default. When aliases are added or modified, those changes will not be effective until the cache is expired. If required, the cache can be disabled in the `backend.configuration.js` file by changing the cache object to the following:

```
cache: {
  siteSettings: 2 * 60 * 60 * 0 // DISABLED
}
```

- **Facets.Router.js:** This facets router needs to know how to handle URL aliases. The `facetLoading` function has been modified as follows:
 - `model.fetch()` now handles results using a promise and uses the Search API response for retrieving the URL corrections.
 - Added the `getFacetsAliasesMapping(corrections)` function. This function returns a mapping of the facet aliases.
 - Added the `unaliasUrl(alias_url, corrections)` function. This function processes the aliased URL and returns the canonical URL.
 - The Search API response is checked to see if corrections are present. If false, the view is rendered as usual. If true, the URL is unaliased by calling a new function named `unaliasUrl(alias_url, corrections)`.
- **ItemDetails.Router.js:** The `productDetails(api_query, base_url, options)` function has been modified to support item aliases. The Search API call success callback now checks if the response contains corrections. If corrections are present, then the user is redirected to the canonical item URL.

Issue Fixes

This release contains many bug fixes including:

Images Overflow the Modal

Previously, when the dimensions of an image loading in a modal dialog were greater than it's container, the image covered many elements of the page. This has been corrected so that images now load within the wrapping container.

The following files were modified:

- **ImageLoader.js:** updated the `fixImagesForLoader` function.

```
var fixImagesForLoader = function (s)
{
  return s.replace(/<img(?:[^>]*)src="([^\"]+)"(?:[^>]*)/gi, function(all, textBefore, url, textAfter)
  {
    textBefore = textBefore || '';
    textAfter = textAfter || '';
    // do nothing if image contains data-loader="false" attribute
    if ( (textBefore && textBefore.indexOf('data-loader="false"') !== -1) || (textAfter && textAfter.indexOf('data-loader="false"') !== -1) )
    {
      return all;
    }
    var params = _.parseUrlOptions(url);
    var height = params.resizeh || default_height;
    var style_attrs = 'style="min-height:' + height + 'px;min-width:' + height + 'px"';
    var ret = '<img data-image-status="pending" data-src="' + url + '" ' + style_attrs + textBefore + ' ' + textAfter;
    return ret;
  });
};
```

- **item_image_gallery.txt:** added the `data-loader="false"` attribute to the `img` src tags.
- **quick_view.txt:** added the `data-loader="false"` attribute to the `img` src tags.

Numeric Fields in Merchandising Zones

Previously, using a merchandising rule with a numeric field as a filter threw an error. This has been fixed so that the page correctly displays.

The following file was modified: **Context.DefaultHandlers.js**

```
if (values.length)
{
  filters[key] = values.join(',');
}
else
{
  delete filters[key];
}
```

Console Log Errors

Previously, a `Couldn't load Categories` error message was returned in the browser developer tools console. This was debugging information only and had no impact on customers but the messaging has been commented out to prevent the errors from printing. `console.log` statements were commented out in the following files:

- **Content.LandingPages.js**
- **Facets.Translator.js**

Unique Creation Dates for Product Reviews

Previously, the creation date of a Product Review returned the value of the `created` field on the NetSuite record. This is the time-stamped date reflecting the date when the record was created within NetSuite. When Product Review records were imported via CSV, all reviews were time stamped with the date of that import giving users and analytic engines the false impression that all reviews were entered on the same date. Now, a new data field has been added to the Product Review record so that the date the review was created by the user can be reflected in the front end instead of the creation date of the corresponding NetSuite Product Review record.

To implement this change, the `Model.js` file was modified to assign the value from the new custom field, `custrecord_ns_prr_creation_date`, to the `create_on` variable. When a value for the custom field does not exist, the value from the standard NetSuite `created` field is returned the same as it was before this release.

```
,   created_on: review.getFieldValue('custrecord_ns_prr_creation_date') || review.getFieldValue('created')
```



Important: To take advantage of the new field you must **update** the Reference Product Review Records bundle. (Do not re-install the bundle. Instead, do an update.)

Item with Invalid Matrix Item options not setting correct URL

Previously, when a user selects a matrix item with an invalid combination of options, the URL was not correctly updated and kept all selected options. On page refresh of the invalid URL, the following console error was returned and a blank page was displayed.

```
Uncaught RangeError: The combination you selected is invalid
```

This has been corrected so that when a user selects multiple options on a matrix item resulting in an invalid combination, the invalid options are cleared and the URL is correctly updated.

Internal Error Thrown when Accessing Products with Reviews

This error has been corrected by adding the `recordTypeHasField` function in `commons.js` as follows. This function checks if certain fields exist on a record. Fields are included or excluded in search columns based on what the function returns.

The following files were modified: `commons.js`, `Model.js`

```
function recordTypeHasField (record_type_name, field_name)
{
  'use strict';

  try
  {
    var record = nlapiCreateRecord(record_type_name);
    return _.contains(record.getAllFields(), field_name);
  }
  catch (error)
  {
    return false;
  }
}
```


Session Lost when Switching Languages

Previously, when a user logged in while using a language different from the site default language and then changed the selected language, the session was lost. This was corrected by adding session parameters when switching domains.

The following files were modified:

- **MultiHostSupport.js**

```
// add session parameters to target host
url = SC.Utills.addParamsToUrl(url, SC.Utills.getSessionParams(application.getConfig('siteSetting
s.touchpoints.login')));
```

- **NavigationHelper.js**

```
// add session parameters to target host
fixed_url = SC.Utills.addParamsToUrl(fixed_url, SC.Utills.getSessionParams(application.getConfig(
'siteSettings.touchpoints.login')));
```

- **Utils.js**

```
function getSessionParams (url)
{
  // add session parameters to target host
  var params = {}
  ,   ck = getParameterByName(url, 'ck')
  ,   cktime = getParameterByName(url, 'cktime');

  if (ck && cktime)
  {
    params.ck = ck;
    params.cktime = cktime;
  }

  return params;
}

function getParameterByName(url, param_name) {
  param_name = param_name.replace(/\[/, '\\[').replace(/\]/, '\\]');
  var regex = new RegExp('[\\?&]' + param_name + '=(^&#)*')
  ,   results = regex.exec(url);
  return results === null ? '' : decodeURIComponent(results[1].replace(/\+/g, ' '));
}
.
.
.
```

Merchandising Rules Content Persisting

Previously, merchandising rules content displayed when that content was accessed by the direct URL. This behavior is expected. However, merchandising rules content also displayed when navigating any page that contained a fragment of that URL. This was corrected by rendering content only after the backbone history has started.

The following files were modified:

- **Content.EnhancedViews.js:** set `Backbone.history.started` to `true`.

```
Backbone.History.started = true;
```

- **Content.js:** Added the following conditional statement to check if `Backbone.History.started` has been called.

```
if (!Backbone.History.started)
{
    return;
}
.
.
.
else if (content_zone.contenttype === 'merchandising')
{
    EnhancedViews.previousPlaceholders.push(content_zone.target);
}
```

Blank Search Page in Internet Explorer 8

In this release, the search page now correctly displays in IE8. Previously, it displayed blank. This was corrected by using an underscore function instead of the `Objects.keys` method in the `Utils.js` file.

Changed:

```
if (params && Object.keys(params).length)
```

To:

```
if (params && _.keys(params).length)
```

Unable to Edit Item Quantities in the View Cart Page on iPad Devices

Previously, users could not successfully edit item quantities in the View Cart Page on some iPad devices. This has been corrected.

Added noindex, nofollow Meta Tags to Review Pages

Previously, product review pages in SuiteCommerce Advanced sites were crawled by Google and sometimes returned rankings higher than the product pages. To avoid this, we included all reviews within the PDP itself with pagination. The number of reviews displayed can be changed in the `backend.configuration.js` file with a maximum of 25 reviews allowed.

Images not Loaded Until User Action

Previously, images were not loaded on the home page or in the search results page until a user action was initiated. This has been corrected by modifying the following files:

- **ImageLoader.js:** added the `afterRender` parameter in the `mountToApp` function.

```
application.getLayout().on('afterAppendView afterRender', function()
```

- **typeahead_macro.txt:** Added the `data-loader="false"` attribute to the `img` class.

```
">
```

- **Merchandising.Zone.js:** added the following trigger to the `appendItems` method.

```
// notify the layout that the content might have changed
this.options && this.options.application && this.options.application.getLayout().trigger('after
Render');
```

addThis Widget is Disabled by Default

In this release the `addThis` widget is disabled by default in the `Configuration.js` file.

```
, addThis: {
  enable: false
```

Implemented `rel="next"` and `rel="prev"` Link Elements

The HTML link elements `rel="next"` and `rel="prev"` are used to indicate the relationship between component URLs in a paginated series. These elements were missing in the header and have now been added to improve SEO. To do this, the `setLinkRel` method has been added to the `Content.EnhancedViews.js`.

```
if (previous_page)
{
  this.setLinkRel('prev', previous_page);
}

if (next_page)
{
  this.setLinkRel('next', next_page);
}

return this;
}

, setLinkRel: function (rel, link)
{
  jQuery('<link />', {
    rel: rel
    , href: link
  }).appendTo(this.$head);
}
```

Images not Wrapped in `noscript` Tag In SEO Page Generator

Previously, to prevent loading of images in SEO generated pages, each image tag was wrapped in a `<noscript>` tag for SEO generated content. To improve performance, the `#main` div is now wrapped with the `noscript` tag.

Narrow by Section Displayed with Zero Results

Previously, when using faceted navigation, if zero results were returned the “Narrow By” section did not display. This has been corrected by adding the following conditional statement in the faceted navigation macro `faceted_navigation_macro.txt`. Now the Narrow By section will display when zero results are returned.

```
<% if (has_facets || applied_facets.length) { %>
```

Last Image is Display when First is Selected in Chrome

Previously, when viewing the item details page in Chrome, the last image of an item was displayed even when the first image was selected. To correct this the following CSS was added to the `image-gallery.less` file.

```
.bx-viewport li { min-height: 1px; min-width: 1px; }
```

Crashing in Internet Explorer 7

Previously, Reference ShopFlow would crash in IE7 due to an incorrect handling of `value` in the CSS selector of the `.rating-area` style. This style is not used, therefore, it has been commented out of the `rating.less` file.

```
/* .rating-area button[value="0"] {
  left: -20px;
  position: absolute;
  top: 0;
}*/
```

Product Review Error Displayed when Product Reviews not Installed

Previously, when the Reference Product Review Records bundle was not installed, the following error was incorrectly displayed when loading the Product Detail pages:

```
The record type [CUSTOMRECORD_NS_PR_REVIEW] is invalid.
```

To correct this a `REVIEWS_CONFIG` object was added to the `SC.ENVIRONMENT`. `starRating` macros are not rendered when `enabled` is set to false. In addition, the review component inside the product display page is not instantiated.

