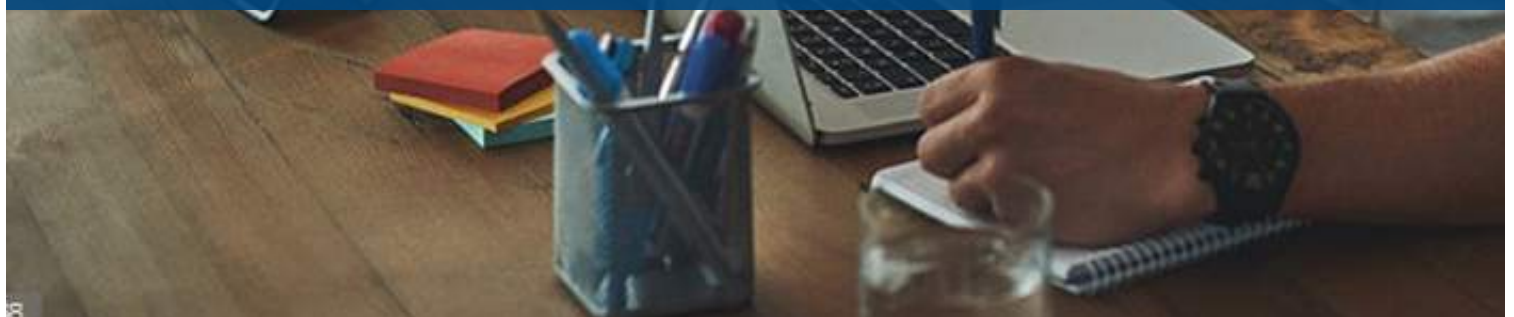




# SuiteCommerce Advanced Performance Checklist Complying to NetSuite Website Performance Standards

Version 2.8, 2017



## Table of Contents

<b>Introduction.....</b>	<b>4</b>
<b>NetSuite SuiteCommerce Advanced Performance Standards.....</b>	<b>4</b>
<b>SuiteCommerce-Specific Performance Best Practices.....</b>	<b>6</b>
a. <b>Caching .....</b>	<b>6</b>
Enabling CDN on checkout .....	8
Cache Clearing .....	9
b. <b>SEO Engine Output.....</b>	<b>10</b>
c. <b>SSP Performance.....</b>	<b>12</b>
d. <b>SuiteScript &amp; Workflows .....</b>	<b>13</b>
e. <b>SSP Environments .....</b>	<b>18</b>
shopping.environment.ssp .....	18
shopping.user.environment.ssp .....	18
f. <b>DNS Prefetch .....</b>	<b>19</b>
g. <b>Categories .....</b>	<b>19</b>
h. <b>Items API .....</b>	<b>21</b>
i. <b>Content Management .....</b>	<b>22</b>
Content delivery and CXM .....	22
Merchandizing zones fieldset .....	22
j. <b>Images Resizing .....</b>	<b>22</b>
k. <b>Single Secure Domain for Web Store Shopping and Checkout .....</b>	<b>23</b>
l. <b>Web Store Performance Enhancement – User Event Script.....</b>	<b>23</b>
Execution Context.....	23
m. <b>Web Store Performance Enhancement – User Event Script Execution Context.....</b>	<b>24</b>
n. <b>Flexibility in Specifying the Related Field Sets.....</b>	<b>25</b>
o. <b>Shopping Cart Performance Enhancement .....</b>	<b>25</b>
<b>General Performance Best Practices.....</b>	<b>26</b>
a. <b>Image Assets .....</b>	<b>26</b>
Change background to CSS instead of images .....	26
Image hosting domain.....	27
b. <b>CSS and JS optimized.....</b>	<b>28</b>
c. <b>Third-party applications.....</b>	<b>29</b>
d. <b>Optimize fonts .....</b>	<b>31</b>
e. <b>Navigation &amp; Relative URLs .....</b>	<b>31</b>

<b>f. Languages.....</b>	<b>32</b>
<b>Appendix 1 – Image Assets .....</b>	<b>33</b>
Image resizing and scaling.....	33
Compressing images .....	34
Creating an image sprite.....	35
Creating background gradients.....	35
<b>Appendix 2: Issues with third-party scripts .....</b>	<b>37</b>
Checking that third-party scripts are loaded after first-party ones .....	37
Checking that third-party requests are sent after the page has been loaded....	38
Emulating SPOF scenarios .....	39
<b>Appendix 3 - Categories.....</b>	<b>39</b>
<b>Appendix 4 – Items API .....</b>	<b>39</b>
Example .....	39
Excluding facet information through coding .....	40
Excluding facets to improve performance .....	41
<b>SEO Debug.....</b>	<b>42</b>
<b>Removing functionality from the SEO page generator .....</b>	<b>43</b>
<b>Appendix 6 - SuiteScript .....</b>	<b>43</b>
Using Content Delivery Network (CDN) cache in SuiteScript .....	43
Using Application Cache for SuiteScript.....	44
Using Cache Headers on non-session related services .....	45

## Introduction

The purpose of this checklist is to help you build websites that meet the NetSuite Performance Standards by correcting typical performance problems we see when auditing sites. In this document we will cover the most common issues that increase the loading time of a site that can be easily checked within Google Chrome with the embedded developer tools and Akamai Debug headers Chrome plugin installed and active.

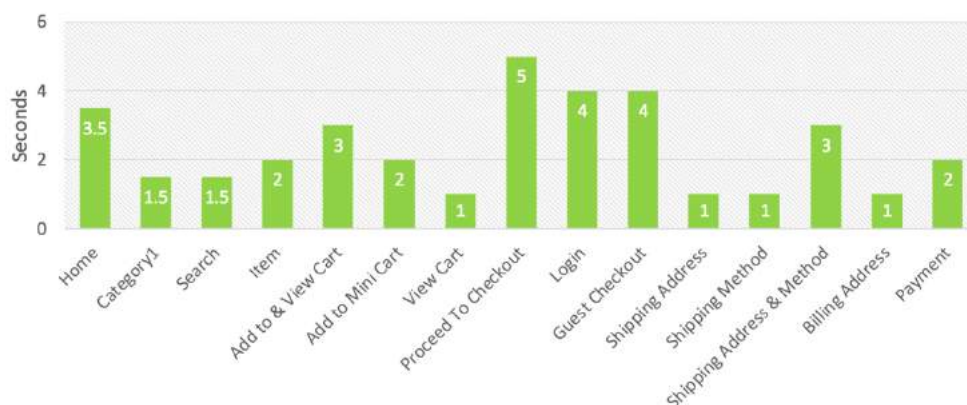
## NetSuite SuiteCommerce Advanced Performance Standards

NetSuite maintains performance standards for all sites built on the SuiteCommerce Advanced (SCA) platform. The metrics you see below are user experience time as measured by Dynatrace using Google Chrome on a desktop computer. User Experience time is the time it takes to load ALL assets on a page or step in full including the time it takes to execute all javascript. These numbers provide a guideline of how long each step should take to complete on a typical SCA website given where the platform is today assuming you are following the best practices outlined in this document.

To navigate through these 15 steps (if your particular website utilized all 15 steps) would take you 35.5 seconds to complete. To put this in perspective, our entire SuiteCommerce Advanced customers completed these exact steps on average in 35.0 seconds in the month of November 2016. While some sites are faster and some slower, this guide will help you build a faster website, and thereby improve your customer experience, conversion rates and ultimately your revenues.

Some of the steps below are simple best practices, and some can be more complex like optimizing SuiteScript or the SEO output. In either case, don't underestimate their importance. The little things add up quickly when it comes to performance.

### NETSUITE 16.1 PERFORMANCE STANDARDS



Bear in mind the NetSuite names it's steps based on the action taken – not the page you are landing on. For example, Proceed to Checkout is the timing from when a user is viewing their cart, clicks the "proceed to checkout" button and the resulting page renders and executes in its entirety.

1. Resource Caching
  - a. CDN Enabled ☐
  - b. Use CDN in checkout if Possible ☐
2. SEO Engine output
  - a. Identify and remove unnecessary XHR/Ajax calls ☐
  - b. No SEO page generation errors on the logs ☐
  - c. Remove third parties ☐
3. SSP application logs
  - a. Log level for SSP applications & scripts are set to “Error” instead of “Debug” ☐
  - b. Do not enable “scriptable cart/checkout” unless needed ☐
  - c. Enabling “asynchronous after submit sales order processing” ☐
  - d. Uninstall unnecessary bundles ☐
  - e. Undeploy unused scripts ☐
  - f. Avoid calling Suitelets from SSP libraries ☐
4. SuiteScript& Workflows
  - a. Check that SuiteScript responses are cached when possible ☐
  - b. Avoid creating too many cookies. ☐
5. SSP Environments
  - a. Remove unnecessary information from environments ☐
6. DNS Prefetch
  - a. If using older version of SCA remove line #3 ☐
7. Categories
  - a. Minimize amount of categories ☐
  - b. Remove categories from Checkout and MyAccount ☐
  - c. Change getCategories to getSiteCategoryTree ☐
8. Items API
  - a. Review fieldsets and remove unused fields ☐
  - b. Exclude facets ☐
  - c. Remove Matrix child items ☐
9. Content Management
  - a. Check fieldset in merchandizing zone = related/correlated items ☐
  - b. Ensure facets are not included in merchandizing zones’ fieldsets ☐
  - c. Move content out of content delivery/CXM and into templates ☐
10. Image Assets
  - a. Resized ☐
  - b. Compressed ☐
  - c. Change background to css instead of images ☐
  - d. Image hosting domain ☐
11. CSS and JS optimized ☐
12. Third-party applications ☐
13. Optimized fonts ☐
14. Languages ☐
15. Using relative URLs instead of absolute URLs ☐

## SuiteCommerce-Specific Performance Best Practices

### a. Caching

There are 2 sources of resource caching on SCA: application-layer caching and content delivery network (CDN) caching.

Our CDN is a large, geographically distributed network of specialized servers provided by Akamai that accelerate the delivery of web content and rich media to internet-connected devices. Application caching works in a similar way as CDN caching, but at the NetSuite application server layer, storing its contents into Redis.

There is a separate application cache for compiled JavaScript code. It contains all compiled JavaScript code, which has been used during SEO page generation.

Every SSP page (and service call (.SS)) is responsible for its caching header. If the author of an SSP or SS file wants to add a caching header, they must add the following line into the code of the SSP file:

```
<% response.setCDNCacheable(response.CACHE_DURATION_LONG); %>
```

The following table shows currently available CDN cache life values

SuiteScript constant	CDN Header: Edge-Control	TTL in Application cache (Redis)
response.CACHE_DURATION_LONG	cache-maxage=7d	Maximum of 2-3 hours
response.CACHE_DURATION_MEDIUM	cache-maxage=2h	2 hours
response.CACHE_DURATION_SHORT	cache-maxage=5m	5 minutes
response.CACHE_DURATION_UNIQUE	no-store	0



How to know if a resource is being cached using AKAMAI debug headers plugin installed and active in Chrome browser:

