

Release Notes Prior to SuiteCommerce Advanced Denali



General Notices

Sample Code

NetSuite Inc. 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.

NetSuite 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, customers may not use the Service in excess of limits or thresholds that NetSuite considers commercially reasonable for the Service. If NetSuite reasonably concludes that a customer’s use is excessive and/or will cause immediate or ongoing performance issues for one or more of NetSuite’s other customers, NetSuite may slow down or throttle such customer’s excess use until such time that the customer’s use stays within reasonable limits. If a 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.

Integration with Third Party Applications

NetSuite may make available to Customer certain features designed to interoperate with third party applications. To use such features, Customer may be required to obtain access to such third party applications from their providers, and may be required to grant NetSuite access to Customer’s account(s) on such third party applications. NetSuite cannot guarantee the continued availability of such Service features or integration, and may cease providing them without entitling Customer to any refund, credit, or other compensation, if for example and without limitation, the provider of a third party application ceases to make such third party application generally available or available for interoperation with the corresponding Service features or integration in a manner acceptable to NetSuite.

Copyright

This document is the property of NetSuite Inc., and may not be reproduced in whole or in part without prior written approval of NetSuite Inc. For NetSuite trademark and service mark information, see www.netsuite.com/portal/company/trademark.shtml.

© 2017 NetSuite Inc.

Table of Contents

SCA Release Notes	1
Release Notes Prior to SuiteCommerce Advanced Denali	1
Reference ShopFlow	2
Reference Checkout	31
Reference My Account	41
Core Improvements for SuiteCommerce Advanced 2015.1	53
User Experience Improvements for SuiteCommerce Advanced 2015.1	55

SCA Release Notes

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 the help topic [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 [Site Management Tools 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
    +    nstanceof 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();
+  }
+};
-
-

```

```
-
SC.startShopping();
```

■ 'js/src/app/Configuration.js'

```
@@ -49,6 +49,7 @@
    , 'ImageLoader'
    , 'UrlHelper'
    , 'CMSadapter'
+   , 'BrontoIntegration'
    ]

    // Default url for the item list
@@ -924,6 +925,13 @@
    // ,    data_ga_tracker: '',
    }
  }
+ });
+
+ _extend(application.Configuration, {
+   useCMS: SC && SC.ENVIRONMENT && SC.ENVIRONMENT.useCMS
+ });

})(SC.Application('Shopping'));
```

■ 'js/src/app/modules/Content/Content.js'

```
@@ -9,10 +9,12 @@
  'use strict';

  return {
-   DataModels: DataModels,
-   EnhancedViews: EnhancedViews,
-   LandingPages: LandingPages,
-   mountToApp: function (Application)
+   DataModels: DataModels
+   , EnhancedViews: EnhancedViews
+   , LandingPages: LandingPages
+   , mountToApp: function (Application)
+   {
+     if (!Application.getConfig('useCMS'))
+     {
+       // Wires the models to the assets root
+       DataModels.Pages.Model.prototype.urlRoot = _.getAbsolutePath(DataModels.Pages.Model.
prototype.urlRoot);
@@ -117,5 +119,6 @@
      Backbone.history && DataModels.loadPage('// + Backbone.history.fragment);
    });
  }
+ }
  };
};
```

7. Add the path 'cmsadapter/*.txt' to the 'templates/templates.config' file.
8. Add the path 'src/app/modules/CMSadapter/*.js' to the 'js/combiner.config' file

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 [Setup 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 [Field Set Setup Script](#).

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
owser.
    </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 [Setup 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:

- [SCA Overview](#)
- [SCA Developer Tools](#)
- [Customize and Extend](#)

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 [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 [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 [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.