The following files were modified:

- `sc.environment.ssp`: added the `REVIEWS_CONFIG` object.

```
SC.ENVIRONMENT.REVIEWS_CONFIG = {
  enabled: <% recordTypeExists('customrecord_ns_pr_review') %>
};
```

- `ItemDetails.View.js`:

```
var reviews_enabled = SC.ENVIRONMENT.REVIEWS_CONFIG && SC.ENVIRONMENT.REVIEWS_CONFIG.enabled;

if (reviews_enabled && product_reviews_placeholder.size() > 0)
```

- **Several macro/template files:** set `showRatingCount` to `false` for the `starRating` div in the `review_form.txt`, `reviews_center_for_item.txt`, and `show_review_macro.txt` files.

```
, showRatingCount: false
```

- **product_list_edit_item.txt:** modified the `item-rating` div to set `showRatingCount` to `true`.
Changed:

```
<div class="item-rating" itemprop="aggregateRating" itemscope itemtype="http://schema.org/AggregateRating">
  <%= SC.macros.starRating({
    max: view.options.application.getConfig('productReviews.maxRate')
  ,   value: rating
  ,   className: 'star pull-left'
  ,   fillRounded: true
  }) %>
  <span class="review-total">
    <%= _('({0})').translate(
      '<span itemprop="reviewCount">' + item_details.get('_ratingsCount') + '</span>'
    ) %>
  </span>
</div>
```

To:

```
<div class="item-rating" itemprop="aggregateRating" itemscope itemtype="http://schema.org/AggregateRating">
  <%= SC.macros.starRating({
    max: view.options.application.getConfig('productReviews.maxRate')
  ,   value: rating
  ,   ratingCount: item_details.get('_ratingsCount')
  ,   showRatingCount: true
  ,   className: 'star pull-left'
  ,   fillRounded: true
  }) %>
</div>
```

Gallery Twitter Card Meta Tags

Previously, Twitter Gallery Cards could not be approved because the `twitter:site` and `twitter:creator` meta tags were not defined. In this release, these meta tags were added to the `metaTagMappingTwitterGalleryCard` object in the **Configuration.js** file.


```
, 'twitter:site': seo_twitter_site
, 'twitter:creator': seo_twitter_creator
```

ShopFlow 1.05.0

This document details changes to Reference ShopFlow since the previous release. For a complete **DIFF** of files between this release and Reference ShopFlow 1.04.0, click the following link to download the .patch file:


[.patch](#).

Use a tool such as Crucible to open this [.patch](#) file and graphically visualize all additions, deletions, and modifications. The [.patch](#) file can also be viewed in any text editor and will display appropriate syntax highlighting when syntax is set to Diff.

 **Note:** The Bundle ID for Reference ShopFlow 1.05.0 is **59235**.

Save For Later

Using Save For Later, users can move items from the Cart to a list for later reference and then add items from the Save for Later list directly to the Cart.

 **Note:** The Save for Later feature is built on the same architecture as Product Lists.


Save for Later provides the following functionality:

- Items in the Save for Later list are restored with each login session.
- By default, users can change the quantities in the Save for Later list.
- When an item in a Save for Later list goes out of stock, that item can no longer be added to the Cart and an out of stock message is displayed.
- When a shopper is not logged in they are redirected to the Login page when the Save for Later button for an item in the cart is clicked.

For detailed information on Save for Later, see the help topic [Save For Later](#).

List Headers

Reference ShopFlow 1.05.0 now includes the List Header component. This component defines sorting and pagination options used in pages that render lists of items or records. The List Header component can be used within different views and interacts with the collection displayed within that view. It is reusable and configurable to suit a variety of uses.

 **Note:** The List Header component was previously released with Reference My Account 1.03.0.


For detailed information on List Headers, see [List Headers](#).

ShopFlow 1.04.0

This document details changes to Reference ShopFlow since the previous release. For a complete **DIFF** of files between this release and Reference ShopFlow 1.03.0, click the following link to download the [.patch](#) file:

[.patch](#).

Use a tool such as Crucible to open this [.patch](#) file and graphically visualize all additions, deletions, and modifications. The [.patch](#) file can also be viewed in any text editor and will display appropriate syntax highlighting when syntax is set to Diff.

 **Note:** The Bundle ID for Reference ShopFlow 1.04.0 is **56319**.

The following enhancements have been made:

- [Minimum Field Set Requirements Updated](#)
- [Google Cached Pages Correctly Generated](#)

- Added Configurable Cache Parameters
- Performance Enhancements

Minimum Field Set Requirements Updated

The following minimum Field Set requirements changes have been made:


- **New typeahead Field Set:** To support enhanced performance, the **typeahead** Field Set is now required. You can manually add this Field Set to the Fields Sets subtab on the Web Site record at Setup > SuiteCommerce Advanced > Set Up Web Site. You can also use the setup script to automatically populate all required Field Sets. For the complete setup script see the help topic [Field Set Script](#). The following Field Set definition has been added to the Setup Script.

```
siteRecord.selectNewLineItem('fieldset');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetname', 'Type Ahead');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetid', 'typeahead');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetrecordtype', 'ITEM');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetfields', 'itemid,displayname,storedisplayname2,urlcomponent,itemimages_detail');
siteRecord.commitLineItem('fieldset');
recId = nlapiSubmitRecord(siteRecord);
```

- **New Field Sets for Related and Correlated Items:** Previously, related and correlated item detail information was included in the detail and order field sets. These have been removed from the detail and order field sets and are now defined in two new field sets. This enables calling the information asynchronously later in the script to improve initial load of the item details page and the cart. The following Field Set information has been added to the Setup Script:

```
siteRecord.selectNewLineItem('fieldset');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetname', 'Related Items Details');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetid', 'relateditems_details');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetrecordtype', 'ITEM');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetfields', 'relateditems_detail');
siteRecord.commitLineItem('fieldset');

siteRecord.selectNewLineItem('fieldset');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetname', 'Correlated Items Details');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetid', 'correlateditems_details');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetrecordtype', 'ITEM');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetfields', 'correlateditems_detail');
siteRecord.commitLineItem('fieldset');
recId = nlapiSubmitRecord(siteRecord);
```

 **Note:** For the full setup script, see the help topic [Import Field Sets](#).

Google Cached Pages Correctly Generated

Google caches snapshots of each page it crawls. Users can then retrieve cached versions of pages from Google search results. Previously, an error was returned when users attempted to retrieve Google's cache of SuiteCommerce Advanced site URLs.

Now, when our application is loaded from an external domain such as Google cached pages, only the SEO output markup and styles are rendered. No JavaScript is run.

For **Reference ShopFlow 1.04**, a new `isCrossOrigin` method was added to the `index.ssp`:

```

/* declare SC namespace here */
var SC = window.SC = {
  ENVIRONMENT: {
    jsEnvironment: (typeof nsglobal === 'undefined') ? 'browser' : 'server'
  }
,  isCrossOrigin: function()
  {
    return '<%= Environment.currentHostString %>' !== document.location.hostname;
  }
,  isPageGenerator: function()
  {
    return typeof nsglobal !== 'undefined';
  }
,  getSessionInfo: function(key)
  {
    var session = SC.SESSION || SC.DEFAULT_SESSION || {};
    return (key) ? session[key] : session;
  }
};

```

and we control when to load or not load things inside the index like this:

```

if (!SC.isCrossOrigin())
{
  }

```

If you are updating an existing **Reference ShopFlow 1.03** implementation, add the IF statements, `if ('<%= Environment.currentHostString %>' === document.location.hostname)`, to the `index.ssp` file as shown in the following code snippet. This IF statement checks if the request is from an external domain. When the request is from an external domain, related code then uses this information to ensure that Javascript is not loaded and that the application is not started.

```

<body>

  <script>
    If ('<%= Environment.currentHostString %>' === document.location.hostname)
    {
      document.write('<div class="seo-remove">');
      loadScript({
        url: '<%= session.getAbsoluteUrl("shopping", "sc.environment.ssp?lang=" + Language
+ "&cur=" + Currency) %>'
        ,   seo_remove: true
      });
      document.write('<\\div>');
    }
  </script>
  <script>
    If ('<%= Environment.currentHostString %>' === document.location.hostname)
    {
      document.write('<div class="seo-remove">');
      for (var i = 0; i < SC.LOAD_FILES.length; i++)
      {
        loadScript({

```



```

        url: SC.ENVIRONMENT.baseUrl.replace('{{file}}', SC.LOAD_FILES[i])
        ,   seo_remove: true
      });
    }
  }
  document.write('<\\div>');
</script>

<noscript>
  <div class="container">
    <div class="alert alert-error alert-noscript">
      <strong>Javascript is disabled on your browser.</strong><br>
      To view this site, you must enable JavaScript or upgrade to a JavaScript-capable br
rowser.
    </div>
  </div>
</noscript>

<div id="main" class="main"></div>

<script>
  If ('<%= Environment.currentHostString %>' === document.location.hostname)
  {
    // This minimizes flickery of the 1st page load!
    jQuery && jQuery('#main').empty();
  }
</script>
</body>

```



Note: Clear your site cache after applying this fix and then wait for Google to regenerate cached pages.

Added Configurable Cache Parameters

Content rendered by the Content Delivery service is cached for two hours, by default. Previously, to modify this caching duration a **time to live (ttl)** parameter needed to be added to the Content.DataModels.js module. Now, caching duration can be controlled by modifying the Configuration.js file. Two types of cache can be set:

- **CDN Cache:** Content cached on the Content Delivery Network
- **Application Cache:** Data cached on the NetSuite servers


```

,   cache: {
    // cdn cache duration for content pages. Valid values are 'SHORT', 'MEDIUM', 'LONG'
    contentPageCdn: 'MEDIUM'

    // application cache for content pages - value in seconds and must be between 2 hours
    and 5 minutes
    ,   contentPageTtl: 2 * 60 * 60
  }

```

The Content.DataModels.js file has been modified to call these two configuration parameters. Previously, this file set the cache as short.

 **Note:** For more information on valid caching values SHORT, MEDIUM, and LONG, see the help topic `setCDNCacheable(type)`.

Performance Enhancements

The following performance enhancements have been made in Reference ShopFlow. If you have customizations to the files involved in these changes, you should consider merging these changes or editing your customized files to incorporate the changes. To maximize performance benefits, it is recommended that you implement **ALL** changes. Each enhancement may not have the same performance gain when implemented in isolation.

- Stop Bootstrapping the Cart
- Remove Duplicate Language File
- Reorganize How the JavaScript is Loaded in the Page
- Optimize Images from SEO Output
- Removed Unnecessary Spaces from SEO Output
- Redirect Users When pushState is not Enabled
- Remove `jQuery(document).ready` from Starter.js
- Configurable Application Linking
- Optimize Image Loading and Sizes in the PDP
- Move SocialNetworks Initialization to `.once(afterAppendView)`
- Load SocialNetworks Scripts only if the View Contains Placeholders
- Remove the Timestamp from Images Created with BXSlider
- Parallelize `sc.user.environment.ssp`
- Stop Bootstrapping Product Lists

Stop Bootstrapping the Cart

Previously, loading the user shopping cart caused delays when loading the `sc.user.environment.ssp`.

