



Ad Load Performance Best Practices

Released October 2008

These Best Practices have been developed by the IAB Ad Load Performance Working Group with guidance from the IAB Ad Operations Council.

About the IAB Ad Load Performance Working Group:

The IAB Ad Load Performance Working Group consist of key IAB member companies working together in order to create best practices aimed at decreasing the load time of advertisements within Web pages.

Key Contributors:

AOL
Google
Microsoft
Time Inc.
Turner
Univision
Yahoo!, Inc.

About the IAB Ad Operations Council:

The Ad Ops Council is dedicated to improving the operational efficiency of interactive advertising. Ad Ops Council working groups regularly include agency-side representatives to help improve communication, understanding, and work process in many areas of the buyer-seller relationship.

A full list of Council member companies can be found at:

http://www.iab.net/member_center/35088?iabid=a0350000002Cmy1AAC

This document can be found on the IAB website at: www.iab.net/adload

IAB Contact Information:

Jermaine Griffin
Manager of Industry Services, IAB
212-380-4704
Jermaine@iab.net

Jeremy Fain
Vice President of Industry Services, IAB
212-380-4724
Jeremy@iab.net

Table of Contents

Introduction	3
Best Practices	4
1. Make Things Smaller	4
1.1. Reduce Requests.....	4
1.2. Reduce Domains.....	4
1.3. Reduce Size	5
2. Get Content Closer to the User.....	6
2.1. Use a Content Delivery Network (CDN)	6
2.2. Cache Content.....	7
2.3. Re-use Connections	8
3. Advanced Techniques	8
3.1. Make Ad Request Asynchronously	8
3.2. Polite Download.....	9
Appendix A - Testing	10
Appendix B – Tools and Resources	12
Checklists	14
Agency/Creative Shop Checklist.....	14
Ad Vendor Checklist.....	14

Introduction

This document explains several best practices that can be employed by creative agencies, ad-serving vendors, and publishers to reduce the load time for ads. Ads that load faster can help web pages load faster, leading to higher user satisfaction, lower page abandonment, and, most importantly for agencies and marketers, more opportunity for ads to yield higher conversion or click through rates. These best practices can typically be implemented with very little cost and can be a win-win-win situation for all parties involved in the development and serving of interactive advertising.

Reducing the load time for an ad consists of many of the same steps for reducing the load time for a Web page. In order to get the ad's components to the user faster, two basic categories of best practices emerge:

- **Make things smaller** – Both making the size (in bytes) of an ad smaller, as well as reducing the number of its components.
- **Get content closer to the user** – Hosting content physically close to the user, as well as ensuring that unchanged content for subsequent visits does not need to be retrieved again.

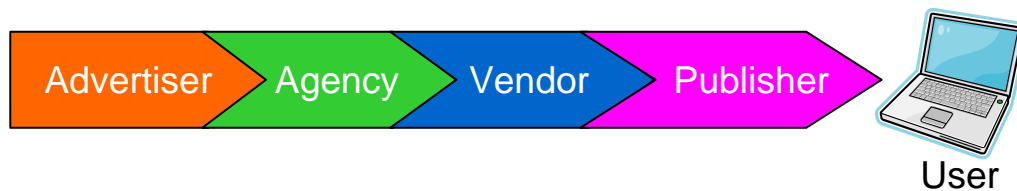
The best practices that follow were born from industry work on Web page improvement but have been specifically tailored for online ads. In addition, an **Advanced Techniques** section discusses more sophisticated methods of least impact ad delivery.

The **Testing** and **Tools** appendices of this document outline techniques, best practices, and available tools for functional and performance testing of ads.

Finally, the **Checklist** section of this document provides a quick reference that can be incorporated into the QA process for agencies, ad servers, and publishers.

Ad Creation and Delivery Ecosystem

In the ad creation and delivery ecosystem, there are typically several key parties involved from start to finish:



This document focuses on the portions of the ecosystem involving the Creative Agency, Ad-Serving Vendor, and Publisher. Some of the best practices below are only applicable to one party (e.g. image compression would be done by the Creative Agency), while others are equally applicable to all. To help understand how to best leverage each best practice, labels are used to indicate applicability within the ecosystem.

Best Practices

1. Make Things Smaller

A display ad is rarely comprised of a single component. One ad will often have multiple components which include images, JavaScript files, Flash, counting instrumentation, etc. The number and size of external components affects load time from the end user perspective. Reducing both the number of components, and their size, are some of the most effective techniques to improve load time.

1.1. Reduce Requests

Agency

Aside from the obvious directive to only include necessary files for an ad, there are methods that can be employed to reduce the number of components without compromising functionality. Multiple JavaScript files can be combined into a single file. There may be valid development or business reasons to maintain distinct files, but consider combining the files into one at delivery time. The same reduction holds true for CSS files. This concatenation of files can simplify the minification process (see [Section 1.3 Reduce Size](#)).

HTTP *redirect* is a common technique with ads. A redirect does come at the cost of an additional HTTP request. In cases where the redirect is used for counting impressions, it may be worth investigating if counting via *beaconing* is a viable alternative. Although a beacon will generate an HTTP request as well, there may be situations where information for multiple ads can be consolidated into one beacon or the beacon can be requested in a manner that does not delay the user experience, i.e., after the content of the page has loaded. Redirects, for purposes other than counting, should be avoided.