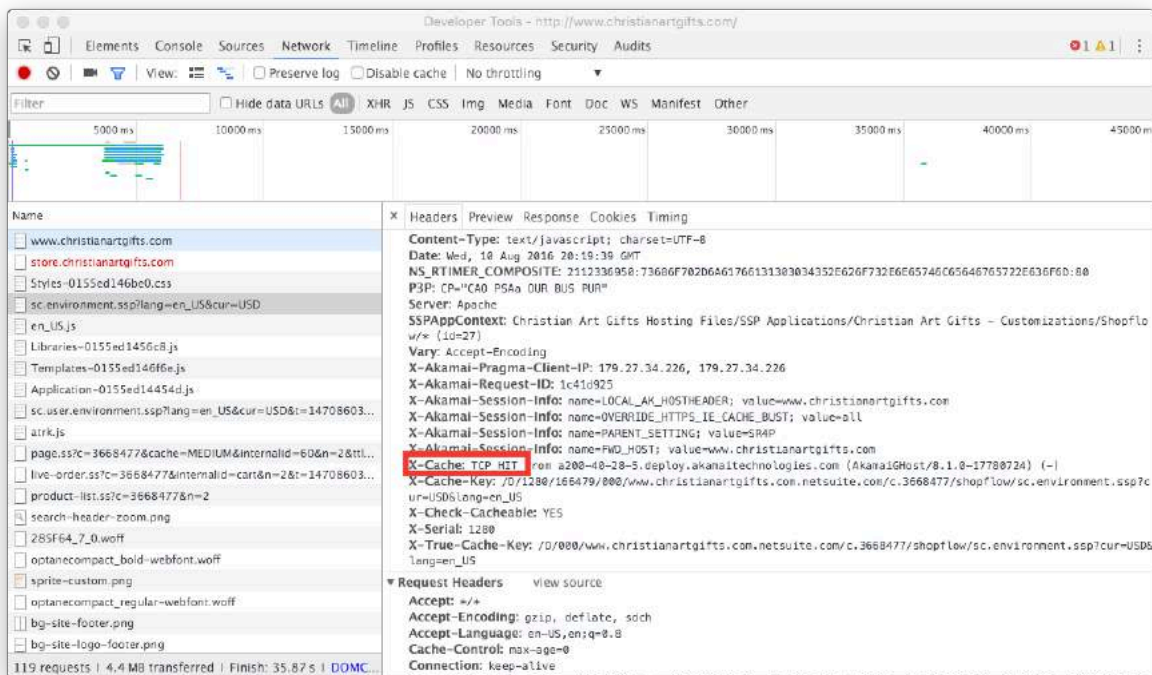


Figure 1. X-Cache: TCP\_HIT

Akamai contains two tiers of nodes: edge nodes (local to the end user) and parent nodes (shared by several edge nodes). User requests are received and handled by edge nodes. When a request is made by a client for an asset, it is handled by an edge node. If the edge node has the asset in cache (and if the asset is not expired), it will be returned immediately from cache to the client. If the edge node can't fulfil this request, then it will propagate this request up to its parent node. If the parent node has the asset in cache, it will be returned to the edge node which will cache it for future use, and the edge node will return it to the client. If the parent node can't fulfil this request, then it will propagate the request up to the origin server (Netsuite). The origin then returns the asset to the parent node which caches it (if applicable) and returns the asset to the edge node. The edge node will now cache the asset (if applicable) and return it the asset to the end client. All of this happens automatically provided you specify your cache header in your SSP or SS file, e.g.:

```
<% response.setCDNCacheable(response.CACHE_DURATION_LONG); %>.
```

In Figure 1 we can see a local CDN edge node cache hit. We know this because the X-Cache header contains the value "TCP\_HIT". If there is a miss on an edge node and a hit on a parent node, we will see "TCP\_MISS" on this header instead, and a "TCP\_HIT" value on the X-Cache-Remote header.

If the resource is not cached on either local or parent nodes, then the resulting response will have "TCP\_MISS" values on both its X-Cache and X-Cache-Remote headers. If the cached asset is stale (beyond it's cache life), but present, the CDN will return "TCP\_MISS\_REFRESH".

Please note: the cache life you choose for an SSP application applies to **all** pages rendered by that SSP application. For example, a homepage and an item page and rendered by the same SSP (index.ssp or shopping.ssp) file. That said, each URL is cached individually.

### Potential performance problems:

CDN may not be enabled for some or all domains shown on the website setup page. This could be due to:

- The “Enable CDN” checkbox isn’t ticked for the website domain(s) or more likely:
- The domain (CNAME) isn’t configured correctly (which is a common mistake).

### Action Item(s):

- Configure your CNAME correctly at the domain provider level using the CNAME alias generated by Netsuite and shown in your website domain setup in Netsuite.
- Don’t host your website at the domain root, i.e. <http://mywebsite.com> because the CDN configuration will break email services to that domain. Always use a hostname such as “www” for your website. i.e. <http://www.mywebsite.com> and point this host using a CNAME to NetSuite. Be sure to follow NetSuite guidance on this correctly using the HELP inside NetSuite.

### Further Information:

- <https://developers.suitecommerce.com/til-thursday-troubleshooting-dns-problems-with-the-cdn>

### Enabling CDN on checkout

As of 16.2, SuiteCommerce supports using CDN on checkout for customers using their own secure domain. Simply turning this feature on will benefit customers by serving static assets directly from the CDN. CDNs provide a means of placing content closer to end users thereby reduction round trip time to NetSuite servers. User specific assets that cannot be cached will be requested by the CDN directly from NetSuite and served to the end user over HTTPS.

### ADVANCED:

Customers who have development resources on hand can further maximize caching on the CDN setting cache control headers in the SSP and SS calls using:

```
<% response.setCDNCacheable(response.CACHE_DURATION_MEDIUM); %>
```

This requires splitting up files into cacheable and non-cacheable components. For example checkout.environment.ssp today contains both user specific and site specific details. If the file was split into two, the latter details could be cached on the CDN as a separate request.

CDN on checkout has a few implications one might not be aware of:

- As checkout is HTTPS-only, customer's SSL certificate (and key) **MUST** be uploaded to CDN provider
- Private information (payment info) is proxied through a third-party



**Potential performance problems:**

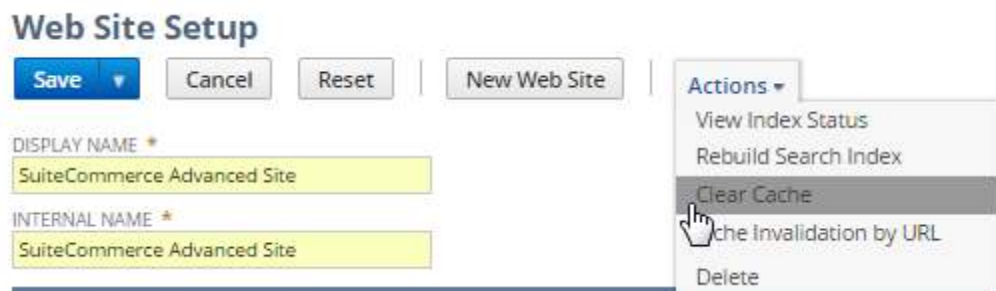
- Slow loading time for checkout steps

**Action Item(s):**

- If having a vanity checkout domain, enable checkout CDN and configure the DNS to point to the correct CNAME

**Cache Clearing**

As you develop your SuiteCommerce Advanced web site, you may need to redeploy an SSP application. To see changes to content delivered from the file cabinet, script files and Item Search API request output, use the Clear Cache link.



After you click Clear Cache, a popup screen displays, confirming that you want to clear the cache. When you click OK in the popup screen, your request is queued for processing. Note that it can take a few hours for this to complete fully.

You can submit one request to clear cache at a time per site. After the request is submitted, the Clear Cache link is disabled. If you return to the Web Site Setup page before your request has been processed, you will see the date and time of the previous request in the More Action menu instead of the Clear Cache link. You can only submit a new cache invalidation request after the previous request has been processed.

Because most data is cached in SSP and SuiteScript method calls, you can submit a cache invalidation request using the URL that points to an SSP or SuiteScript file. Use the Cache Invalidation by URL page to specify the SSP or SuiteScript file that contains the definition for the content you want to clear from the cache.

For example, you may have a banner on your website that you plan to change each day. By default, this will be cached for 7 days. To replace this banner image with a new one, you can use the cache invalidation by URL feature to purge the image directly. Alternatively, you may need to purge the SSP file that includes a definition for the banner on your site for example, /page.ss. On the Cache Invalidation by URL page, you would send the cache invalidation request using your website domain along with a URL similar to the example below:

**Cache Invalidation by URL**

Save Cancel

DOMAIN NAME ★  
account. .com

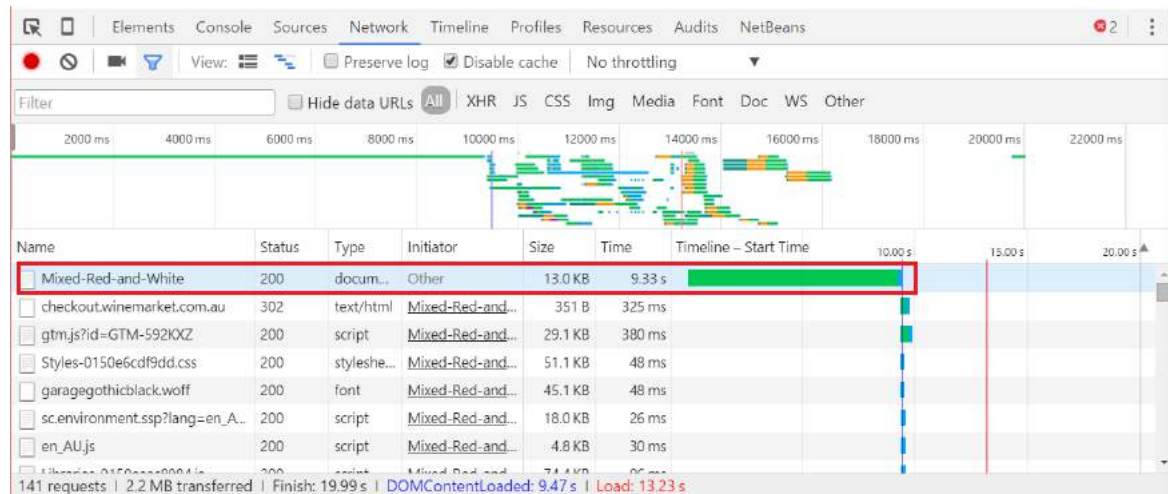
URL  
/

**b. SEO Engine Output****Background:**

SuiteCommerce Advanced consists of 3 single page applications (SPAs) – shopping, checkout and my account. The entire application is JavaScript, and since most search engines cannot execute JavaScript well (or at all) – SuiteCommerce runs all direct requests for site pages through a headless browser server side (called the SEO Generator) to execute all scripting and service calls and output that as pure HTML. That HTML is passed to the browser on most directly navigated pages (typing a URL into the address bar and pressing enter) so search engines that are spidering a site can read its content. After this HTML is pushed down to the client machine it is replaced with the application JavaScript for actual users. Most of the time this HTML output is cached on Akamai (our primary CDN) for a LONG cache life of 7 days, and so loads quickly to the user or search engine spider. However, URLs that are rarely visited or new sites while in development (without much traffic) can have pages that are not cached at the CDN. These URLs when directly navigated to will trigger the SEO generator to execute. This is provided they are shopping URLs, not checkout or my account – the SEO generator doesn't execute for checkout or my account. Executing all the content on a page and converting it from Javascript can be time consuming. As such you want to always ensure that a) your SEO output is error free, and b) is as streamlined as possible. You can do this by excluding everything that isn't absolutely required to be in the SEO output of a page from a search engine perspective. Third party content being just one example of this.

The SEO engine can be a bottleneck on first page loading times of non-cached pages. Luckily, its components can easily be measured and optimized through a careful analysis of the SEO debug log output.

SEO page generation times can be measured through a tool like webpagetest.org, or Chrome's dev tools as depicted in the image below. In this example, the long server response time is not due to the server being slow, but the fact that the SEO generator is executing server side to render the content, and the content has not been optimized for the generator.



Since, the SEO engine supports up to 96MB please ensure that the total size of your webpages do not approach this limit. Be mindful of polling or looping requests that might continually balloon the size of your page.

### Potential performance problems:

- There is a memory limit of 96MB per page on your site; if you exceed this, it will cause an error. Note that this limit includes objects created by JavaScript: thus you may look at your files and think it's below 96MB, but the memory demands of the page may still exceed this.
- The page generator has a time limit of 30 seconds to build the page based, and this includes executing all scripts and code on the page, and downloading any third-party resources that are required. If, after 30 seconds, it is not built then an error is served.
- There is a loop in the code and this causes the SEO engine to exceed its limit and timeout.
- You have errors in your SEO output.