Now, the application environment loads first. After the application starts, the shopping cart information is fetched and the shopping cart notifies the application when the cart information is ready to be displayed.

Remove Duplicate Language File

Previously, the language file was loaded on the environment and `user.environment`. Now, the language file is loaded only by the `index.ssp` file.

Reorganize How the JavaScript is Loaded in the Page

Previously, javascript libraries were loaded from a file that was not cached.

Now, the `BootUtilities` file was removed from the application and its capabilities moved to the following:

- The console and json shim libraries were moved to a combiner file named `Libraries-xxxxxxx.js`
- The `loadScript` function was moved to the top of the `index.ssp` file.
- The `html5` shim is included in-line at the top of the `index.ssp` file.
- Style information was moved to the bottom of the `index.ssp` file. Although style information is generally loaded at the beginning of a web-based application, this was done for performance improvements.

Optimize Images from SEO Output

Previously, images were loaded in the SEO generated HTML causing a delay in the load of JavaScript files. Now, images are loaded only when the page views and content are rendered.

To prevent loading of images in SEO generated pages, each image tag is wrapped in a `<noscript>` tag for SEO generated content only. The `minifyMarkup()` function within the `Underscore.templates.js` file has been modified in the following two ways:

- Images in template and macro output are wrapped in a `<noscript>` tag:

```
minifyMarkup = function (text)
{
  if (isSeo() && text)
  {
    text = text
    ...
    ...

    // Performance: wrap all images with noscript if in SEO so the browser
    // don't start loading the images when parsing the SEO markup.
    // We do this with a regexp instead using parsed object because of the
    // SEO engine. The following regexp wrap all <img> tags
    // with <noscript> only if they are not already wrapped. It supports the
    // three formats: <img />, <img></img> and <img>
    .replace(/(<img\s+[^>]*\s*</img>|<img\s+[^>]*\/>|(?<img\s+[^>]*>)(?!<s*</img>))
    (?!\s*<s*\/noscript\s*>)/gmi, '<noscript>$1</noscript>');

  }
  return text || '';
}
```

- The `jQuery.html()` and `jQuery.append()` methods are overridden to apply the `<noscript>` tag:

```
if (isSeo())
{
  var jQuery_originalHtml = jQuery.fn.html;
  jQuery.fn.html = function(html)
  {
    if (typeof html === 'string')
    {
      html = minifyMarkup(html);
      html = removeScripts(html);
    }
    return jQuery_originalHtml.apply(this, [html]);
  };

  var jQuery_originalAppend = jQuery.fn.append;
  jQuery.fn.append = function(html)
  {
    if (typeof html === 'string')
    {
      html = minifyMarkup(html);
      html = removeScripts(html);
    }
  }
}
```

```

    return jQuery_originalAppend.apply(this, [html]);
  };
}

```

Removed Unnecessary Spaces from SEO Output

Previously, SEO generated pages included spaces and debug comments in the HTML the same as the rendered views and content. Because each whitespace and comment requires the creation of a new node by the SEO generator, performance is improved by removing them. This is done in the following two files:

- **Underscore.templates.js:** New lines and tabs between HTML tags are removed for SEO generated pages. Also, the `<!-- begin macro` and `<!-- begin template` debug comments are removed.
- **Underscore.templates.js:** The `jQuery.html()` and `jQuery.append()` methods have been overridden to minify markup injected by content rendered by the Content Delivery and Product Merchandising services.

Redirect Users When pushState is not Enabled

By using `pushState`, Reference applications avoid multiple HTTP request-response cycles when updating URLs. Because IE8 and IE9 do not support `pushState`, a function immediately following the open head tag has been added to minimize the time for URL redirects. The user is redirected to a fixed URL as soon as the page loads.

```

if (!history.pushState && SC.ENVIRONMENT.jsEnvironment === 'browser' && (location.pathname !==
"/" || location.search !== "") && location.hash === '')
{
  location.replace('/#' + location.pathname + location.search);
  document.write("");
}

```

Remove jQuery(document).ready from Starter.js

Previously, the `application.start()` function was not initiated until document ready. Now, the `application.start()` function is done in parallel with loading additional resources. To accomplish this, the `jQuery(document).ready` function was removed from the `Starter.js` file, and the order of loading core JavaScript files in the `index.ssp` was changed so that the `Application.js` file is loaded last.

```

loadScript([
  '<%= session.getAbsoluteUrl("shopping", 'js/libs/Libraries-0146f82c3202.js') %>'
, '<%= session.getAbsoluteUrl("shopping", 'templates/Templates-0146f82c24a9.js') %>'
, '<%= session.getAbsoluteUrl("shopping", 'js/Application-0146f82c2a83.js') %>'
]);

```



Note: You can configure from where to load these files. See [Configurable Application Linking](#).

Configurable Application Linking

Previously, the core combined JavaScript files could be loaded from only the `sc.user.environment.ssp` file. Now, you can configure these files to load from the `sc.user.environment.ssp` file, `sc.environment.ssp` file, or `index.ssp` file. By default, the core JavaScript files are loaded from the `index.ssp` file. The following table summarizes the benefits.

Note: Combined JavaScript files include the Libraries.js, Templates.js, and Application.js files.

| Location | Cache | Performance |
|-------------------------|------------------|-------------|
| sc.user.environment.ssp | no cache | poor |
| sc.environment.ssp | MEDIUM (2 hours) | medium |
| index.ssp (default) | LONG (7 days) | best |

Optimize Image Loading and Sizes in the PDP

In the Product Display Page, three sizes of the same image are available: thumbnail, main, and zoom. Previously, for mobile devices, the large zoom image size was downloaded and then shrunk to fit the screen width. Now, only the smaller image size is downloaded for mobile devices. This is done in the `item_image_gallery.txt` template file:

```
<% registerMacro('itemImageGallery', function (images, view) {
  var resizeImage = view.application.resizeImage
  , image_resize_id = SC.ENVIRONMENT.screenWidth < 768 ? 'thumbnail' : 'zoom';

  %>
```

Move SocialNetworks Initialization to .once(afterAppendView)

Previously, social network libraries made two requests prior to a page being rendered. Now, social network requests are made only after the initial page is rendered by moving the request to the first `afterAppendView`.

```
mountToApp: function(){
  layout.on('afterAppendView', function(){
    if(!scriptLoadedYet && layout.containsPlaceholdersFor('facebook and addthis'))
    {
      loadScripts('facebook.js', 'addthis.js')
    }
  });
}
```

Load SocialNetworks Scripts only if the View Contains Placeholders

Previously, if addThis and Facebook were enabled in the `Configuration.js` file, social media widgets were loaded regardless of whether the page requested had these in the DOM. Now, the addThis and Facebook scripts are loaded only when the DOM has placeholders for these widgets. This is done in the `SocialSharing.js` file:

```
mountToApp: function(){
  layout.on('afterAppendView', function(){
    if(!scriptLoadedYet && layout.containsPlaceholdersFor('facebook and addthis'))
    {
      loadScripts('facebook.js', 'addthis.js')
    }
  });
}
```

Remove the Timestamp from Images Created with BXSlider

Previously the BXSlider widget included a timestamp internally for onload image events. This caused images to incorrectly cache. Now, the timestamp has been removed. The following line has been commented out of the file `jquery.bxslider.js`:

```
if($(this).is('img')) $(this).attr('src', $(this).attr('src') + '?timestamp=' + new Date().getTime());
```

Parallelize `sc.user.environment.ssp`

Previously, the application start up was done only after the user profile was loaded by the `sc.environment.ssp` application. Now, `sc.user.environment.ssp` is loaded asynchronously and a global promise object is resolved after the asynchronous profile request is completed. This enables core resources to load and the application startup process to begin, postponing rendering of the UI.

To accomplish this, the following changes have been implemented.

- **Asynchronous Loading:** the `sc.environment.ssp` file is loaded asynchronously.
- **Use of a Global promise object:** the promise is resolved after the user profile is loaded.
- **Provided a new Asynchronous User Profile Method:** the `application.getUserPromise()` method is provided for asynchronous user profile requests where needed. The existing `application.getUser()` method is still used for synchronous calls.



Important: Previously, the profile module was ready when any module code was run. Now, a module's `mountToApp()` function can be run without the profile data being available. Use the `getUserPromise()` function from static code such as modules to ensure that the promise object is resolved. Use the synchronous `getUser()` function as needed from views and templates.

- **Refactor of the Application Architecture:** modules can now be mounted before profile information is available.
- **Initial Rendering of Content is Postponed Until the Profile is Ready:** The `Shopping.Layout.showContent` method is overridden to show content only when the profile promise is resolved and the profile has loaded. The `Layout.showContent` method is now responsible for triggering the routers.

Stop Bootstrapping Product Lists

Previously, Product List data was loaded at the initial startup in `sc.user.environment.ssp`, which slowed the loading of other core resources. Now, the new method `getProductListsPromise()` in the `ProductList.js` module returns a promise resolve only after the Product List data is available. In addition, modules dependent on the ProductList module have been refactored using `application.getProductLists`.


Changes to files are mostly related to calling the `getProductListsPromise()` method before displaying the Product List control on the page.



Note: For pages displaying product list information, the application displays a Loading Lists... message until that data is available.

ShopFlow 1.03.0

With the release of Reference ShopFlow 1.03.0 the following enhancements have been made:

 **Note:** The Bundle ID for Reference ShopFlow 1.03.0 is **53050**.

Product Lists

When a web store is using Reference My Account 1.03.0 with this version of Reference ShopFlow, Product Lists (Wishlist) is now supported.


Using product lists, users in a B2B or B2C can do the following:

- Flag Items to be added to a product list as they are browsing in your web store.
- Create new lists when flagging items.
- Edit Item order details from within the Product Lists pages.
- Move Items between Product Lists.
- Add an entire list of products to the shopping cart with a single click.
- Use Product Lists to Reorder lists of items.

For detailed information on Product Lists, see the help topic [Product Lists](#)


Reference Product Lists Bundle

To use the Product Lists feature, you must install the Reference Product Lists Bundle. This bundle installs the custom records required to store and manage Product List data.


 **Note:** The Bundle ID for Reference Product Lists Records is **53051**.

Product Reviews Updates

To better support the bundle upgrades, this version of ShopFlow has decoupled the Product Reviews custom records from the Reference ShopFlow bundle. Now, you must install a separate **Reference Product Reviews Records** bundle to support the product reviews capability.

 **Note:** The Bundle ID for Reference Product Reviews Records is **53053**.

After installing this bundle, product review data can be viewed on Item records under the SuiteCommerce Extensions tab. Also, to support a more user friendly experience, Product Review custom fields have been renamed.

 **Important:** If you are using Product Reviews in a current implementation, do not uninstall the current version of ShopFlow. Uninstalling will remove the Product Review custom records, lists, and fields, and all associated data will be lost. You can keep the current custom records in place to retain all data and install the new Reference Product Reviews bundle for new product reviews data until data migration can take place.

For detailed information on working with Product Review records, see the help topic [Product Reviews](#).

Minimum Field Set Requirements Updated

To support the new My Account Billing feature, the **order** Field Set is now required. You can manually add this Field Set to the Fields Sets subtab on the Web Site record at Setup > SuiteCommerce Advanced > Set Up Web Site. You can also use the setup script to automatically populate all required Field Sets. For the complete setup script see the help topic [Field Set Script](#). The following Field Set definition has been added to the Setup Script.

```

siteRecord.selectNewLineItem('fieldset');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetname', 'order');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetid', 'order');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetrecordtype', 'ITEM');
siteRecord.setCurrentLineItemValue('fieldset', 'fieldsetfields', 'itemimages_detail,itemoptions_detail,onlinecustomerprice_detail,displayname,internalid,itemid,storedisplayname2,urlcomponent,outofstockmessage,showoutofstockmessage,isinstock,isbackorderable,ispurchasable,pricelevel1,pricelevel1_formatted,stockdescription,matrixchilditems_detail,itemtype,minimumquantity,isonline,isinactive');
siteRecord.commitLineItem('fieldset');

```

Pinterest Enhancements

Reference ShopFlow now supports Product Rich Pins from Pinterest in the product details page. By default, shoppers can hover over an image and then click a Pinterest icon, or click a Pinterest button from within the cart details region. The Pinterest entry automatically populates with item details such as the product image, pricing, or availability.

For more information, see the help topic [Pinterest](#).

Support for Twitter Product Cards

Reference ShopFlow now supports Twitter Product Cards which enable the ability to attach media experiences to Tweets that link to product detail pages. Shoppers can Tweet your products by clicking the Twitter button on the product details page and a card visible to all of their followers is automatically added to their Tweet. This Card can show item details including an image, description, and two other key item details such as price, availability, sizes, or colors.

For more information, see the help topic [Twitter Product Cards](#).

Issue Fixes

This release contains many bug fixes including:

- **Improved Performance of the Live Order Model:** When returning item data to the cart, only item attributes required for display to the user are retrieved. This results in significant performance improvements. Now, approximately 15, cached item attributes are retrieved. Previously, all item attributes were retrieved.
- **Errors When Reordering Items:** Previously, when multiple orders existed in a users account, the error message `Script Execution Usage Limit Exceeded` was thrown when attempting to reorder those orders. This has been resolved.
- **Zero Results in Pasted URLs:** Previously, when copying and pasting URLs with spaces in keywords, zero results were returned due to double encoding of spaces in keywords. Now the correct number of results are returned.
- **Enhanced Cross Site Scripting (XSS) Prevention:** All output from `request.getParameter` calls now compare URL parameters to text and the ability for users to run code through the URL parameter is eliminated.
- **IE Compatibility View Support:** Users that have IE Compatibility View enabled for SuiteCommerce Advanced Sites they are visiting can now successfully proceed to Checkout. Previously, Compatibility View needed to be disabled for the site in order for the user to proceed.
- **Search Type Ahead Fixes:** Previously, when a user entered search terms and clicked enter without waiting for the type ahead feature to complete, the results page does not display. Now the results page correctly displays based on the search query entered without waiting for the type ahead feature values to complete loading.

- **Misdirection of Items with URL Component Starting with Cart Misdirected Correction:** Previously, when users attempted to navigate to an item starting with a URL component of Cart they were incorrectly redirected to the shopping cart page. This has been fixed so that the user is corrected directed to the item details page.
- **Cart Performance Issue in Site Builder Sites:** To increase cart performance the `items_fields_standard_keys` object in the `backend.configuration.js` file has been modified to restrict the number of fields returned by default.

ShopFlow 1.02.0

With the release of Reference ShopFlow 1.02 the following enhancements have been made:

- **Content Delivery Updates:**
 - **Content Delivery in Checkout and My Account:** You can now include content generated from the Content Delivery service in Reference Checkout and Reference My Account implementations. When multiple Reference Implementations are being used for your web store, you can specify which implementations use a Content Delivery record by using the **app** tag with the name of the implementation as follows: Checkout, MyAccount, or Shopping. For example, using the tag **app:Shopping** causes the content to display in only the Shopping Reference implementation.
 - **Content Delivery in Backbone Modals:** You can now include content generated from the Content Delivery service inside Backbone Modals. To use Content Delivery content inside a modal, the target ID defined in the Content Delivery record must be referenced by adding the prefixViewId "in-modal-". You can also specify whether that content should display only when inside a modal, inside or outside of a modal, or outside of a modal.
 - **CDN Caching Enhancement:** Now content returned using the Content Delivery service is cached using CDN.
- **Support for Correlated and Related Items:** By default, when an item has correlated and related items associated with an item on the item record, the Reference ShopFlow implementation now displays those correlated and related items on the item product details page within your site. If you are upgrading from a previous version of ShopFlow, ensure that you have the proper Field Sets defined to support this change. For more information, see *Defining Required Field Sets* and *Correlated and Related Items*.
- **Matrix Item Support:** Matrix Items are supported and displayed as default functionality. Previously, the `itemOptionId` values were not available from the Item Search API for matrix items. These values needed to be manually configured in the `Configuration.js` file. Now, the Item Search API includes this information so manual configuration is NOT needed.



Important: If you are upgrading from a previous version of ShopFlow and have already configured matrix items in your `Configuration.js` file you should remove these configurations. Any settings in the `Configuration.js` file take precedence over the default behavior.

- **Google Universal Analytics:** NetSuite now supports Google Universal Analytics tracking code. Universal Analytics gives you access to improved data processing, new collection methods, and more analysis tools.



Note: To prevent historical data from being lost, the Google Analytics (ga.js) technology will continue to be supported. However, for any new implementations, we recommend that Universal Analytics (analytics.js) be used.

NetSuite now also supports multiple analytics trackers. You simultaneously gather data for both Classic Google Analytics and Google Universal Analytics. You can also add your own custom analytics module to gather data in addition to the Google Analytics data.

For more information, see the help topic [Google Universal Analytics](#).

- **CDN Cache Duration:** The CDN cache duration in the `sc.environment.ssp` file has been changed from LONG to MEDIUM.
- **New Backend Configuration file:** In this release, a new configuration file has been added that enables you to set a number of server-side objects. Many of the available configuration options provide better performance. The file is located at Web Site Hosting Files > Live Hosting Files > SSP Applications > NetSuite Inc. - Shopping > Reference ShopFlow > `ssp_libraries`.


For detailed information, see *Configuring Backend Objects Using the `backend.configuration.js` File*

- **General SEO Enhancements:**
 - **Added <h1> to <h5> Tags and Default Placement:** Previously H tags were not used by default on the Facet, Category, and Product pages. Now, H tags are used to give order and hierarchy to the content in the web pages, which improves SEO results.
Examples include:
 - Product pages: H1 is used for Product Names and H2 is used for subtab titles.
 - Faceted pages: H1 is used for the main page title, H2 is used for the Product Name, H3 for the Facet name, and H4 for the Facet options
 - **Absolute URLs Used In Canonical Tags:** Previously, relative URLs were used in canonical tags. To improve SEO performance, absolute URLs are now used.
 - **Deindexed ShopFlow URLs for Google:** Previously, duplicate indexing was performed because Google crawled the main site as well as the ShopFlow SSP URL. Now a robots.txt file has been added that causes Google to ignore ShopFlow URLs when indexing. You can modify this file as needed and place it in your hosting root.
 - **Removed Cases of Double H1s:** Cases of redundant H1s in Faceted search results have been restructured to have a single H1. The duplicate H1s were used for Responsive Design. Now, the H1 for mobile sites has been removed.
 - **Quick View Fix:** Previously, due to linking characteristics within the page, search engine results were often returning Quick View, instead of the product title. This has been fixed to ensure that the Quick View links are hidden from search engine crawlers and the product name is correctly returned in search results.
 - **Schema.org Fixes:** The Schema.org implementation has been redesigned to properly display structured data information in search engines results.

Reference Checkout

- [Checkout 2.05.0](#)
- [Checkout 2.04.1](#)
- [Checkout 2.04.0](#)
- [Checkout 2.03.0](#)
- [Checkout 2.02.0](#)
- [Checkout 2.0](#)

Checkout 2.05.0

 **Note:** The Bundle ID for Reference Checkout 2.05.0 is 96111.

With this release of Reference Checkout, Bronto integration is available. Bronto is a NetSuite company that provides an advanced marketing automation engine and solutions for shopping cart

abandonment, post-purchase campaigns and so forth. Bronto can be easily integrated with your Reference ShopFlow, My Account, and Checkout implementations.

The following Bronto Applications are supported:

- Cart Recovery
- Conversion Tracking
- Coupon Manager
- Pop-up Manager

To implement Bronto integration, you need to modify the Configuration.js file to include your `accountId`.

```
, bronto: {
  accountId: ''
}
});
```

With the `accountId` configured, the BrontoIntegration module causes a configuration.js file to load after the SuiteCommerce Application has initialized. This configuration.js file is cached in the Bronto CDN and uses the `accountId` to pull down the appropriate settings to enable and configure the various integrations setup in the Bronto Connector.

Note: The BrontoIntegration module essentially just provides an interface to the Bronto CDN. Therefore, any updates or enhancements to Bronto applications are completely separate from SuiteCommerce Advanced releases and will be made available through the Bronto CDN. There is no additional customization needed in the SuiteCommerce source code to take advantage of Bronto enhancements.

For detailed information on configuring and managing your Bronto applications, refer to the Bronto documentation.

Checkout 2.04.1

Note: The Bundle ID for Reference Checkout 2.04.1 is **69019**.

In this release, a bug with Internet Explorer 8 has been fixed. The issue is a result of an unsupported regular expression character ("^") that causes SuiteCommerce pages to improperly render. This issue affects only customers using Internet Explorer 8. All other browsers are not affected by this issue.

To implement the fix, the **Content.EnhancedViews.js** file at Reference `js/src/app/modules/Content/Content.EnhancedViews.js` was modified with the following change:

Note: This file was also modified in ShopFlow and My Account. If you are using those implementations, you must also make the change or upgrade to the latest bundle for each.