### Action Item(s):

- Execute the SEO debugger on the pages specified on the checklist (home page, category pages, search results and product detail pages (PDPs))
- Make sure that there are no SEO page generation errors on the logs. Use the ?seodebug=T parameter to check. Here are some examples:
  - [view-source:www.mywebstore.com/?seodebug=T](http://view-source:www.mywebstore.com/?seodebug=T)
  - [view-source:www.mywebstore.com/Salida-Backpack?seodebug=T](http://view-source:www.mywebstore.com/Salida-Backpack?seodebug=T)
- Identify and remove unnecessary XHR/Ajax calls
  - Remove third-party requests from the SEO page generator whenever possible
  - Remove unnecessary functionality from the SEO page generator such as product lists.
- Exclude content from the generator where not required:
  - See [Appendix 5](#) for instructions.
- Make sure that the site takes less than 30 seconds to build the page
- Ensure that page does not have polling or looping that causes your page weight to continually balloon until it exceed 96MB
- Make sure that the setTimeout() and setInterval() methods are not used in your JavaScript

- Avoid using the append() and animate() jQuery methods as they might cause problems with the SEO page generator.

#### Further Information:

- [Appendix 5](#)
- <https://developers.suitecommerce.com/coding-and-debugging-with-the-seo-page-generator>
- <https://developers.suitecommerce.com/section4054469891#procedure4054649256>

### c. SSP Performance

SSP applications and scripts could be logging extensive amounts of information. It is suggested that log level for all SSP applications and scripts are changed to “Error”, and that recurrent log entries generated through nlapiLogExecution calls are traced and removed.

Sample SSP Application log:

The screenshot shows the NetSuite SSP Application configuration page for 'Shopflow'. The 'Execution Log' tab is selected, displaying a table of log entries. The log level is set to 'Debug', which is highlighted with a red box. The table shows multiple log entries for the script 'psg\_dm.getMerchRules' with a log level of 'Debug'.

VIEW	TYPE	TITLE	DATE	TIME	USER	DETAILS	REMOVE
View	Debug	psg_dm.getMerchRules	8/19/2015	12:37 am		<pre>END psg_dm.getMerchRules: nlapiGetContext().getRemainingUsage(): 931 ***** </pre>	Remove
View	Debug	psg_dm.getMerchRules	8/19/2015	12:37 am		<pre>START psg_dm.getMerchRules: nlapiGetContext().getRemainingUsage(): 933 ***** zone : undefined </pre>	Remove
View	Debug	psg_dm.getMerchRules	8/19/2015	12:34 am		<pre>END psg_dm.getMerchRules: nlapiGetContext().getRemainingUsage(): 931 ***** </pre>	Remove
View	Debug	psg_dm.getMerchRules	8/19/2015	12:34 am		<pre>START psg_dm.getMerchRules: nlapiGetContext().getRemainingUsage(): 933 ***** zone : undefined </pre>	Remove
View	Debug	psg_dm.getMerchRules	8/19/2015	12:19 am		<pre>END psg_dm.getMerchRules: nlapiGetContext().getRemainingUsage(): 931 ***** </pre>	Remove
View	Debug	psg_dm.getMerchRules	8/19/2015	12:18 am		<pre>START psg_dm.getMerchRules: nlapiGetContext().getRemainingUsage(): 933 ***** zone : undefined </pre>	Remove
View	Debug	psg_dm.getMerchRules	8/19/2015	12:09 am		<pre>END psg_dm.getMerchRules: nlapiGetContext().getRemainingUsage(): 931 ***** </pre>	Remove
View	Debug	psg_dm.getMerchRules	8/19/2015	12:09 am		<pre>START psg_dm.getMerchRules: nlapiGetContext().getRemainingUsage(): 933 ***** zone : undefined </pre>	Remove

#### Potential performance problems:

- When log level is in “Debug”, it logs every transaction made in the site, this can mean that massive amounts of information are being logged regularly.
- Elevated Permissions not enabled. Users have to go through suitelets to elevate permissions to talk to record types that the shopper role doesn’t have access to.

#### Action Item(s):

- Ensure SSP application and scripts log levels are set to “Error”.
- Instead of using Suitelets to access certain record types, try enabling elevated permissions for your shopping SSP. See NetSuite Help article:
  - [https://system.netsuite.com/app/help/helpcenter.nl?fid=section\\_4589350481.html](https://system.netsuite.com/app/help/helpcenter.nl?fid=section_4589350481.html)

## d. SuiteScript & Workflows

Perhaps one of the most misunderstood and greatest opportunities to improve performance is SuiteScript and Workflows. SuiteScript and Workflows provide enormous customization capabilities for both the NetSuite application, SuiteCommerce Advanced (SCA), and SuiteCommerce In-Store (SCIS). However, using this capability without understanding it can also significantly hurt your performance. Since we are focusing on SuiteCommerce Advanced in this document – we are going to focus on the three record types that SCA utilizes: Sales Order, Customer and/or Lead records. SuiteScripts and Workflows against these record types all have a performance cost. When you install bundles into your account – even those created by a third party, those bundles may include SuiteScripts or Workflows against these record types which will typically trigger in the webstore and slow down any or all of these steps:

- Add to mini cart
- Proceed to checkout
- Log in
- Register as new customer
- Guest checkout
- Submit order

### VERY IMPORTANT:

Therefore, it is critical to uninstall bundles, undeploy SuiteScripts and Workflows you don't absolutely require; and for those that you do require – make sure that a) you understand their performance cost, and b) ensure they have been properly optimized.

Lastly, if your SuiteScripts are required for the NetSuite UI, but not in your webstore – change the deployment to be “ALL EMPLOYEES” instead of “ALL ROLES”. The setting “ALL ROLES” includes the webstore shopper role.



Similarly, if your Workflows are not required to be executed by the webstore, deselect it from the workflow context field:

The screenshot shows the 'Workflow' configuration page in NetSuite. Under the 'Event Definition' tab, the 'ON CREATE' checkbox is checked. The 'TRIGGER TYPE' is set to '- All -'. The 'EVENT TYPES' field is empty. In the 'CONTEXTS' dropdown, 'Web Store' is selected. The 'CONDITION' field contains 'Custom Form = null'. The 'SAVED SEARCH CONDITION' dropdown is also empty. Below the 'Event Definition' section, the 'Fields' and 'History' tabs are visible, with 'SHOW INACTIVES' unchecked.

For example, if you have a workflow based on the sales order record type, and you set the workflow initiation Context Type to Web Store, this means that only sales orders submitted through the Web Store will initiate the workflow. Sales orders submitted through the Web Services, CSV Import, or any other context will not initiate the workflow.

### Potential performance problems:

- When log level is in “Debug”, it logs every transaction made in the site, this can mean that massive amounts of information are being logged regularly.
- Suitescripts or workflows against sales order will impact in performance for Site Builder or SCA.
- In order to view scripts deployed against the records:

The screenshot shows the 'Scripted Record' page in NetSuite. The 'NAME' field is set to 'Customer'. Below the page, there is a table with columns 'SCRIPT', 'API VERSION', 'STATUS', and 'BEFORE LOAD'. The table is currently empty with the message 'No records to show.' To the right, the 'Customization Manager' dropdown menu is open, showing a hierarchy: 'Scripting' > 'Scripted Records'. Other options in the menu include 'Script Deployments', 'Scheduled Script Status', 'Map/Reduce Script Status', 'Script Execution Logs', 'SSP Applications', 'Script Debugger', and 'Workflows'.

- Clients scripts – These are not running in the web store unless scriptable cart is on
- User events – These are not running in the web store unless all employees is checked (This also runs for client Scripts)
- Sales order – By selecting Employees, it will allow the script to run only in NETSUITE, not in the webstore. If All Roles is selected it will run in the webstore as well.



- Suitelets – Sometimes a customer needs to go to particular type of record in Netsuite to perform some functions and those records are not available through the API. In this case the permissions of the SSP Applications should be elevated. Customers shouldn't be needing to use Suitelets.

- The type of form used can slow down performance due to templates with customizations.

- When scripting, avoid using the validateLine() function, instead putting all logic in the recalc() function. The recalc() function is called only once after all items are added to the order. The validateLine() function is called once for every item on the order thus potentially impacting performance.
- User events and client scripts can be installed by other bundles (also third parties).
  - This should be taken into account since it will add time which won't be reduced. Example: Avalara can add 2 seconds to the total time.

**Scripts** View Deployments

[New Script](#)

**FILTERS**

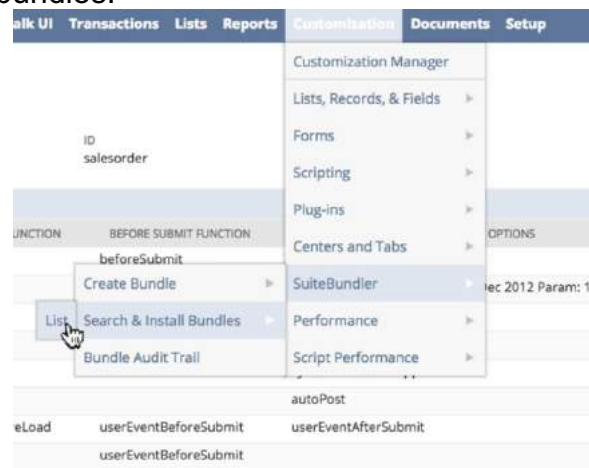
TYPE: - All - API VERSION: - All - SCRIPT FILE: - All - FROM BUNDLE: - All -

	FROM BUNDLE	ID	TYPE	API VERSION	DEPLOYMENTS	SCRIPT	LIBRA
View	3127	customscript_3127_submit_online	Suitelet	1.0	Deployments	3127.G8.online.submission.js	3127
View	Match	customscript_3way_ws_checksSubsMatchField	Workflow Action	1.0	Deployments	WD_NAW_3WAY_WS_checksSubsMatchField.js	
View	43003	customscript_8375_bundle_install	Bundle Installation	1.0	Deployments	8375_bundle_install.js	
Edit   View		customscript172	RESTlet	1.0	Deployments	abacus_rest.js	
Edit   View		customscript186	User Event	1.0	Deployments	custombutton.js	

- User-event SuiteScripts deployed against the Sales Order, Customer or Lead records or forms used by the web store can affect cart and/or checkout times with or without “scriptable cart/checkout” being turned on.
- Client SuiteScripts deployed against the Sales Order, Customer or Lead records or forms used by the web store can affect cart and/or checkout times only if Scriptable Cart/Checkout is turned on.
- Adding SuiteBundles that contain the above script types will affect website performance.

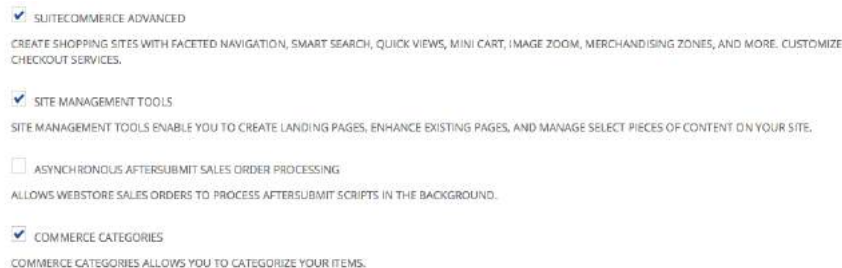
## Action Item(s):

- Add as much as possible in scheduled scripts so it doesn't add time to the process and its done asynchronously.
- Uninstall unused bundles:



- Go back to the default form to see how it used to perform before.
- Cache SuiteScript responses when possible.
  - Use `nlapiGetCache` as highlighted above (do not cache user content).
  - Alternatively, wrap Suitelets in a `ss` that does `nlapiResolveURL` and `nlapiRequestURL`. Service calls (`.ss`) can be cached in both the CDN and Redis (do not cache user content).
  - If the aforementioned approaches are not possible due to having user content, cache the results on session (`context.setSessionObject`).
  - Use cache headers on non-session related services.
- Keep SuiteScript responses lightweight when called from the front end. For example, typically there is no need to pass all columns for a given record.

- Avoid creating too many cookies. They travel back and forth on every request for a domain.
- Do not enable “scriptable cart/checkout” unless you require its capabilities.
- Enable “asynchronous after submit sales order processing” on the Company’s Enable Features page. This setting causes all afterSubmit user events and workflows triggered by web store checkout to run asynchronously.



- Uninstall any bundles that are not being currently used.
- Undeploy scripts and workflows which are not being used.
- Check roles for client scripts deployed against the sales order
  - Scripts which should execute for back end users but not web store customers should be deployed to "all employees" instead of "all roles".
- Optimize user events and client scripts deployed against the sales order, customer or lead records.
  - Substitute client scripts and user events for Scheduled Scripts where possible.
  - Avoid multiple after submit scripts that load and submit the same record.
  - Avoid using the validateLine() function, instead putting all logic in the recalc() function. The recalc() function is called only once after all items are added to the order. The validateLine() function is called once for every item on the order.
  - Use the SuiteScript caching layer to mitigate roundtrips to the DB.
- Check that back end code on live order model is fast enough.
  - Avoid calling Suitelets from SSP libraries.
- Keep your web store sales order form and your default customer form minimal by removing all fields that are not necessary for said purposes - both custom and native.
  - Look out for scripting at the form level. Avoid custom sub lists on these forms.

## Further information:

For best practices on optimizing SuiteScript performance please review the SAFE document:

- [https://system.netsuite.com/core/media/media.nl?id=30101418&c=NLCORP&h=c5f0cd3c54906c472c07&\\_xt=.pdf](https://system.netsuite.com/core/media/media.nl?id=30101418&c=NLCORP&h=c5f0cd3c54906c472c07&_xt=.pdf)

## e. SSP Environments

### [shopping.environment.ssp](#)

Checking the output size of shopping.environment.ssp is critical as this resource can take a long time to generate in cache miss situations. Standard contents of shopping.environment.ssp are site settings, categories, default session information, review configuration, content delivery entries, CMS entries and some extra settings.

The default size for this file is 45kb (around 13kb gzipped) but its size grows considerably once categories and content delivery entries start being added. This file should always be cached.

#### **Potential performance problems:**

- shopping.environment.ssp is not being cached on the CDN when it should always be cached.
- Loading unnecessary information, resulting in higher time for the loading of the file – especially for the SEO generator.

#### **Action Item(s):**

- Make sure that caching this resource works correctly. All content in this resource is expected to be cached so review that no non-cacheable content is added here (see [caching resources section](#)).
- Only store cacheable information that is used on all pages and is required at load time in shopping.environment.ssp.
- Measure the size of your shopping.environment.ssp. If the size exceeds 150kb, perform a thorough review of this file.
- Make sure that all the information stored here is needed. If it is needed only on a particular page, consider loading it on demand.
- Categories information should not contain unnecessary information such as category descriptions or images. Consider loading these on demand if the file is too large (see the section on [categories](#))

### [shopping.user.environment.ssp](#)

This file is never cached. Standard contents of shopping.user.environment.ssp are profile information, session information, product list information and cases.

#### **Potential performance problems:**

- Unnecessary information being included in the file

#### **Action Item(s):**

- All cacheable, non-user-specific information should be moved into sc.environment.ssp.

## f. DNS Prefetch

DNS pre-fetching is a technique to improve performance when loading subsequent pages that are on a different domain. 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 in older versions of SCA:

```
if (typeof nsglobal === 'undefined') {  
    var prefetch_url = 'https://example.securedcheckout.com';  
    document.write('<link rel="prefetch" href="' + prefetch_url + '>');  
    document.write('<link rel="dns-prefetch" href="' + prefetch_url + '>');  
}
```

This was designed to encourage the browser to perform a DNS lookup on the checkout domain (in this case “example.securedcheckout.com”), to speed up the transition to checkout. When this code was written, not all browsers supported the “dns-prefetch” method. To work around this, the “prefetch” method was also used.

However, “prefetch” encourages the user agent to make a full HTTP request and load the file into cache that resides at the given URL. As no file typically resides at the location defined in this code, the response is a 404 error. This doesn’t negatively impact user experience, but it does waste a round-trip and generate a needless error in the logs.

### Potential performance problems:

- DNS Prefetching shouldn’t impact end-user experience. It does cause an unnecessary error and waste a server round-trip. (Note that 404 responses aren’t cached by the CDN, so these requests are routed via the CDN back to origin and can be slow.)

### Action Item(s):

- If you are running an older version of SuiteCommerce Advanced, remove line #3 (“prefetch”), leaving “dns-prefetch” intact. Removing the entire block is also a valid action, as the benefit to this code is extremely small.

## g. Categories

Categories are a common source of performance issues. This is due to the fact that the Reference SuiteCommerce Advanced application bootstraps the whole category structure by default in all 3 single page applications (SPAs): Shopping, Checkout and MyAccount.

Keep in mind that it was designed this way because it can make no assumptions on the size of said category tree as it varies significantly from customer to customer, and that removing categories when not needed is much simpler than adding them when needed.

In general, project-by-project decisions need to be made in order to optimize categories for each use case. Note that categories are not needed on login and checkout pages for most projects.

In general, the bootstrapped category structure in `environment.ssp` is included in order to:

- Render the navigation menu
- On a category landing page, show its title, description, image and subcategories. However, this data can be retrieved on demand. Such is the approach taken by `PSCategories.ss` (built by Netsuite's PS Efficiencies team).

Please Note: There are two types of categories in SuiteCommerce Advanced (SCA). SuiteCommerce Vinson released after 16.2 includes Commerce Categories that can be used by CXM. Prior to this release, categories from the SiteBuilder product could be used via `PSCategories.ss`.

**Potential performance problems:**

- First page load is severely affected by the time it takes to bootstrap the whole category structure.

**Action Item(s):**

- The top-level category information must contain only what is necessary for site navigation.
- Information such as mark-up, templates or images should be removed from this structure and should be retrieved from a specific service (`get-category.ss`). The only fields that should be found in the category tree under `environment.ssp` are the name and category id.
- Minimize the amount of categories returned for each product listing page. Ideally, each product listing page will return information for a single branch of the category tree.
- Ensure top level categories are cached under `sc.environment.ssp`.
- Remove categories from Checkout and My Account unless needed.

**Further Information:**

- [Appendix 3 - Categories](#)



## h. Items API

It is a very common mistake to include fields that are never used in the item API calls. Every additional field has a performance cost and some more than others. Therefore unused fields should be removed from fieldsets to improve the performance of the site.

Another great improvement in the Items API can be to exclude the facets when they are not needed, for example when we are in the Product Detail Page (PDP), and we are not displaying the facets, there is no need to include these in the API call.

The standard call for Matrix child items demands a great amount of time to load as it returns all field for each matrix child item. Therefore, it is better to use fieldsets for item options instead of this call.

Lastly, per page rendered there should only be two API calls: a) one for item details, and b) another for related or correlated items. In some cases, you may wish to use both in which case there would be three API calls on a PDP. There should only be one Item API call on a category or product listings page (showing multiple items).

### **Potential performance problems:**

- Including unused fields in fieldsets delaying the loading time.
- Not excluding facets
- Use of Matrix Child Details
- Several calls to Items API.
- Too many facets

### **Action Item(s):**

- Remove unused fields from fieldsets
- Exclude facets when not needed on PDPs.
- Remove matrix child items. Use fieldsets instead.
- Ensure you are calling the Item API the correct number of times per page type.
- Keep the number of facets to a minimum. OOTB limit is 40.

### **Further Information:**

- [Appendix 4 – Items API](#)

## i. Content Management

### Content delivery and CXM

Excessive content delivery use can have negative consequences on the web store's performance as it involves services, which can take a long time - twice: both on the client side and the SEO page generator.

**Potential performance problems:**

- When creating a header/footer using content delivery instead of templates, it can slow down the pages' loading
- A lot of JavaScript is introduced on top of the web applications which is also rendered by the SEO Engine. This can mean an increase in the loading times.

**Action Item(s):**

- Move as much content as possible out of content delivery/CXM and into templates.
- Minify content delivery output
- Postframe.js is now included in cms.js so it should not be called

### Merchandizing zones fieldset

In case of existing a merchandizing zone, check the fieldset which load the items, it should be loaded with a call to the related-items or co-related-items instead of a custom search.

**Potential performance problems:**

- Custom search used to load merchandizing zone
- Other fieldset loading more fields than needed used to load the merchandizing zone.

**Action Item(s):**

- Merchandizing zones' items should be loaded using related-items or co-related-items and it should contain only the needed fields.

**Further Information:**

- <https://developers.suitecommerce.com/sectionN2687121>

## j. Images Resizing

SuiteCommerce Advanced includes an image resizing service, which works by appending parameters to a product image's URL.

Image sizes are configured in the NetSuite application under the advanced section of the website's settings. For details on configuring this, see:

<https://developers.suitecommerce.com/sectionn2664101#bridgehead3836967289>

**Potential performance problems:**

- Images are not resized and are therefore being loaded with higher resolution than needed.

**Action Item(s):**

- Ensure that the resizing service is being used appropriately for product listing pages (see link above).
  - Check that *resizeid*, *resizeh* and *resizew* parameters are being appended to product images throughout the site (product listings, merchandising zones and product detail pages)
- Ensure that images are not scaled in the browser. In other words, check that the resized images match their container size (see Appendix 1).
- Product listing page thumbnail dimensions should not be bigger than 250px.
- Ensure that the size of raw product images on the file cabinet does not exceed 5 megapixels.

**Further Information:**

- Appendix 1 [Image resizing and scaling](#).

## k. Single Secure Domain for Web Store Shopping and Checkout

Release 2017.1 includes the ability to set up your shopping domain as secure. Now you can leverage this security feature to run both shopping and checkout under a single HTTPS domain. A single domain provides consumers with a seamless buying experience. It removes the delays associated with shopping in one domain and moving to another for checkout.

## l. Web Store Performance Enhancement – User Event Script

**Execution Context**

For NetSuite accounts that use SuiteCommerce, NetSuite 2017.1 includes a new option that can help improve web store performance. This option is available for all user event script deployments. It controls whether a script executes when the triggering event takes place in the web store. The Execute in Commerce Context option appears as a check box on script deployment records. When this box is cleared, the script does not execute in response to a triggering event that occurs in SuiteCommerce Advanced, SuiteCommerce Site Builder, or SuiteCommerce InStore. When your account is upgraded to 2017.1, this box is added and automatically checked on all existing user event script deployment records. However, if appropriate, you can edit any script deployment record and clear the box. In contrast, when you create a new script deployment record in 2017.1, the box is cleared by default.

Scripts can significantly slow web store performance. By leaving this option cleared, you can improve web store response times. However, in some cases, you may want a script to execute in response to web store activity. To enable the option, you must check the Execute in Commerce Context box. For details about working with script deployment records, see the help topic Script Deployment. Previous releases of NetSuite also included features that let you customize the impact of scripts on the web store.

Other existing features include:

- Asynchronous afterSubmit Sales Order Processing – When you enable this feature, all afterSubmit user events and workflows triggered by web store checkout run asynchronously. For details, see the help topic Enabling Ecommerce Features.
- Scriptable Cart – This option lets you designate specific forms to be used for sales orders generated through shopping cart activity. You can customize these forms, and any scripts they use, for the web store. For details on this feature, see the help topic Scriptable Cart.

### m. Web Store Performance Enhancement – User Event Script Execution Context

For NetSuite accounts that use SuiteCommerce, NetSuite 2017.1 includes a new option that can help improve web store performance. This option is available for all user event script deployments. It controls whether a script executes when the triggering event takes place in the web store.

The Execute in Commerce Context option appears as a check box on script deployment records. When this box is cleared, the script does not execute in response to a triggering event that occurs in SuiteCommerce Advanced, SuiteCommerce Site Builder, or SuiteCommerce InStore.

When your account is upgraded to 2017.1, this box is added and automatically checked on all existing user event script deployment records. However, if appropriate, you can edit any script deployment record and clear the box.

In contrast, when you create a new script deployment record in 2017.1, the box is cleared by default.

**Script Deployment**

SCRIPT  
Sales Order After Submit

APPLIES TO   
Sales Order

APP ID  
customscript\_soaftersubmit

ID

☒ DEPLOYED

☐ EXECUTE IN COMMERCE CONTEXT

Scripts can significantly slow web store performance. By leaving this option cleared, you can improve web store response times. However, in some cases, you may want a script to execute in response to web store activity. To enable the option, you must check the Execute in Commerce Context box. For details about working with script deployment records, see the help topic Script Deployment.