Previous Code:

```
, enhanceMetaTags: function (view)
{
  var enhanced_meta_regex;
  this.$head
    // then we add the description
```

```

        .find('meta[name="description"]').attr('content', view.metaDescription || view.getMetaDe
scription() || '').end()
        // and keywords meta tags
        .find('meta[name="keywords"]').attr('content', view.metaKeywords || view.getMetaKeyword
s() || '').end()
        // then we append the tags specific to this view
        // excluding any extra descriptions and keywords meta tags
        .append(view.getMetaTags().not('[name="description"]').not('[name="keywords"]'));

/* jshint ignore:start */
// related to issue https://github.com/jshint/jshint/issues/823
enhanced_meta_regex = /\<\!\-\- EnhanceMetaTags:STARTS[^\]+EnhanceMetaTags:ENDS \-\-\>/;
/* jshint ignore:end */
if (document.head.innerHTML.match(enhanced_meta_regex))
{
    document.head.innerHTML = document.head.innerHTML.replace(enhanced_meta_regex, '');
}

if (view.metaextra)
{
    endTo(this.$head);
}

return this;
}

```

Updated Code:

```

, enhanceMetaTags: function (view)
{
    this.$head
        // then we add the description
        .find('meta[name="description"]').attr('content', view.metaDescription || view.getMetaDe
scription() || '').end()
        // and keywords meta tags
        .find('meta[name="keywords"]').attr('content', view.metaKeywords || view.getMetaKeyword
s() || '').end()
        // then we append the tags specific to this view
        // excluding any extra descriptions and keywords meta tags
        .append(view.getMetaTags().not('[name="description"]').not('[name="keywords"]'));

    // remove head's elements that are between enhanceMetaTagsStart and enhanceMetaTagsEnd
    var remove_element = false; //flag used to determine when remove elements.

    jQuery('head').children().each(function ()
    {
        var $element = jQuery(this)
        ,     element_id = $element.attr('id');

        if (element_id === 'enhanceMetaTagsStart')
        {
            remove_element = true;
        }

        if (remove_element)

```

```

    {
        $element.remove();
    }

    if (remove_element && element_id === 'enhanceMetaTagsEnd')
    {
        remove_element = false;
    }
});

//add meta extra elements between enhanceMetaTagsStart and enhanceMetaTagsEnd
if (view.metaextra)
{
    jQuery('<script id="enhanceMetaTagsStart"></script>' + view.metaextra + '<script id="enhanceMetaTagsEnd"></script>').appendTo(this.$head)
}

return this;
}


```

Checkout 2.04.0

This document details changes to Reference Checkout since the previous release. For a complete **DIFF** of files between this release and Reference Checkout 2.03.0, click the following link to download the .patch file:


[.patch](#).

Use a tool such as Crucible to open this .patch file and graphically visualize all additions, deletions, and modifications. The .patch file can also be viewed in any text editor and will display appropriate syntax highlighting when syntax is set to Diff.

 **Note:** The Bundle ID for Reference Checkout 2.04.0 is **69019**.

Multiple Ship To

With this release, the new Multiple Ship To (MST) feature enables you to configure your site so that shipments can be made to different addresses from within the same order. For detailed information, setup, and configuration of MST see the help topic [Multiple Ship To](#).

 **Important:** In this Limited Release version of MST there are several known restrictions. See the help topic [Known Limitations](#) for further details.

Issue Fixes

This release contains many bug fixes including:

Skip Login and Promo Codes

Previously, promo codes added while on the View Cart page resulted in an error. This has been corrected by modifying the following code in the **CheckoutSkipLogin.js** file.

Changed:

```
LiveOrderModel.prototype.save = _.wrap(LiveOrderModel.prototype.save, wrapper);
```

To:

```
if (window.location.protocol !== 'http:') { LiveOrderModel.prototype.save = _.wrap(LiveOrderModel.prototype.save, wrapper); }
```

Skip Login and Guest Email Field

With Skip Login enabled, an anonymous user skips the login/register page when they navigate to Checkout. Previously, upon login in the Skip Login flow, the guest email field continued to display. To correct this, the following two changes were made in the **OrderWizard.Module.RegisterEmail.js** file:

- The `isActive` function was added to check that the user is logged in before rendering the module.

```
,   isActive: function()
  {
    return (this.wizard.application.getUser().get('isGuest') === 'T' || (this.wizard.application.getConfig('checkout_skip_login') && this.wizard.application.getUser().get('isLoggedIn') !== 'T'));
  }
```

- The `render` function was changed to check the results of the `isActive` function.

```
,   render: function ()
  {
    if (!this.isActive())
    {
      return this.$el.empty();
    }

    var profile = this.profile = this.wizard.options.profile;

    this._render();

    if (profile.get('email') && this.wizard.isPaypalComplete())
    {
      this.trigger('ready', true);
    }
  }
```

Blank Review Page in Internet Explorer 8

In this release, the review page now correctly displays in IE8. Previously, it displayed blank. This was corrected by changing the name of the reserved JavaScript keyword `class` to `className` in the following files:

- `credit_memo_details.txt`
- `invoice_details.txt`
- `item_details_line_macro.txt`
- `order_wizard_showshipments_module.txt`
- `quote_details.txt`

- `return_authorization_details.txt`

Pay with Gift Certificate Persists when Feature is Disabled

Previously, the Pay with Gift Certificate option displayed in the Payment Method page even when the feature was disabled in the account. This has been resolved by adding a check to determine if the Gift Certificate module is enabled. The following files have been modified:

- `commons.js`:

```
, giftCertificatesEnabled: context.getSetting('FEATURE', 'GIFTCERTIFICATES') === 'T'
```

- `Module.PaymentMethod.GiftCertificates.js`:

```
// Determines if the current module is valid to be shown and operate with
, isActive: function ()
{
  return SC.ENVIRONMENT.giftCertificatesEnabled;
}

, render: function()
{
  // Is Active is overridden by child modules, like Shipping to hide this module in Multi Sh
  ip To
  if (!this.isActive())
  {
    this.$el.attr('class', '');


    return this.$el.empty();
  }
}
```

Checkout 2.03.0

This document details changes to Reference Checkout since the previous release. For a complete **DIFF** of files between this release and Reference Checkout 2.02.0, click the following link to download the .patch file:

[.patch](#).

Use a tool such as Crucible to open this .patch file and graphically visualize all additions, deletions, and modifications. The .patch file can also be viewed in any text editor and will display appropriate syntax highlighting when syntax is set to Diff.

 **Note:** The Bundle ID for Reference Checkout 2.03.0 is 59236.

Skip Login

When Skip Login is enabled, the login/register page is skipped when an anonymous user navigates to Checkout. The user is redirected directly to the first checkout step. By default, Skip Login is disabled. To configure your implementation to use the Skip Login feature, override the `checkout_skip_login` object in the `backend.configuration.js` file and set to true.

```
, checkout_skip_login: true
```

Support for Google Universal Analytics

Google Universal Analytics is now supported for Reference Checkout. Previously, the module was available only for Reference ShopFlow. For detailed information on using Google Universal Analytics, see the help topic [Google Universal Analytics](#).

Item Pricing Fixes

Previously, in some cases the Item Price in the shopping cart displays 0.00 and the actual price is crossed out. This error has been fixed in the current release but you can manually fix this issue without upgrading to the latest bundle.

To manually fix this issue without upgrading to the latest bundle:

1. Download the file Model.js located at Reference Checkout/ssp_libraries.
2. Look for the following code:

```
var amount = toCurrency(original_line.amount)
// Total may be 0
, total = (original_line.promotionamount !== '') ? toCurrency(original_line.promotionamount)
: toCurrency(original_line.amount)
, discount = toCurrency(original_line.promotiondiscount) || 0;
```

And update to the following:

```
var amount = toCurrency(original_line.amount)
// Total may be 0
, total = (original_line.promotionamount') ? toCurrency(original_line.promotionamount) : toC
urrency(original_line.amount)
, discount = toCurrency(original_line.promotiondiscount) || 0;
```



Note: The difference in the above code sample is changing
original_line.promotionamount !== '' to original_line.promotionamount'.

3. Save Model.js with a new name.
4. Upload Model.js to the original folder.
5. Update the Reference Checkout SSP Application to reference the new uploaded file.

Reference Checkout Performance Enhancements

The following performance enhancements have been made in this version of Reference Checkout. If you have customizations to the files involved in these changes, you should consider merging these changes or editing your customized files to incorporate the changes. To maximize performance benefits, it is recommended that you implement **ALL** changes. Each enhancement may not have the same performance gain when implemented in isolation.

Fewer Redirects

Previously, when a shopper selected Proceed to Checkout, the Reference Checkout performed three redirects.

Now, the application uses HTTP POST rather than GET. This is a general change for all touchpoints within the Reference Implementations. These touchpoints are handled by the module defined in the NavigationHelper.js file. This module uses the following logic when processing a data-touchpoint.

If the following three conditions are true, the Reference Implementations perform a POST operation:

- a shopper clicks a data-touchpoint, and
- the current touchpoint is different from target touchpoint, and
- the target touchpoint is within a secure domain

In all other cases, the Reference Implementation performs a GET operation as before.

DNS Pre-fetching

DNS pre-fetching is a technique to improve performance when loading pages. DNS prefetching attempts to resolve domain names before a shopper clicks on a link. To perform DNS-prefetching, the following was added to the <head> tag of the index.ssp file:

```
<script>
if (typeof nsglobal === 'undefined')
{
  <%
var checkout_tokens = session.getSiteSettings(['touchpoints']).touchpoints.checkout.split('
/')
, prefetch_url = checkout_tokens[0] + '//' + checkout_tokens[2]
, prerender_url = session.getSiteSettings(['touchpoints']).touchpoints.login;
%>
var prefetch_url = '<%= prefetch_url %>'
, prerender_url = '<%= prerender_url %>';
document.write('<link rel="prefetch" href="' + prefetch_url + '">');
document.write('<link rel="dns-prefetch" href="' + prefetch_url + '">');
document.write('<link rel="prerender" href="' + prerender_url + '">');
}
</script>
```

Changes to Shopping Cart Bootstrap in the Login and Register Pages

Previously, loading the user shopping cart caused delays when loading the sc.user.environment.ssp.

Now, when loading the Login and Register pages, the application environment loads first. After the application starts, the shopping cart information is fetched and the shopping cart notifies the application when the cart information is ready to be displayed.

Checkout pages continue to bootstrap the shopping cart as before.

Checkout / Login Unification

Previously, if a shopper was not logged in and clicked Checkout, they would have to access the Checkout page, then be redirected to the Login page.

Now, if a shopper is not logged in, they are redirected directly to the Login page. To accomplish this, the login.ssp file was removed and its purposefulness moved to the index.ssp file.

Remove Duplicate Language File

Previously, the language file was loaded on the environment and user.environment. Now, the language file is loaded only by the index.ssp file.

Skip Login for Guest Users

Previously, if a shopper was a guest user, they would be directed to the login page before proceeding to checkout.

Now, there is a configurable option that enables guest users to proceed directly to checkout, bypassing the login page.

This option is configured using the `checkout_skip_login` property in the `backend-configuration.js` file. By default, this option is disabled. When it is enabled, an anonymous shopper is redirected to the checkout page, skipping the login/register page.

Moved Style Information to Bottom of `index.ssp`

Style information was moved to the bottom of the `index.ssp` file. Although style information is generally loaded at the beginning of a web-based application, this was done for performance improvements.

Remove Bootutilities.js


Previously, javascript libraries were loaded from a file that was not cached.

Now, the `BootUtilities` file was removed from the application and its capabilities moved to the following:

- The console and json shim libraries were moved to a combiner file named `Libraries-xxxxxxx.js`
- The `loadScript` function was moved to the top of the `index.ssp` file.
- The html5 shim is included in-line at the top of the `index.ssp` file.

Checkout 2.02.0

With the release of Reference Checkout 2.02.0 the following enhancements have been made:

 **Note:** The Bundle ID for Reference Checkout 2.02.0 is now 49333.

- **Limited Fields Returned in Searches:** There is a new `backend.configuration.js` file that restricts the fields returned in searches, providing enhanced performance. Previously, for Site Builder sites **all** fields were returned using the Commerce API. Now, only a subset of files is returned. For SuiteCommerce Advanced sites, you can customize which fields are returned using the Item Search API for Checkout versus ShopFlow order pages. For both Site Builder sites and SuiteCommerce Advanced sites, you can modify the `backend.configuration.js` file as needed to return only those fields used in your web store implementation.

For detailed information, see *Configuring Backend Objects Using the `backend.configuration.js` File*

- **Multiple Versions of the Bundle can be Installed:** Multiple versions of the Reference Checkout bundle can be installed and deployed to different sites. To differentiate between bundles, unique version information is included in the Root URL and SSP application name. For example the SSP application name for this release is Reference Checkout 2.02.0 and the Root URL is `https://checkout-2-02-0.netsuite.com`.

Because the version information is included in the SSP application name, you can also differentiate bundle versions when assigning Touch Points.

- **Support for Correlated and Related Items:** By default, for Site Builder sites when an item has correlated and related items associated with an item on the item record, the Reference Checkout implementation now displays those correlated and related items in the View Cart page. For more information, see the help topic [Correlated and Related Items](#).

Checkout 2.0

The latest version of Reference Cart and Checkout has been redesigned to leverage the same modern, scalable, and extendable HTML and JavaScript architecture as the Reference Shopping and My Account bundles available for SuiteCommerce Advanced sites. This new design provides a flexible, fast, and open front-end solution.



Important: Reference Cart and Checkout is available as a SuiteApp and supports both SuiteCommerce Site Builder and SuiteCommerce Advanced sites. The bundle is released outside of the normal phased release cycle and may not be immediately available when your account is upgraded to Version 2013 Release 2. Consult with your NetSuite account manager for availability. If you are currently using Reference Cart & One Page Checkout, you must re-implement any customizations made to that bundle in the Reference Cart and Checkout bundle. This is due to many core enhancements in Reference Cart and Checkout including architectural backend changes, CSS changes, and the use of different templates.

For details about Reference Cart and Checkout enhancements, see the following:

- [New Reference Cart and Checkout Architecture](#)
- [Checkout Steps Configuration Wizard](#)

New Reference Cart and Checkout Architecture

The Reference Cart and Checkout SuiteApp includes HTML and CSS templates, JavaScript libraries (jQuery.js, Bootstrap.js, Underscore.js, and Backbone.js), and SuiteScript service files. You can use the supplied code to build an out-of-the box cart and checkout solution with little or no customization, or you can modify and extend the code as needed for a fully customized experience.

For detailed information on the new architecture, refer to the following help topics:

- [SuiteCommerce Development Overview](#)
- [Core SuiteCommerce Advanced Developer Tools](#)
- [Customize and Extend Core SuiteCommerce Advanced Modules](#)

Checkout Steps Configuration Wizard

The latest version of this SuiteApp also includes a new checkout steps configuration wizard. Previously, Reference Cart and Checkout was designed with a default three-step checkout process where you could customize the order of the steps. Now, Reference Cart and Checkout is designed with **step groupings** that consist of a set of customizable modules.

With the checkout configuration wizard you can:

- Add or remove steps from step groupings as needed
- Customize each step to include or exclude modules as needed
- Add configuration parameters to each module
- Extend the implementation to add your own custom cart and checkout modules

For example, the following code snippet in the configuration.js file defines the **Shipping** step group. The Shipping step group consists of two steps, **Shipping Address** and **Choose delivery method**. The Shipping Address step contains a single module and the Choose delivery method step contains two modules. You can redefine the Shipping step group by modifying this code.

```
{ name: 'Shipping' , steps: [ { name: 'Shipping Address' , url: '' , modules: [
  'OrderWizard.Module.Address.Shipping' ] } , { name: 'Choose delivery method' , url:
```

```
'shipping/method' , modules: [ 'OrderWizard.Module.Address.Shipping' ,
'OrderWizard.Module.Shipmethod' ] } ] }
```

Reference Cart and Checkout includes the following pre-defined modules that you can add as needed:

- OrderWizard.Module.Address.Billing.js
- OrderWizard.Module.Address.js
- OrderWizard.Module.Address.Shipping.js
- OrderWizard.Module.Confirmation.js
- OrderWizard.Module.PaymentMethod.Creditcard.js
- OrderWizard.Module.PaymentMethod.GiftCertificates.js
- OrderWizard.Module.PaymentMethod.Invoice.js
- OrderWizard.Module.PaymentMethod.js
- OrderWizard.Module.PaymentMethod.PayPal.js
- OrderWizard.Module.PaymentMethod.Selector.js
- OrderWizard.Module.RegisterEmail.js
- OrderWizard.Module.RegisterGuest.js
- OrderWizard.Module.Shipmethod.js
- OrderWizard.Module.ShowPayments.js
- OrderWizard.Module.ShowShipments.js
- OrderWizard.Module.TermsAndConditions.js

Reference My Account

- [My Account 1.06.0](#)
- [My Account 1.05.1](#)
- [My Account 1.05.0](#)
- [My Account 1.04.0](#)
- [My Account 1.03.0](#)
- [My Account 1.02.0](#)

My Account 1.06.0



Note: Reference My Account is available in two editions. The Bundle ID for Reference My Account 1.06.0 is **96209**. The Bundle ID for Reference My Account Premium 1.06.0 is **96162**.

With this release of Reference My Account, Bronto integration is available. Bronto is a NetSuite company that provides an advanced marketing automation engine and solutions for shopping cart abandonment, post-purchase campaigns and so forth. Bronto can be easily integrated with your Reference ShopFlow, My Account, and Checkout implementations.

The following Bronto Applications are supported:

- Cart Recovery
- Conversion Tracking
- Coupon Manager
- Pop-up Manager

To implement Bronto integration, you need to modify the Configuration.js file to include your `accountId`.

```
, bronto: {
  accountId: ''
}
});
```

With the `accountId` configured, the BrontoIntegration module causes a configuration.js file to load after the SuiteCommerce Application has initialized. This configuration.js file is cached in the Bronto CDN and uses the `accountId` to pull down the appropriate settings to enable and configure the various integrations setup in the Bronto Connector.

Note: The BrontoIntegration module essentially just provides an interface to the Bronto CDN. Therefore, any updates or enhancements to Bronto applications are completely separate from SuiteCommerce Advanced releases and will be made available through the Bronto CDN. There is no additional customization needed in the SuiteCommerce source code to take advantage of Bronto enhancements.

For detailed information on configuring and managing your Bronto applications, refer to the Bronto documentation.

My Account 1.05.1

Note: Reference My Account is available in two editions. The Bundle ID for Reference My Account 1.05.1 is **69018**. The Bundle ID for Reference My Account Premium 1.05.1 is **69017**.

In this release, a bug with Internet Explorer 8 has been fixed. The issue is a result of an unsupported regular expression character ("`^`") that causes SuiteCommerce pages to improperly render. This issue affects only customers using Internet Explorer 8. All other browsers are not affected by this issue.

To implement the fix, the **Content.EnhancedViews.js** file at `Reference js/src/app/modules/Content/Content.EnhancedViews.js` was modified with the following change:

Note: This file was also modified in ShopFlow and Checkout. If you are using those implementations, you must also make the change or upgrade to the latest bundle for each.

Previous Code:

```
, enhanceMetaTags: function (view)
{
  var enhanced_meta_regex;
  this.$head
    // then we add the description
    .find('meta[name="description"]').attr('content', view.metaDescription || view.getMetaDescription() || '').end()
    // and keywords meta tags
    .find('meta[name="keywords"]').attr('content', view.metaKeywords || view.getMetaKeywords() || '').end()
    // then we append the tags specific to this view
    // excluding any extra descriptions and keywords meta tags
    .append(view.getMetaTags().not('[name="description"]').not('[name="keywords"]'));

  /* jshint ignore:start */
  // related to issue https://github.com/jshint/jshint/issues/823
```

```

enhanced_meta_regex = /\<!\|-\ EnhanceMetaTags:STARTS[^\|]+EnhanceMetaTags:ENDS \|-\>/;
/* jshint ignore:end */
if (document.head.innerHTML.match(enhanced_meta_regex))
{
    document.head.innerHTML = document.head.innerHTML.replace(enhanced_meta_regex, '');
}

if (view.metaextra)
{
    endTo(this.$head);
}

return this;
}

```

Updated Code:

```

, enhanceMetaTags: function (view)
{
    this.$head
        // then we add the description
        .find('meta[name="description"]').attr('content', view.metaDescription || view.getMetaDescription() || '').end()
        // and keywords meta tags
        .find('meta[name="keywords"]').attr('content', view.metaKeywords || view.getMetaKeywords() || '').end()
        // then we append the tags specific to this view
        // excluding any extra descriptions and keywords meta tags
        .append(view.getMetaTags().not('[name="description"]').not('[name="keywords"]'));

    // remove head's elements that are between enhanceMetaTagsStart and enhanceMetaTagsEnd
    var remove_element = false; //flag used to determine when remove elements.

    jQuery('head').children().each(function ()
    {
        var $element = jQuery(this)
            , element_id = $element.attr('id');

        if (element_id === 'enhanceMetaTagsStart')
        {
            remove_element = true;
        }

        if (remove_element)
        {
            $element.remove();
        }

        if (remove_element && element_id === 'enhanceMetaTagsEnd')
        {
            remove_element = false;
        }
    });
}

```

```
//add meta extra elements between enhanceMetaTagsStart and enhanceMetaTagsEnd
if (view.metaextra)
{
    jQuery('<script id="enhanceMetaTagsStart"></script>' + view.metaextra + '<script id="enhanceMetaTagsEnd"></script>').appendTo(this.$head)
}

return this;
}
```

My Account 1.05.0

This document details changes to Reference My Account since the previous release. For a complete **DIFF** of files between this release and Reference My Account 1.04.0, click one of the following links to download the .patch file:

- For My Account 1.05.0: [.patch](#).
- For My Account Premium 1.05.0: [.patch](#)

Use a tool such as Crucible to open this .patch file and graphically visualize all additions, deletions, and modifications. The .patch file can also be viewed in any text editor and will display appropriate syntax highlighting when syntax is set to Diff.

Note: Reference My Account is available in two editions. The Bundle ID for Reference My Account 1.05.0 is **69018**. The Bundle ID for Reference My Account Premium 1.05.0 is **69017**.