Previous releases of NetSuite also included features that let you customize the impact of scripts on the web store. Other existing features include:

- Asynchronous afterSubmit Sales Order Processing – When you enable this feature, all afterSubmit user events and workflows triggered by web store checkout run asynchronously. For details, see the help topic Enabling Ecommerce Features.
- Scriptable Cart – This option lets you designate specific forms to be used for sales orders generated through shopping cart activity. You can customize

these forms, and any scripts they use, for the web store. For details on this feature, see the help topic Scriptable Cart.

## n. Flexibility in Specifying the Related Field Sets

To improve the performance of search result pages in SuiteCommerce Advanced, we have added flexibility in specifying the field set to be loaded when retrieving items for the following synthetic fields:

- Matrix Child Items (Detail)
- Related Items (Detail)
- Correlated Items (Detail)

Previously, the synthetic fields retrieved item fields in the following manner:

- The Matrix Child Items (Detail) synthetic field always retrieved matrix child item fields from the matrixchilditems reserved field set.
- The Related Items (Detail) synthetic field always retrieved item fields from the relateditems reserved field set.
- The Correlated Items (Detail) synthetic field always retrieved item fields from the correlateditems reserved field set.

Since there was no flexibility in specifying the number of item fields in the reserved field set, a large number of item fields were returned for each item. Consequently, the performance of pages that retrieved lot of matrix child items, related items, or correlated items degraded.

In 2017.1, you can use the following newly added Item Search API parameters to override the default field set with any other field set:

- matrixchilditems\_fieldset
- relateditems\_fieldset
- correlateditems\_fieldset

By overriding the default field set with a field set that has fewer fields, the response time in retrieving search result pages can be significantly reduced. For more information, see the help topic Item Search API Input Parameters.

## o. Shopping Cart Performance Enhancement

Prior to release 2017.1, shoppers experienced performance degradation when the number of unique items in their shopping carts increased. The performance degradation was caused due to the overhead associated with retrieving all the item fields from the database for each unique item in the cart.

With 2017.1, the performance of the shopping cart has been enhanced by retrieving all the item fields from the cache, thereby reducing the turnaround time. This performance improvement is evident in larger shopping carts with more than 50 unique items.

## General Performance Best Practices

### a. Image Assets

Image sizes should be < 300KB depending on the image resolution.

Images usually make up the most significant portion of a page's weight on an eCommerce website. Therefore, any gains that can be made here will have significant benefits.

Once images have been resized and requested by users, the resized images will typically be cached on the CDN - provided it has been enabled and configured in your account. Assets like images and CSS inherit the cache life from the SSP application that calls them which for most sites is 7 days.

**Note:** this means your resized images will be cached on the CDN but your original un-resized images will not be cached.

#### **Potential performance problems:**

- Images increase the loading time of the web page due to its the size of the data transfer.

#### **Action Item(s):**

- Ensure that small images and icons are combined into a sprite or Base64 encoded into the site's CSS
- Check that all large images (banners, carousel, etc.) are properly compressed. This is a common mistake.

#### **Further Information:**

- Appendix 1 [Compressing images](#)

### Change background to CSS instead of images

Background images should be replaced with CSS backgrounds such as gradients whenever possible. These are much lighter and mobile friendly than background images.

#### **Potential performance problems:**

- Huge images are used as background and slow down the web site.

#### **Action Item(s):**

- Replace background images with CSS gradients when possible

#### **Further Information:**

- Appendix 1 [Creating background gradients](#)



## Image hosting domain

Images should be hosted at the final domain; we should not see images sourced from system.netsuite.com ever.

### Potential performance problems:

- If images are not hosted in the site's domain, these will not be cached in the CDN.

### Action Item(s):

- Source all images from the site's main using *relative URLs*, as opposed to using absolute URLs and never retrieve images from system.netsuite.com.
- Eliminate any redirects (response codes: 3XX), as redirects slow down your site.

### Further Information:

- <https://developers.suitecommerce.com/optimize-your-images-for-better-performance>

## b. CSS and JS optimized

Specific CSS and JavaScript files are critical resources for SuiteCommerce Advanced applications as the site will not run until they are fully loaded. They should be optimized as much as possible so that they are downloaded quickly and efficiently. Additionally, the techniques highlighted in this section will also make the SEO engine run faster.

### Potential performance problems:

- c. Unused CSS and/or JS functions being included in core files
- d. Duplicate CSS and/or JS
- e. Un-optimized/inefficient JS being executed, extending the loading time.

### Action Item(s):

- Check that all of the application's JavaScript is concatenated (combined) into a single file and minified.
  - This means that a single JavaScript file should be requested on page load (Application.js / shopping.js / checkout.js / myaccount.js).
  - Not more than a single template file should be requested as well.
- Check that all of the application's styles (CSS) are concatenated into a single file and minified.
  - This means that a single CSS file should be loaded (Styles.css / shopping.css / checkout.css / myaccount.css).
- Remove any @import statements on the final compiled CSS if possible.
- As CSS is required to render a webpage, it should be called at the top of the header.
- Remove unused CSS. There are several tools which help accomplish this, including Chrome Dev Tools (Figure 2 below) and Google PageSpeed Insights.

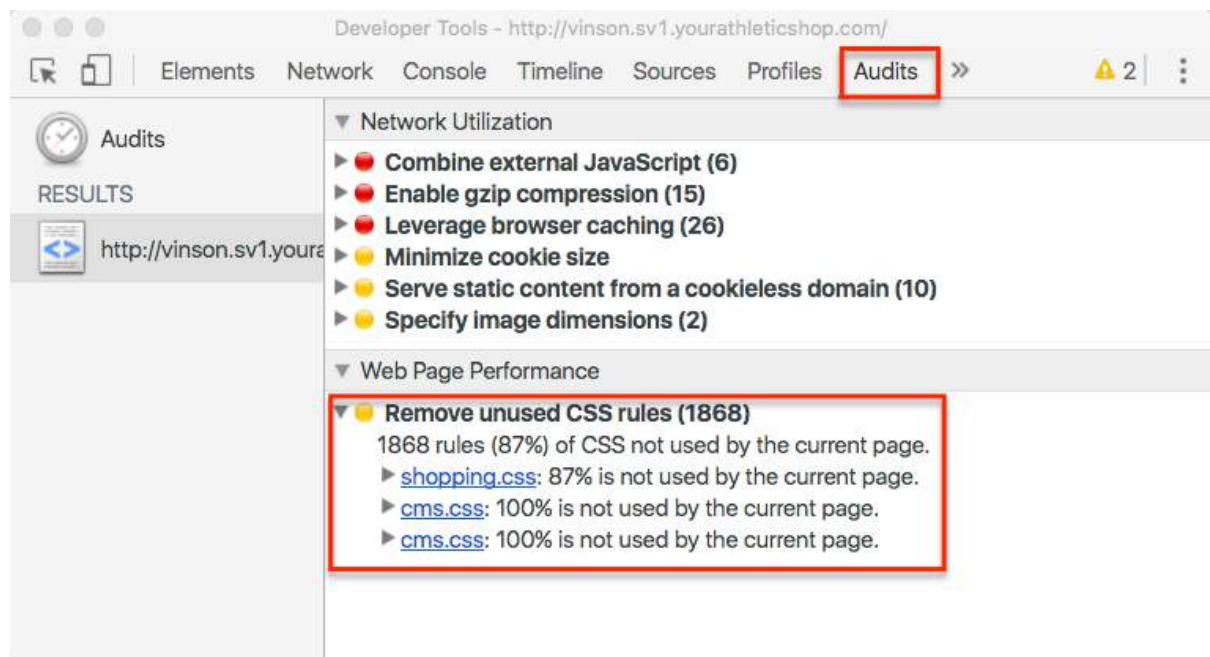


Figure 2. Unused CSS

## c. Third-party applications

Third-party scripts and libraries often have a significant impact on the front-end side of any web application, including SuiteCommerce Advanced. They typically involve requests to third-party domains which NetSuite has no control of, both from a performance and functional point of view.

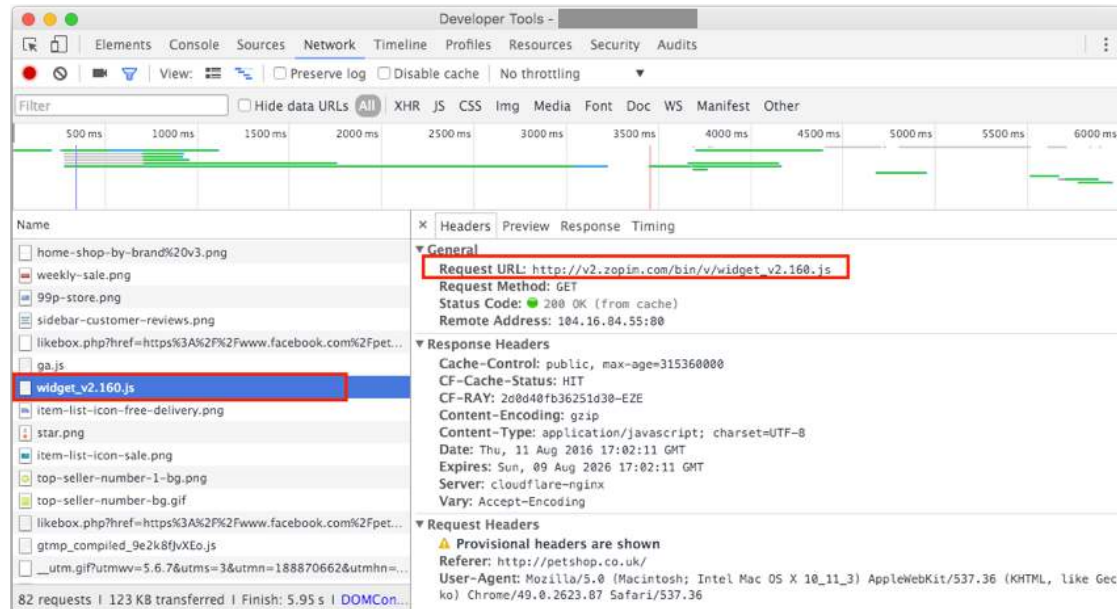
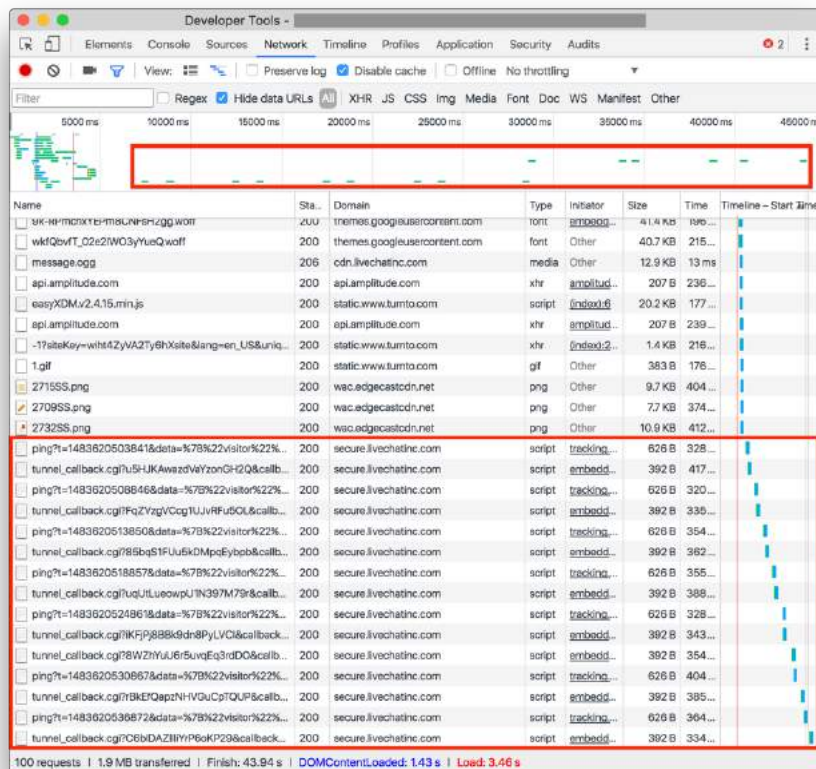


Figure 3. A third-party request

### Potential performance problems:

- Creating multiple requests to external domains slows down page load.
- There is risk for SPOF (single point of failure) issues if third-party scripts are required when rendering the webpage but are slow or fail to respond.
- Third-party scripts are not cached in the content delivery network (CDN), and may not have their own CDN.
- If loaded ahead of SuiteCommerce application code – 3<sup>rd</sup> party content can slowdown the rendering of the page itself.
- Third-party code can make a litany of additional requests which can significantly add to the time it takes to fully load a page.
- Third-party services may load duplicate code, or worse, different versions of common software libraries used, such as jQuery, resulting in unexpected/unpredictable/broken behaviour.

Example:



### Action Item(s):

- Prioritize first-party components over third-party
- Make sure that all third-party scripts are minified
- Move third-party scripts inside NetSuite's file cabinet when possible to avoid SPOF and benefit from CDN caching
- Include these scripts inside the application's main JavaScript file to minimize additional requests (Application.js, shopping.js, etc.)
- Ensure third-party scripts are loaded asynchronously where possible
- Check that you are using the latest version of third-party code
- Exclude third-party scripts from the SEO engine (see section 2) – that aren't required to render critical content for SEO purposes.
- Use a tag manager (such as Google Tag Manager) to manage your third party scripts

### Further Information:

- [Appendix 2: Issues with third-party scripts](#)

## d. Optimize fonts

Fonts can slow down a page considerably if loaded synchronously or sub-optimally.

### Potential performance problems:

- Page is slowed down by fonts loading synchronously.
- Duplicate requests to load a single font.
- Multiple fonts from the same family are downloaded as separate requests.

### Action Item(s):

- Try not to use more than 2 or 3 fonts for a given site.
- Load fonts asynchronously when possible, using a web-safe font to prevent FOIT (flash of invisible text)
- Load all required variants of a given font (italic, bold, etc.) in a single request
- Make sure that font files are being cached on the CDN. They should automatically be cached if they are stored on NetSuite's file cabinet, although other hosting solutions for fonts can be also very fast (like Google Fonts).

## e. Navigation & Relative URLs

Always use relative URLs instead of absolute URLs in SuiteCommerce Advanced when doing internal linking otherwise you will trigger a reloading of the SuiteCommerce environments and SSP applications on subsequent navigation within your website. Core SSPs and environments should only be loaded once for each of: shopping, checkout, and my account.

### Potential performance problems:

- Page reloads when clicking internal links. This is not typical in a single page application, and greatly slows it down.

### Action Item(s):

- Ensure all internal links have relative URLs and not absolute URLs. Hrefs have to be: "/.../" instead of: "http://..."

## f. Languages

### **Potential performance problems:**

- Language files are taking time to load.

### **Action Item(s):**

- If the site is only in English. Remove additional language files



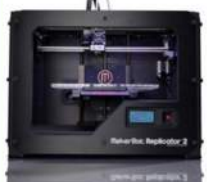



## Appendix 1 – Image Assets

### Image resizing and scaling

Image resizing setup can be found under the Advanced tab on the website setup page.

Detect if images are larger than their containers using Chrome Dev tools. If images which come from NetSuite's resizing service (such as product listing thumbnails) are being resized in the browser, then back end settings should be adjusted as shown below. If other images which do not come from NetSuite's resizing service experience this issue, then the image should be uploaded in the right size on the File Cabinet.

																																											
Flashforge Creator Wood Case 3D Printer	MakerBot Replicator 2 Desktop 3D Printer	MakerBot Replicator 2 Desktop 3D Printer	APC InRow OA Cooling System																																								
<table border="1"> <thead> <tr> <th>Quantity</th> <th>Price</th> </tr> </thead> <tbody> <tr> <td>0 - 9</td> <td>\$1,199.99</td> </tr> <tr> <td>10 - 19</td> <td>\$1,099.99</td> </tr> <tr> <td>20 - 29</td> <td>\$999.99</td> </tr> <tr> <td>30 +</td> <td>\$899.99</td> </tr> </tbody> </table>	Quantity	Price	0 - 9	\$1,199.99	10 - 19	\$1,099.99	20 - 29	\$999.99	30 +	\$899.99	<table border="1"> <thead> <tr> <th>Quantity</th> <th>Price</th> </tr> </thead> <tbody> <tr> <td>0 - 9</td> <td>\$1,299.99</td> </tr> <tr> <td>10 - 19</td> <td>\$1,199.99</td> </tr> <tr> <td>20 - 29</td> <td>\$1,099.99</td> </tr> <tr> <td>30 +</td> <td>\$999.99</td> </tr> </tbody> </table>	Quantity	Price	0 - 9	\$1,299.99	10 - 19	\$1,199.99	20 - 29	\$1,099.99	30 +	\$999.99	<table border="1"> <thead> <tr> <th>Quantity</th> <th>Price</th> </tr> </thead> <tbody> <tr> <td>0 - 9</td> <td>\$1,399.99</td> </tr> <tr> <td>10 - 19</td> <td>\$1,299.99</td> </tr> <tr> <td>20 - 29</td> <td>\$1,099.99</td> </tr> <tr> <td>30 +</td> <td>\$999.99</td> </tr> </tbody> </table>	Quantity	Price	0 - 9	\$1,399.99	10 - 19	\$1,299.99	20 - 29	\$1,099.99	30 +	\$999.99	<table border="1"> <thead> <tr> <th>Quantity</th> <th>Price</th> </tr> </thead> <tbody> <tr> <td>0 - 9</td> <td>\$4,639.99</td> </tr> <tr> <td>10 - 19</td> <td>\$4,539.99</td> </tr> <tr> <td>20 - 29</td> <td>\$4,439.99</td> </tr> <tr> <td>30 +</td> <td>\$4,339.99</td> </tr> </tbody> </table>	Quantity	Price	0 - 9	\$4,639.99	10 - 19	\$4,539.99	20 - 29	\$4,439.99	30 +	\$4,339.99
Quantity	Price																																										
0 - 9	\$1,199.99																																										
10 - 19	\$1,099.99																																										
20 - 29	\$999.99																																										
30 +	\$899.99																																										
Quantity	Price																																										
0 - 9	\$1,299.99																																										
10 - 19	\$1,199.99																																										
20 - 29	\$1,099.99																																										
30 +	\$999.99																																										
Quantity	Price																																										
0 - 9	\$1,399.99																																										
10 - 19	\$1,299.99																																										
20 - 29	\$1,099.99																																										
30 +	\$999.99																																										
Quantity	Price																																										
0 - 9	\$4,639.99																																										
10 - 19	\$4,539.99																																										
20 - 29	\$4,439.99																																										
30 +	\$4,339.99																																										

★ ★ ★ ★ ★ (2)

Image Resizing				
IMAGE SIZE ID *	DESCRIPTION	ENABLED	MAXIMUM WIDTH	MAXIMUM HEIGHT
fullscreen		Yes	1,300	1,300
main	Should be 160px x 160px	Yes	600	600
thumbnail		Yes	225	225
tinythumb		Yes	100	100
zoom		Yes	1,200	1,200

Developer Tools - http://ecscaloydsoffice.suitestore.us/search

Elements Console Network Sources Timeline Profiles Resources Security Audits Backbone

Filter

element.style { }

img { appearance: 445a9259.css:90; height: auto; max-width: 100%; }

img { Styles-0148855a7ed1.css:206; max-width: 100%; width: auto; height: auto; vertical-align: middle; border: 0; -ms-interpolation-mode: bicubic; }

Inherited from a { appearance: 445a9259.css:90; color: #147ba2; text-decoration: none; }

a { Styles-0148855a7ed1.css:206; }

159 x 159 pixels (Natural: 225 x 225 pixels)

## Compressing images

Images can be compressed without reducing their dimensions (height/width). The larger the image, the more you will gain from compressing it. This is why compressing large images such as banners is so important.

There are also 2 types of compression: lossy and lossless. Lossless compression generates a smaller but identical image (pixel perfect) while lossy compression generates even smaller lower-quality images. Lossy compression is generally recommended as it greatly reduces the image's size, while quality loss is generally imperceptible to the human eye.

Use <https://compressor.io/compress> to easily compress images and see a before and after comparison of the image and its size. Highest compression rates can usually be achieved using ImageOptim (<https://imageoptim.com/>). For the best compression of PNGs, ImageAlpha (<https://pngmini.com/>) is the best.



If you are not sure which images need compressing on a given page, test it on <http://www.webpagetest.org/>. The image compression section will score each image on the page and sort them by compression gain, indicating how much weight will be lost by compressing each image.

### Compress Images: 35/100

802.4 KB total in images, target size = 287.4 KB - potential savings = 515.0 KB

FAILED - (221.7 KB, compressed = 42.2 KB - savings of 179.5 KB) - <http://ecscalloydsoffice.suitestore.us/sitelmg/banners/mb-lists-clean.jpg>  
 FAILED - (188.7 KB, compressed = 38.9 KB - savings of 149.8 KB) - <http://ecscalloydsoffice.suitestore.us/sitelmg/banners/mb-laptop-clean.jpg>  
 FAILED - (164.3 KB, compressed = 29.6 KB - savings of 134.7 KB) - <http://ecscalloydsoffice.suitestore.us/sitelmg/banners/mb-printers-clean.jpg>  
 WARNING - (17.2 KB, compressed = 10.4 KB - savings of 6.8 KB) - <http://ecscalloydsoffice.suitestore.us/images/1027141775.media.3d.01.jpg?resizeid=27&resizeh=225&resizew=225>  
 WARNING - (14.0 KB, compressed = 8.3 KB - savings of 5.7 KB) - <http://ecscalloydsoffice.suitestore.us/images/1026668843.media.3d.01.jpg?resizeid=27&resizeh=225&resizew=225>  
 WARNING - (13.9 KB, compressed = 8.6 KB - savings of 5.3 KB) - <http://ecscalloydsoffice.suitestore.us/images/1026674053.media.3d.01.jpg?resizeid=27&resizeh=225&resizew=225>  
 WARNING - (9.8 KB, compressed = 5.6 KB - savings of 4.2 KB) - <http://ecscalloydsoffice.suitestore.us/images/1025250766.media.mac.01.jpg?resizeid=27&resizeh=225&resizew=225>  
 WARNING - (8.6 KB, compressed = 5.1 KB - savings of 3.5 KB) - <http://ecscalloydsoffice.suitestore.us/images/1013848914.media.mac.01.jpg?resizeid=27&resizeh=225&resizew=225>

Large images such as backgrounds or banners/sliders are usually a good target for compression.

## Creating an image sprite

Given the following images found in the footer of an SCA site, we will convert them into an image sprite.



Download all the individual images and upload them into a sprite generation tool such as <http://instantsprite.com>.

### Instant Sprite

CSS Sprite Generator

@bgrins Follow! Articles FAQ Feedback

Select multiple image files from your computer  
or  
Drag and drop files from your desktop onto the page  
or  
Use a few sample images to try it out

#### Thumbnails

You can drag and drop the images to change their order in the sprite.

X	delta_1.png	[ type: image/png   size: 3.00 kb ]
X	maestro_1.png	[ type: image/png   size: 3.05 kb ]
X	mastercard_1.png	[ type: image/png   size: 3.23 kb ]
X	solo_1.png	[ type: image/png   size: 3.70 kb ]
X	visa_1.png	[ type: image/png   size: 2.99 kb ]
X	visa-electron_1.png	[ type: image/png   size: 3.36 kb ]

This will generate the markup and CSS necessary to use said sprite. Base64 encoding is also generated in case it is preferred. CSS class prefixes can be also configured.

#### Sprite

Right click the image to save it to your computer. [Open image in a new window](#) [Copy Base64](#)



#### Usage

You can use a [RegExp](#) object for capturing the file name.