Issue Fixes

This release contains many bug fixes including:

Credit Memos with Discount Items

Previously, when Credit Memos contained discount items with no internal id, an error was thrown when trying to view those Credit Memos in My Account. This was corrected so that these Credit Memos will display as expected. The following code changes were applied:

- **Model.fix.js:** Updated the `loaditem` method by adding the following line.

```
itemsToQuery = _.compact(_.pluck(itemsToQuery, 'internalid'));
```

- **Model.fix.js:** Updated the `setitems` method by adding the following.

```
,      description: record.getLineItemValue('item', 'description', i)
```

- **credit_memo_details.txt:** Added the following line.

```
<%= _('<span class="show-phone">Item: </span>$(0)').translate(name || item.get('description') | ' ') %>
```

Invoices when Multicurrency Disabled

Previously, with multiple currencies disabled, the currency column returned null on Invoice list pages and an error occurred when drilling down into an invoice. This has been fixed by making the following changes in the Model.js file.

Change:

```
var filters = [
  new nlobjSearchFilter('appliedtotransaction', null, 'is', receipt.getId())
,   new nlobjSearchFilter('type', null, 'anyof', ['CustCred', 'DepAppl', 'CustPymt'])
]
```

To:

```
var isMultiCurrency = context.getFeature('MULTICURRENCY')
,   amount_field = isMultiCurrency ? 'appliedtoforeignamount' : 'appliedtolinkamount'
,   filters = [
    new nlobjSearchFilter('appliedtotransaction', null, 'is', receipt.getId())
  ,   new nlobjSearchFilter('type', null, 'anyof', ['CustCred', 'DepAppl', 'CustPymt'])
]
```

Change:

```
,   new nlobjSearchColumn('appliedtoforeignamount')
]
```

To:

```
,   new nlobjSearchColumn(amount_field)
]
```

Change:

```
,   appliedtoforeignamount : toCurrency(payout.getValue('appliedtoforeignamount'))
,   appliedtoforeignamount_formatted : formatCurrency(payout.getValue('appliedtoforeignamount'))
)
```

To:

```
,   appliedtoforeignamount : toCurrency(payout.getValue(amount_field))
,   appliedtoforeignamount_formatted : formatCurrency(payout.getValue(amount_field))
```

Items on Returns not set as Display in Web Site

Previously, when an item was not set as Display in Web Site on the Item record but was part of a Return, the line on the Returns page corresponding to that item displayed undefined and no sku for the name value. This has been corrected in the Model.js file. The `getLines` function has been modified to preload all items using `StoreItem` and load inactive items manually.

Blank Credit Memo Page in Internet Explorer 8

In this release, Credit Memo pages now correctly display in IE8. Previously, they displayed blank. This was corrected by changing the name of the reserved JavaScript keyword `class` to `className` in the following files:

- `credit_memo_details.txt`
- `invoice_details.txt`

- item_details_line_macro.txt
- order_wizard_showshipments_module.txt
- quote_details.txt
- return_authorization_details.txt

Google Authorship Markup

The ability to embed the Google Authorship markup `rel="author"` or `rel="publisher"` has been added to the SuiteCommerce Advanced solution.

The following files have been modified for this addition:

- **Configuration.js:**

Added the authorship functions:

```
, seo_google_plus_authorship_author = function ()
{
    // Author for individual contents
    //return 'https://plus.google.com/+YourAuthorName';
}
, seo_google_plus_authorship_publisher = function ()
{
    // Publisher for brand contents
    //return 'https://plus.google.com/+YourPublisherName';
}
```

Extended the configuration:

```
_.extend(application.Configuration, {

    linkTagGooglePlusAuthorship: {
        'author': seo_google_plus_authorship_author
    },
    'publisher': seo_google_plus_authorship_publisher
}
```

- **SocialSharing.js:**

```
function setMetaTags ()
{
    .
    .
    .
    , link_tag_google_plus_authorship = application.getConfig('linkTagGooglePlusAuthorship');
    .
    .
    .
    clearMetaTagsByConfiguration(link_tag_google_plus_authorship);
    clearLinkTagsByConfiguration(link_tag_google_plus_authorship);
    .
    .
    .
    // In all pages
    setLinkTagsByConfiguration(self, link_tag_google_plus_authorship);
}
```

Note: Additional fixes related to meta data tags were implemented in this file. You should review the diffs for this file to ensure that you capture all changes you may require.

Adding Reorder Event Tracking

Previously, orders placed from the Cart and from My Account triggered an `add-to-cart` event. For tracking purposes, orders placed from My Account should instead trigger a `reorder` event. To correct this, the `trackEventReorderAll` method was added to the following views:

■ OrderHistory.Views.js:

```
, trackEventReorderAll: function (items)
{
    var self = this;

    0 < items.length && items.forEach(function (item)
    {
        self.trackEventReorder(item);
    });
}

, trackEventReorder: function (item)
{
    var application = this.options.application;

    application.trackEvent && application.trackEvent({
        category: 'Reorder'
        , action: 'button'
        , label: item.get('_url') + item.getQueryString()
        , value: 1
    });
}

.
.
.
, reorderAll: function (e)
{
    e.preventDefault();

    var self = this
    , add_items = []
    , application = this.options.application;

    .
    .
    .

    self.trackEventReorderAll(add_items);
```

■ OrderItem.Views.js:

```
, trackEventReorder: function (item)
{
    var application = this.options.application;

    application.trackEvent && application.trackEvent({
```

```

        category: 'Reorder'
      ,   action: 'button'
      ,   label: item.get('_url') + item.getQueryString()
      ,   value: 1
    });
  }
  .
  .
  .
  ,   orderItems: function (e)
  {
    e.preventDefault();

    var    self = this
      ,    application = this.options.application
    .
    .
    .

    self.trackEventReorder(itemToCart);
  }

```

Order by Price in Product Lists

Previously, when sorting items in Product Lists by price, some items did not correctly sort because discounts and bulk pricing values were not considered.

To correct this the following two files were modified:

- **Models.js:** added logic for in memory rules and sorting.

Change:

```

return this.searchHelper(filters, sort_column, sort_direction === '-1' ? 'DESC' : 'ASC', include_store_item);

```

To:

```

    var search_lines = this.searchHelper(filters, sort_column, sort_direction === '-1' ? 'DESC' : 'ASC', include_store_item);

    if (include_store_item && sort_column === 'price')
    {
      //-1 for descending, 1 for ascending
      search_lines = this.sortLinesByPrice(search_lines, sort_direction === '-1' ? -1 : 1);
    }

    return search_lines;
  }

  //UX expect the list to be sorted by price considering discounts and bulk pricing
  //this price is not present on datastore, so in memory rules and sorting are required.
  ,   sortLinesByPrice: function (lines, sort_direction)
  {
    'use strict';

```

```

return _.sortBy(lines, function(line)
{
    //defaults to price level 1
    var price_detail = line.item.onlinecustomerprice_detail || {}
    ,   price = price_detail.onlinecustomerprice || line.item.pricelevel1 || 0
    ,   quantity = line.quantity;

    if (quantity && price_detail.priceschedule && price_detail.priceschedule.length)
    {
        var price_schedule = _.find(price_detail.priceschedule, function(price_schedule)
        {
            return (price_schedule.minimumquantity <= quantity && quantity < price_schedule.maximumquantity) ||
                (price_schedule.minimumquantity <= quantity && !price_schedule.maximumquantity);
        });

        price = price_schedule.price;
    }

    return price * sort_direction;
});

```

- **ProductListItem.Model.js:** added the following line in the `initialize` method.

```
itemDetailModel.set('quantity', this.get('quantity'));
```

My Account 1.04.0

This document details changes to Reference My Account since the previous release. For a complete **DIFF** of files between this release and Reference My Account 1.03.0, click one of the following links to download the .patch file:

- For My Account 1.04.0: [.patch](#).
- For My Account Premium 1.04.0: [.patch](#)

Use a tool such as Crucible to open this .patch file and graphically visualize all additions, deletions, and modifications. The .patch file can also be viewed in any text editor and will display appropriate syntax highlighting when syntax is set to Diff.



Note: Reference My Account is available in two editions. The Bundle ID for Reference My Account 1.05.0 is **59233**. The Bundle ID for Reference My Account Premium 1.05.0 is **59234**.

Returns

Returns is available with Reference My Account and Reference My Account Premium Editions. With Returns, shoppers can initiate and track a Returns request from an existing Purchase Order or Invoice. The Return request is automatically linked to a purchase order or invoice. On the Returns views, shoppers can do the following:

- Submit a Return from an Order or Invoice
- Monitor Returns Process
- Cancel Returns

- Receive Email Notification

For detailed information on Returns, see the help topic [Returns](#).

Case Management


The Case Management feature enables users to submit questions or support queries directly from Reference My Account. See [Case Management](#) for more information.

Quotes

The Quotes feature enables users to receive and view quotes directly from Reference My Account. See the help topic [Quotes](#) for more information.

My Account 1.03.0

With the release of Reference My Account 1.03.0 the following enhancements have been made:

 **Note:** Reference My Account is available in two editions. The Bundle ID for Reference My Account 1.03.0 is **55275**. The Bundle ID for Reference My Account Premium 1.03.0 is **53049**.

Product Lists

When a web store is using Reference My Account 1.03.0 with this version of Reference ShopFlow, Product Lists (Wishlist) is now supported.


Using product lists, users in a B2B or B2C can do the following:

- Flag Items to be added to a product list as they are browsing in your web store.
- Create new lists when flagging items.
- Edit Item order details from within the Product Lists pages.
- Move Items between Product Lists.
- Add an entire list of products to the shopping cart with a single click.
- Use Product Lists to Reorder lists of items.

For detailed information on Product Lists, see the help topic [Product Lists](#).

Reference Product Lists Bundle

To use the Product Lists feature, you must install the Reference Product Lists Bundle. This bundle installs the custom records required to store and manage Product List data.

 **Note:** The Bundle ID for Reference Product Lists Records is **53051**.

Billing

With this version of My Account new Billing and Payments experiences are available. This enables customers in a B2B environment to do the following:

- View account balances and details
- Make partial or full payments against invoices
- View transaction history

- View receipts history
- Print Statements

To simplify the user interface, the My Account navigational menu has also been reorganized. Previously, there was a flat menu structure. This has been changed to a nested structure by default, with the new Billing features captured under a Billing menu item.

For detailed information on Billing, see the help topic [Billing](#).


Issue Fixes

This release contains many bug fixes including:

- **Enhanced Cross Site Scripting (XSS) Prevention:** All output from `request.getParameter` calls now compare URL parameters to text and the ability for users to run code through the URL parameter is eliminated.
- **Support for Printing Receipt Details:** Now users can print receipt details for an invoice from the History & Returns page All Receipts view. Previously, clicking Print from this view resulted in an error.