CSS prefix sprite .class prefix (.\*) class suffix { ... }

```
.sprite { background: url('sprite.png') no-repeat top left; width: 44px; height: 39px; }
.sprite.delta_1 { background-position: 0 0; }
.sprite.maestro_1 { background-position: -49px 0; }
.sprite.mastercard_1 { background-position: -98px 0; }
.sprite.solo_1 { background-position: -147px 0; }
.sprite.visa_1 { background-position: -196px 0; }
```

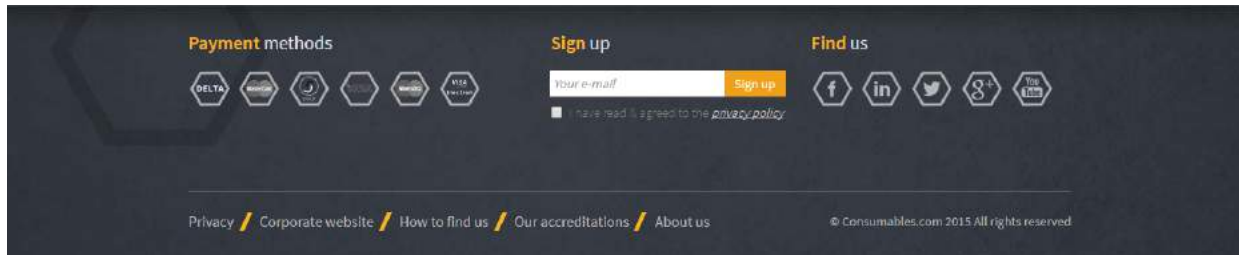
```
<div class='sprite delta_1'></div>
<div class='sprite maestro_1'></div>
<div class='sprite mastercard_1'></div>
<div class='sprite solo_1'></div>
<div class='sprite visa_1'></div>
<div class='sprite visa-electron_1'></div>
```

## Creating background gradients

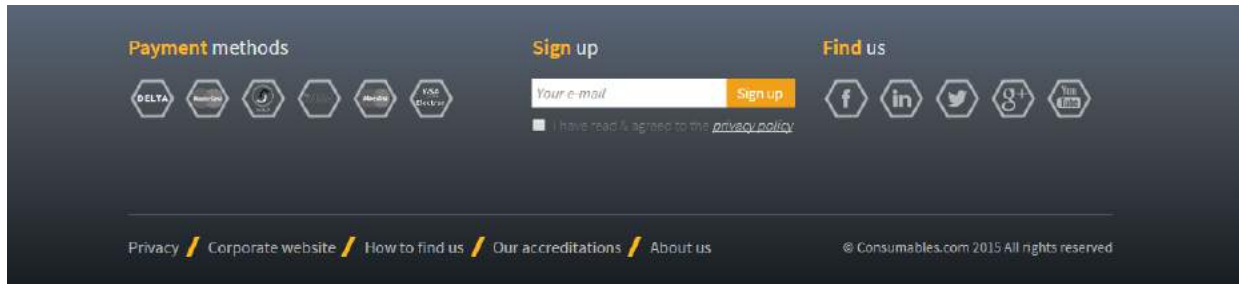
Background images should be replaced with CSS backgrounds such as gradients whenever possible. These are much lighter and mobile friendly than background images.

CSS gradients can easily be generated with online tools such as <http://www.colorzilla.com/gradient-editor/>

Before



After



CSS

```
#Footer {
  /* Old browsers */
  background: #596679;
  /* FF3.6+ */
  background: -moz-linear-gradient(top, #596679 0%, #474d53 36%, #171d21 100%);
  /* Chrome, Safari14+ */
  background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#596679), color-stop(36%,#474d53), color-stop(100%,#171d21));
  /* Chrome10+, Safari5.1+ */
  background: -webkit-linear-gradient(top, #596679 0%,#474d53 36%,#171d21 100%);
  /* Opera 11.10+ */
  background: -o-linear-gradient(top, #596679 0%,#474d53 36%,#171d21 100%);
  /* IE10+ */
  background: -ms-linear-gradient(top, #596679 0%,#474d53 36%,#171d21 100%);
  /* W3C */
  background: linear-gradient(to bottom, #596679 0%,#474d53 36%,#171d21 100%);
  /* IE6-9 */
  filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#596679', endColorstr='#171d21',GradientType=0 );
}
```

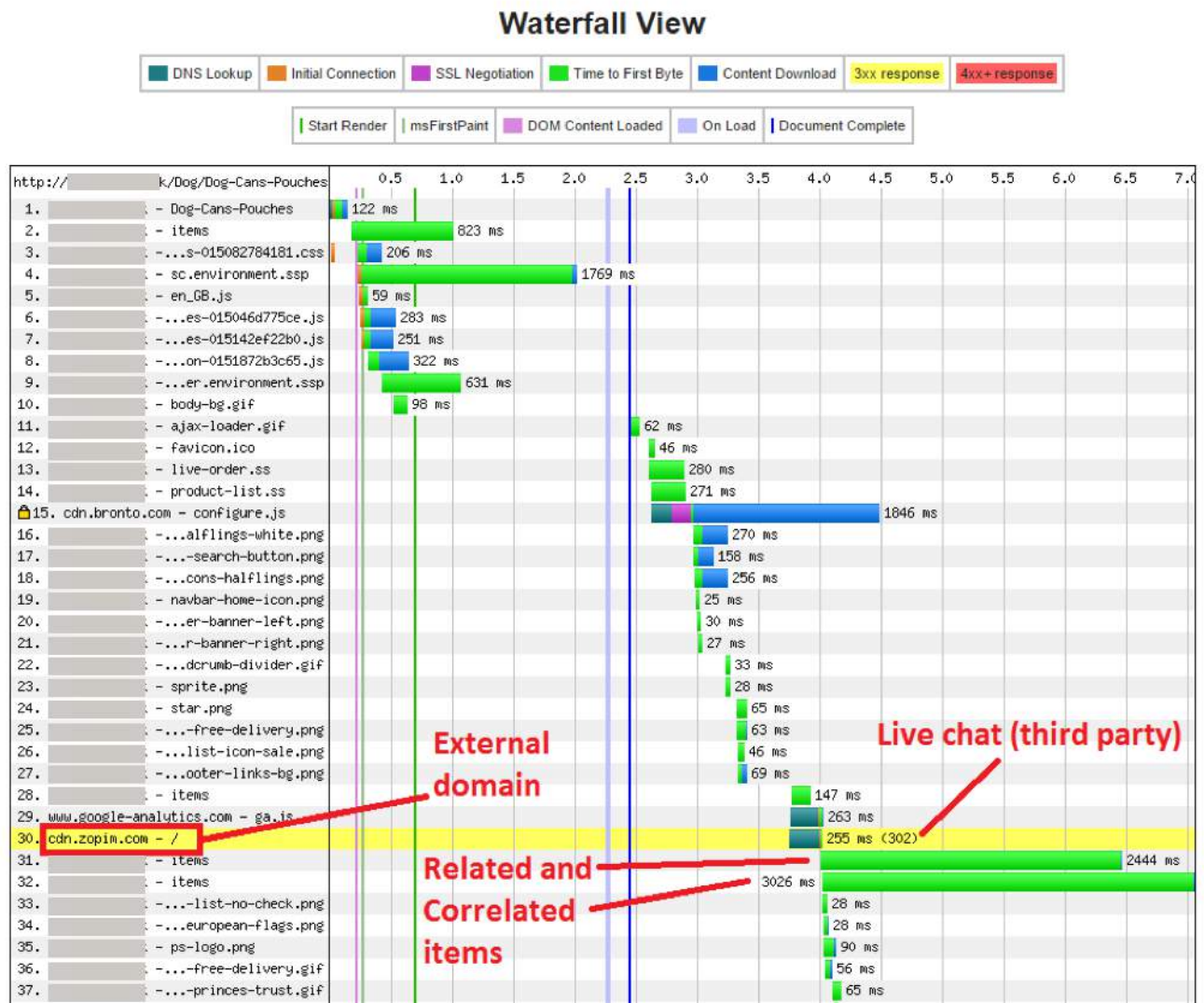


## Appendix 2: Issues with third-party scripts

### Checking that third-party scripts are loaded after first-party ones

Use the waterfall view on WebpageTest and check that third-party requests are on the bottom. They can easily be identified because their domain does not correspond to the site or NetSuite.

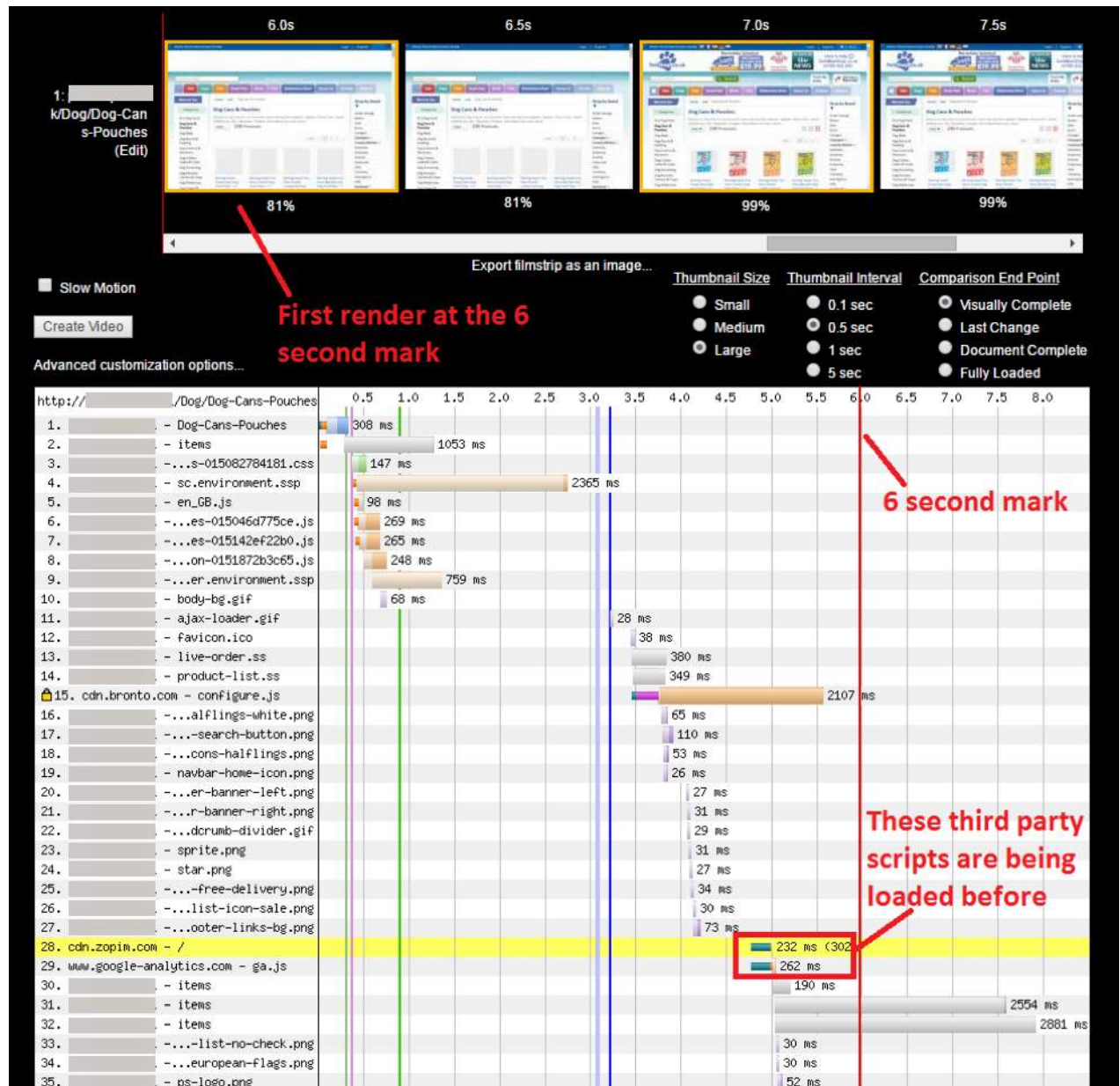
In the example below, both Google Analytics and a live chat widget are requested before Item API calls. This should be reverted



## Checking that third-party requests are sent after the page has been loaded

You can use WebPageTest's filmstrip view for this purpose. Scroll until you find the frame in which the page renders for the first time, and make sure that all third-party requests are done after that point in time.

In the example below you will find that both Google Analytics and the chat widget requests are being sent before the page has been rendered, and those requests should therefore be delayed.





## Emulating SPOF scenarios

We can easily emulate scenarios in which one of the third-party resources fails. In these cases, the site should load and render as expected. Otherwise we depend entirely on the availability of said resources.

Use the SPOF menu on webpagetest and add third-party domains there to see a comparison of how the site loads normally and when the third-party resource fails.

## Appendix 3 - Categories

Categories should not be included in the shopping.user.environment.ssp



Categories should be included in shopping.environment.ssp



## Appendix 4 – Items API

Item API requests can include a “facet.exclude” parameter which accepts a set of field IDs. When this parameter is present, all given facets are excluded from the item API response, reducing the time it takes for the request to resolve and thus improving performance.

### Example

In the following image we compare the output of the following 2 requests:

- Including aggregated pricelevel information  
Time: **1.5 seconds**  
<http://staging.yourathleticshop.com/api/items?fieldset=search&include=facets&n=7&pricelevel=5>
- Excluding aggregated pricelevel information (added facet.exclude parameter)  
Time: **1.1 second**  
<http://staging.yourathleticshop.com/api/items?fieldset=search&include=facets&n=7&pricelevel=5&facet.exclude=pricelevel5>

Left: including aggregated pricelevel information	Right: excluding aggregated pricelevel information
<pre> total: 217, + items: [...], - facets: [   - {     id: "custitem24",     + values: [...]   },   - {     id: "custitem28",     + values: [...]   },   - {     id: "custitem29",     + values: [...]   },   - {     id: "custitem30",     + values: [...]   },   - {     id: "custitem31",     + values: [...]   },   - {     id: "custitem33",     + values: [...]   },   - {     id: "custitem39",     + values: [...]   },   - {     id: "custitem40",     + values: [...]   },   - {     id: "custitem_production",     + values: [...]   },   - {     id: "isonline",     + values: [...]   },   - {     id: "category",     + values: [...]   } ], </pre>	<pre> total: 217, + items: [...], - facets: [   - {     id: "custitem24",     + values: [...]   },   - {     id: "custitem28",     + values: [...]   },   - {     id: "custitem29",     + values: [...]   },   - {     id: "custitem30",     + values: [...]   },   - {     id: "custitem31",     + values: [...]   },   - {     id: "custitem33",     + values: [...]   },   - {     id: "custitem39",     + values: [...]   },   - {     id: "custitem40",     + values: [...]   },   - {     id: "custitem_production",     + values: [...]   },   - {     id: "onlinecustomerprice",     + values: [...],     min: 0,     max: 179.99,     ranges: [ ]   },   - {     id: "isonline",     + values: [...]   },   - {     id: "category",     + values: [...]   } ], </pre>

55 price values

Left: including aggregated pricelevel information  
Right: excluding aggregated pricelevel information

## Excluding facet information through coding

We can exclude facet information through the following code snippet:

```

_.extend(SC.Application("Shopping").Configuration.searchApiMasterOptions.Facets, {
  "facet.exclude": "pricelevel5" // Comma-separated field IDs
});

```

Note that if we exclude a facet that is used on product listing results, we will get the same results but the corresponding facet option on the menu will not be rendered (depicted below).

The image shows two identical product filter panels side-by-side. Each panel has a 'Size' section with checkboxes for X-Small, Small, Medium, Large, Large/XL, X-Large, XX-Large, and ONE SIZE. Below this is a 'Color' section with a yellow color swatch. The left panel also includes a 'Price' section with a range slider from \$10.00 to \$174.99. The right panel does not have the 'Price' section.

Left: including aggregated pricelevel information

Right: excluding aggregated pricelevel information

### Excluding facets to improve performance

Make sure to apply the aforementioned method to exclude all facets that are present in the website's facet menu but comply to at least one of the following:

- Not included in the site's configuration (Configuration.js)
- Not rendered on the facet menu
- Contain a range of values that is too large and therefore impacts performance

This is extremely important since the facet information returned in item API responses is very large as it summarizes not only the facet information for items shown in the current page, but to all items in the current facet/category selection. In other words, if you are currently showing 24 items in the first page of a product listing page that contains 10 pages in total, the item API response contains facet information for all items in the whole 10 pages (up to 240 items).

```

242 <!-- 236 s: 14% #22 co: 20% #28-->
243 <!-- [ sh.j10031.bos ] [ 2015.2.0.31 ]-->
244 <!-- [ 3685575 ] [ /.nl ] [ Wed Sep 02 07:41:36 PDT 2015 ] -->
245 <!-- Not logging slowest SQL -->
246
247 <!-- Parameter from url: seodebug=T:
248 [07:41:36.283] [ 0 ms ] Requested URL with SEO generator relevant params: http://sustainablesupply.cdn.nal.netsuitesting.com/?seodebug=T
249 [07:41:36.293] [ 0 ms ] Source URL: http://sustainablesupply.cdn.nal.netsuitesting.com/shopflow/index.ssp?seodebug=T
250 [07:41:36.293] [ 0 ms ] Rewrite Path: /.nl?sitepath=/shopflow/index.ssp
251 [07:41:44.016] [ 7723 ms ] [07:40:42.333] [ 1441204842333 ms ]
252 Log mode set to: INIT
253 [07:40:42.376] [ 19 ms ] require id: platform/jav8
254 [07:40:42.394] [ 17 ms ] require id: platform/core

```

[07:41:40.164]	[ 2 ms]	Envjs.XMLHttpRequest	Java8	DEBUG:	[Wed, 02 Sep 2015 14:41:40 GMT:163]	Envjs.connection open: http://susta
[07:41:41.178]	[ 1014 ms]	Envjs.XMLHttpRequest	Java8	DEBUG:	[Wed, 02 Sep 2015 14:41:41 GMT:177]	contentEncoding utf-8
[07:41:41.203]	[ 25 ms]	Envjs.XMLHttpRequest	Java8	DEBUG:	[Wed, 02 Sep 2015 14:41:41 GMT:203]	Envjs.connection: finished
[07:41:41.204]	[ 1 ms]	Envjs.XMLHttpRequest	Java8	DEBUG:	[Wed, 02 Sep 2015 14:41:41 GMT:204]	calling ajax response handler
[07:41:41.205]	[ 1 ms]	Envjs.XMLHttpRequest	Core	DEBUG:	[Wed, 02 Sep 2015 14:41:41 GMT:205]	unregistering connection. http://su
[07:41:41.205]	[ 0 ms]	Envjs.XMLHttpRequest	Core	DEBUG:	[Wed, 02 Sep 2015 14:41:41 GMT:205]	connections 0

```
335 [18:43:31.131] [ 36 ms] Envjs.Jav8 DEBUG: [Mon, 11 Jan 2016 02:43:31 GMT:131] init start
336 [18:43:31.132] [ 1 ms] Envjs.Jav8 DEBUG: [Mon, 11 Jan 2016 02:43:31 GMT:132] compId and siteId: 875702, 5
337 [18:43:31.132] [ 0 ms] Envjs.Jav8 DEBUG: [Mon, 11 Jan 2016 02:43:31 GMT:132] dontCacheScripts:false
338 [18:43:31.132] [ 0 ms] Envjs.Jav8 DEBUG: [Mon, 11 Jan 2016 02:43:31 GMT:132] dontCacheSsSsp:true
339 [18:43:31.132] [ 0 ms] Envjs.Jav8 DEBUG: [Mon, 11 Jan 2016 02:43:31 GMT:132] evaluatePage: Max execution limit: 30000
```

```

256 <!-- Parameter from url: seodebug=T:
257 [18:56:49.612] [ 0 ms] Requested URL with SEO generator relevant params:
http://www.sustainableupply.com/?seodebug=T
258 [18:56:49.613] [ 0 ms] Source URL: http://www.sustainableupply.com/shopflow/index.ssp?seodebug=T
259 [18:56:49.613] [ 0 ms] Rewrite Path: /s.nl?sitepath=/shopflow/index.ssp
260 [18:56:55.431] [ 5818 ms] Summary: Error occurred for SEO page generation! sUrl
http://www.sustainableupply.com/shopflow/index.ssp?seodebug=T_Error:
-----ERROR-----
261 TypeError: Cannot read property 'target' of undefined, at <anonymous>:8:43376, at Array.forEach
(native), at Function.N.each.N.forEach (<anonymous>:15:25440), at Object._.extend.enhancePage
(<anonymous>:8:43332), at Object.<anonymous> (<anonymous>:8:204487), at Object.enhancePage
(<anonymous>:15:33608), at <anonymous>:8:44867, at d.Model.fetch.success (<anonymous>:8:41109),
a
Object.O.success (<anonymous>:49:4967), at b4 (<anonymous>:40:21105)
262 ERROR
263 TypeError: Cannot read property 'target' of undefined, at <anonymous>:8:43376, at Array.forEach
(native), at Function.N.each.N.forEach (<anonymous>:15:25440), at Object._.extend.enhancePage
(<anonymous>:8:43332), at Object.<anonymous> (<anonymous>:8:204487), at Object.enhancePage
(<anonymous>:15:33608), at <anonymous>:8:44867, at Object.a [as loadPage] (<anonymous>:8:40822),
at null.<anonymous> (<anonymous>:8:44778), at null.updateUI (<anonymous>:15:33608)

```

[illegible]

## Removing functionality from the SEO page generator

You can wrap functionality under the following snippet, which will prevent it from executing on the SEO page generator:

```
If ( !SC.isPageGenerator() ){  
  //Insert non-page generator content here  
}
```

This is a great tool for performance optimizations as it allows us to remove unnecessary processing from the SEO page generator, such as user-specific functionality.

For example, product lists (e.g. wish list) are clearly not needed for search engines as they will never have any products stored there. The product list module can easily be deactivated for the SEO page generator using the aforementioned snippet.

Another creative use is to disable third-party content such as live chats.

## Appendix 6 - SuiteScript

### Using Content Delivery Network (CDN) cache in SuiteScript

Add the following code to Services, SSP applications or Suitelets in order to cache their responses in the CDN.

```
...  
...  
  
response.setCDNCacheable(response.CACHE_DURATION_UNIQUE); //not cached  
response.setCDNCacheable(response.CACHE_DURATION_SHORT); //5 min  
response.setCDNCacheable(response.CACHE_DURATION_MEDIUM); //2 hours  
response.setCDNCacheable(response.CACHE_DURATION_LONG); //7 days  
response.setContentType('JAVASCRIPT');  
  
...  
...
```

Remember

- Do not set cache headers on user-specific context



## Using Application Cache for SuiteScript

The following code snippet highlights how to cache data in NetSuite's application server. Up to 100KB of text can be cached in a key-value store. This approach works on secure domains, unlike CDN. Remember that this cache is not per-session, so do not store user sensitive data here.

```
var ttl = 5 * 60; //time to live, in seconds;
var cacheName = 'something'; //name of the cache
var key = '_1'; //key
var value = 'Hello'; //value to store

var cache = nlapiGetCache(cacheName);

//Storing
putReturn = cache.put(key, JSON.stringify(value), ttl); //cache stores strings. Returns a status object

//Checking that store worked
var successfullyCached = putReturn != null && putReturn.status == 'SUCCESS';

// retrieving values from the cache
var getValue = cache.get(key); //null if it failed
```



## Using Cache Headers on non-session related services

Cache headers can be specified on the Application's `sendContent` method. This will enable HTTP cache.

```
function service (request)
{
    'use strict';
    // Application is defined in ssp library commons.js
    try
    {
        var method = request.getMethod()
        ,   id = request.getParameter('internalid')
        ,   data = JSON.parse(request.getBody() || '{}')
        // ProductReview model is defined on ssp library Models.js
        ,   ProductReview = Application.getModel('ProductReview');

        switch (method)
        {
            case 'GET':
                var result;

                if (id)
                {
                    // we get the review
                    result = ProductReview.get(id);
                    // if the review is not approved
                    if (result.status !== ProductReview.approvedStatus || result.isinactive)
                    {
                        throw notFoundError;
                    }
                }
                else
                {
                    var params = request.getAllParameters()
                    ,   filters = {}
                    ,   param = '';

                    for (param in params)
                    {
                        filters[param] = params[param];
                    }

                    result = ProductReview.search(filters, filters.order, filters.page, filters.limit);
                }
                // send either the individual review, or the search result
                Application.sendContent(result, {'cache': response.CACHE_DURATION_MEDIUM});
                break;
            }
        }
    }
}
```

## Remember

- Do not set cache headers on user-specific context
- There is no caching on the secure server yet