My Account 1.02.0


With the release of Reference My Account 1.02.0 the following enhancements have been made:

 **Note:** The Bundle ID for Reference My Account 1.02.0 is 50220.

- **Support for Permissions:** By default, My Account now honors all NetSuite roles and permissions available for the currently logged in user. You can assign permissions attributes to links so that those links do not display unless the minimum permissions requirements are met. You can customize permissions requirements as needed, and define permissions for custom blocks of code. When upgrading from an existing implementation, ensure that any custom templates are updated to reflect the new permissions attributes.

For more information, see the help topic [My Account Menu Permissions](#).

- **Forbidden Access Template:** A new `forbidden_error.txt` template is now available. This template defines the page to render when permissions are not sufficient for the currently logged in user and can be customized as needed.

 **Note:** This page is rendered only in cases where a user may have accessed a link to the page even though they do not have permissions to view that page. Normally, links do not render for these users because the links have permissions attributes tied to them.

- **Limited Fields Returned in Searches:** There is a new `backend.configuration.js` file that restricts the fields returned in order and item objects, providing enhanced performance. Previously, for Site Builder sites **all** fields for item objects were returned using the Commerce API. Now, only a subset of fields is returned. For SuiteCommerce Advanced sites, you can customize which order fields are returned using the Commerce API for Reference My Account & Reference Checkout versus Reference ShopFlow order pages. For SuiteCommerce Advanced sites, you can also specify the Fieldset ID to use for managing item fields. For both Site Builder sites and SuiteCommerce Advanced sites, you can modify the `backend.configuration.js` file as needed to return only those fields used in your web store implementation.

For detailed information, see *Configuring Backend Objects Using the `backend.configuration.js` File*

- **Multiple Versions of the Bundle can be Installed:** Multiple versions of the Reference My Account bundle can be installed and deployed to different sites. To differentiate between bundles, unique

version information is included in the Root URL and SSP application name. For example the SSP application name for this release is Reference My Account 1.02.0 and the Root URL is <https://myaccount-1-02-0.netsuite.com>.

Because the version information is included in the SSP application name, you can also differentiate bundle versions when assigning Touch Points.

■ **Bug Fixes:** This release contains many bug fixes including:

- **Additional Role Support for Viewing Orders:** Customers who are assigned **Custom** Customer Center roles can now view Orders Details and Order History & Returns in My Account. Previously, only the Customer Center role was supported. The following error was returned when a user attempted to view order information and they did not have the Customer Center role: "You have entered an Invalid Field Value null for the following field: internalid."
- **Support for Additional Customer Types:** Customers who are assigned the **Company** type can now update profile information and reset passwords in My Account. Previously, users not assigned as the **Individuals** type received the following error when attempting to update profile information or reset passwords: "Sorry, the information below is either incomplete or needs to be corrected."

Contacts are also supported only when:

1. The company has email set up in the Company record.
2. Information updated is for the Company record, not the Contact record. The exception to this is when the password is updated. If the password is updated, it is the password for the Contact record, not the Company record.

- **Reorder Items Page Errors Resolved:** When large numbers of items have been previously ordered and a customer navigates to the Reorder Items page, errors are no longer thrown. The page displays as expected.
- **Account and Site Number Parameters Added:** The following two parameters have been added to all service and Search API calls to enable access to a SuiteCommerce Advanced sites using the shopping.netsuite.com URL. For example, <http://shopping.netsuite.com/api/items?c=3649491&n=2>
 - c = Account Number
 - n = Site Number
- **Improved Combining Processes:** JavaScript files are now combined in a unified way across all Reference applications, improving performance.
- **Update to Tracking links:** Now tracking links are handled so that UPS, FedEx, and other similar links go directly to the service provider's site. Previously, links within My Account linked to Google with the tracking number.
- **Support of Auto Fill Inputs:** Chrome's form Autofill feature now correctly populates with address and credit card information. Previously, forms without the `method=POST` did not Autofill address and credit card information in Chrome browsers.
- **New HTML5 Input Types:** The `type="tel"` and `type="email"` input types have been added for all forms in My Account when referencing telephone numbers and email addresses. Now, when editing a phone number on a mobile device, only numbers are shown in the keyboard. In addition, when entering email addresses, the @ and .com options are displayed in supporting applications.
- **Enhanced Skin Support:** My Account now has its own main skin style sheet.
- **Updated Libraries:** The following front end libraries have been updated to recent versions:
 - jQuery - 1.9.1 to 1.10.2
 - Underscore.js - 1.4.4 to 1.5.1
 - RequireJS - 2.1.1 to 2.1.8

- Backbone.Validation - 0.7.1 to 0.8.0
- **Product Merchandizing Service Available:** You can now configure your My Account implementation to display Product Merchandizing zones. For detailed information, see the help topic [Product Merchandising](#) Product Merchandising.
- **Item Object Optimization:** Previously, when getting items in an order, all attributes for that item were retrieved. Now, only attributes that are to be displayed to the user are retrieved. This increases performance because approximately 15 of 200 item attributes are necessary.

Core Improvements for SuiteCommerce Advanced 2015.1

Improvement documented in this section are included in the core applications files for each Reference SuiteApp.

Enhanced Support for Multiple Languages

Previously, support for multiple languages in a SuiteCommerce Advanced web store was enabled by defining languages in the site record and then the `LanguageSupport` module handled translation strings within the application. Now support for multiple languages is handled by defining a unique domain for each language in the `backend.configuration.js` file.

To enable multiple language support, edit the `backend.configuration.js` file to include the required hosts.

```
,  hosts: [
    /*
    {
      title:'United States'
    },
    {
      title:'American Dolars'
      ,   code:'USD'
    }
  ],
  languages:[
    {
      title:'English'
      ,   host:'en.site.com'
      ,   locale:'en_US'
    }
  ]
}
```

Multiple Language support differs depending on whether you are in the ShopFlow domain versus My Account or Checkout as follows:

- When a user is in the **ShopFlow** domain and changes language, they are redirected to the correct domain based on the definitions provided in the hosts object of the `backend.configuration.js` file. If no hosts are configured, a language selector is not displayed in the web store.
- When a user is in the **My Account or Checkout** domain, the language is changed using the query string by adding the lang parameter with the locale value. The logic for this is similar to the logic provided in the `LanguageSupport` module available in previous releases although this logic is now incorporated in a new `MultiHostSupport` module.

The following modifications were made:

- **LanguageSupport Module:** This module is now obsolete and has been removed from the application.
- **MultiHostSupport.js Module:** This new module was created to handle the language selection.
- **NavigationHelper.js:** This was modified to handle redirection to the correct host when going to the Shopping domain.
- **commons.js:** This was modified so that hosts configuration are retrieved from the backend.configuration.js file.
- **backend.configuration.js:** A placeholder hosts configuration object has been added.
- **language_selector_macro.txt macro:** This macro is now obsolete and has been removed from the application.
- **host_selector_macro.txt:** Added a parameter to determine if the select should trigger a host or a language change.
- **header_top_nav.txt and footer_macro.txt:** These macros were modified to show the host selector.

Use nlapiCache for Caching Site Settings

The Models.js file has been updated to use the `nlapiGetCache` method for defining cache for the `settings` object. A default `TTL` attribute is defined. Caching of the `settings` object can be customized by redefining the `TTL` parameters.

Note: Defining the caching durations for the `settings` object is important for secure sites that do not leverage CDN caching such as Checkout and My Account, but the code is also available for Reference ShopFlow. For Reference ShopFlow pages, the CDN cache takes precedence over settings defined here.

```
Application.defineModel('SiteSettings', {
  cache: nlapiGetCache('Application')
  // cache duration time in seconds - by default 2 hours - this value can be between 5 mins a
  nd 2 hours
  , cacheTtl: 2 * 60 * 60
  , get: function ()
  {
    'use strict';
    var settings = this.cache.get('siteSettings');
    if (!settings) {

      /* .... get the site settings code here .... */

      settings = JSON.stringify(settings);
      this.cache.put('siteSettings', settings, this.cacheTtl);
    }
    settings = JSON.parse(settings);
    settings.is_logged_in = session.isLoggedIn();
    return settings;
  }
});
```

Support for Adding Meta Data from the Site Setup Record

Following are two ways that you can define meta data in NetSuite records:

- Defining Metatags in the Web Store tab of the Item Detail record (Priority 1)
- Defining Meta Keywords and Meta Description attributes in Enhanced Page records of the Content Delivery service (Priority 2)



Note: When meta data is defined in multiple places, the higher priority meta data takes precedence.

Previously, meta data tags defined in the ADDITION TO <HEAD> did not override the platform default meta tags and did not create a new set of meta tags with `class=custom`, as expected. Instead, these meta data tags only replaced meta description and meta keywords.

This has been corrected by updating the EnhancedView.js file as follows:

Change:

```
, enhanceMetaTags: function (view)
{
  var custom_meta_tag_class = 'custom';

  this.$head
    // we remove any existing custom meta tags
    .find('meta.' + custom_meta_tag_class).remove().end()
    // then we add the description
    .find('meta[name="description"]').attr('content', view.getMetaDescription() || '').end()
  ()
    // and keywords meta tags
    .find('meta[name="keywords"]').attr('content', view.getMetaKeywords() || '').end()
  //
```

To:

```
, enhanceMetaTags: function (view)
{
  var custom_meta_tag_class = 'custom';

  this.$head
    // we remove any existing custom meta tags
    .find('meta.' + custom_meta_tag_class).remove().end()
    // then we add the description
    .find('meta[name="description"]').attr('content', view.metaDescription || '').end()
    // and keywords meta tags
    .find('meta[name="keywords"]').attr('content', view.metaKeywords || '').end()
```

User Experience Improvements for SuiteCommerce Advanced 2015.1

The SuiteCommerce Advanced 2015 Release includes many User Experience improvements throughout all three applications. The following sections highlight some of the improvements.

Home Page

The Home Page now includes a default carousal and product merchandising zone.

Orders

The interface for orders has been optimized.

Products and Records Lists

Product and List records now have a single unified look with the following enhancements:

- **List Header:**
 - Unified paging controls
 - Unified positioning of global actions
 - Added an Apply to All button for actions against 2 or more items
 - Standardized sorting and filtering options and display
 - Improved date selection so that you can no longer select a date prior to today in the From field
- **List Body:**
 - The behavior and display of pages when no results are returned has been standardized
 - Standardized look and feel for states
- **Mobile and Tablet Improvements**

Summary Box

- There is now a new design for the summary box.
- The PayPal button is now displayed in a row by itself.

Action Buttons

- Action links have been replaced with buttons.
- The position of action buttons has been standardized.
- A dropdown split button is now used when multiple actions apply to an item.

Forms

Required fields are now indicated and controlled.

Terminology and Copy

- Labels now use consistent terminology for similar concepts.
- Search has been replaced with Search for Products to clarify that records are not searchable.
- Print was replaced with Download to PDF.

My Account

- Arrow icons now point up and down instead of down and right.
- The security CVV fields for credit cards now have an associated explanatory image.
- In Quote records, the Billing Address is now being displayed in the address accordion.

- In the Make a Payment flow, the Order Total Amount is now displayed in the second step.

Footer Placeholder

A footer placeholder has been added.