The guidance to reduce requests is distinct from what is sometimes labeled as a *polite download*. Polite techniques entail showing part of the ad before page content, and then loading the remainder after the content is fully loaded. For more on polite download see [Section 3 Advanced Techniques](#).

Additional, advanced techniques to reduce HTTP requests for images, such as image maps and CSS sprites, exist for general Web pages, but may only be applicable in niche cases for ads.

1.2. Reduce Domains

VendorPublisher

Another contributor of load time is how many distinct locations, or *domains* (e.g. ad.doubleclick.net), ad components are coming from.

Overall Web page load time can be reduced if components can be retrieved in parallel (i.e. several at a time) vs. serially (i.e. one at a time). However, too much of a good thing - parallel component loading - can cause capacity problems, so Web browsers limit the number of components that can be loaded in parallel from any given domain. To combat these limits, the higher levels of parallel retrieval can be achieved by delivering components from multiple domains.

However, delivering ad components from multiple domains can cause performance problems too. For each domain used, the Web browser has to 'look up' the address of the domain via DNS, or the Domain Name Service. These *DNS lookups* can take a significant (and often highly variable) amount of time.

On one hand, ad performance can be improved by the additional parallel component retrieval obtained when using multiple domains; but on the other hand, performance can be diminished by the time required for the DNS lookups required by using multiple domains. A general rule of thumb is to use at least 2, but not more than 4 domains.

Often a domain structure is chosen around the delivery of different types of content. For example, one domain is used for static components (i.e. things that rarely change, like images or support files), and another is used for dynamic content (e.g. targeting logic). This functional division also facilitates moving static content to a Content Delivery Network or CDN (see [Section 2.1 Use a Content Delivery Network](#)), and can also be used to consolidate cookie traffic (see **Cookies** in [Section 1.3 Reduce Size](#)).

1.3. Reduce Size



Reducing the size of each ad component means it can be transferred to the user more quickly. The Best Practices listed below help reduce and minimize the size of the ad unit.

All Ad Units

Compression Tools

Compressing text/script/CSS components with GZIP is a very effective way of reducing size, which in turn reduces load times by reducing the size of the HTTP response. Using GZIP generally reduces the load times significantly.

Minification

Minification is the process of removing all unnecessary characters from the ad code without altering the functionality. By removing the unneeded characters, the overall size of the ad code is reduced and load times are improved. Minifying ad code includes: removing characters that may include white spaces, new line characters, tab characters and in some cases block delimiters. There are tools available to minify code such as JSMIn, as well as several publisher proprietary systems. In addition, shortening file names such as long variable names helps in reducing size, but can generally be done during the publication stage.

Ad Images

Several areas that are often overlooked are the correct format, size, and compression of ad images.

Different image formats (i.e. GIF, JPEG, PNG) are optimal for different kinds of images. For example, the GIF format is ideal for images with a small number of uniform colors. The JPEG and PNG formats are ideal for photos, or images with a large numbers of varied colors.

It is also important to properly size the image (this applies to video ads as well). A common shortcut is to use one large image, and adjust rendering size in the browser via HTML sizing controls. The large image, however, is still downloaded to the user and can significantly reduce load time. Instead, create versions of the image for each rendering size.

In addition to choosing the optimal format and size, compressing images or reducing image quality (but not below acceptable levels) can further decrease file size. See the **Image Optimization** section of [Appendix B – Tools and Resources](#) for suggested tools.

Cookies

Cookies can be a hidden contributor to size. Cookies are bits of text information, and are associated with a particular domain name. Once a cookie is *set* or applied to a domain, all requests to that domain will contain the cookie. The extra bytes added by these cookies can impact performance. Therefore, cookies should only be used where absolutely necessary, and should be consolidated to a single domain. It is also important to set appropriate cookie expiration dates.

It is also advisable to use domain segregation to keep static content separate from dynamic content, and keep the static content free of cookies. Hosting static components on a static domain or using an entirely new domain (like [www.sitenamemg.com](#)) will keep the domain cookie free.

External JavaScript and CSS

It may seem counter-intuitive, but putting commonly used *JavaScript* and *CSS* bits in external files (1 each), instead of including them repeatedly in other files, allows them to be cached and re-used. External files of JavaScript and CSS that are in the HTML are downloaded every time the HTML doc is called, thus reducing the amount of HTTP calls. The main goal is to minimize the amount of HTTP requests without drastically increasing the size of the HTML document.

Video Ad Units

Reducing the Resolution

Reducing the resolution will not cause a loss in effective quality. Resolutions that are too high cannot be displayed on a normal PC or TV. As a best practice, reduce both the height and width by a factor of 2, which means that the total amount of data needed to store the video is reduced by a factor of 4. This reduces the file size significantly.

Reducing the Frame Rate

The frame rate has a significant impact on the file size. Lowering the frame rate of the video to 15 or 10 fps can reduce the file size by as much as 200% or 300%. However, reducing the frame rate may affect the smoothness of the playback. Acceptable frame rates vary depending on the specific video clip. Experiment to find the lowest acceptable value and be careful to check the minimum and maximum fps specifications for the publishers and vendors the ad will be served through.

Increasing the Key Frame Rate

In most video files, the entire image in the first frame is stored in the file. Later frames are not saved as complete images—only the differences between the frames are saved. The key frame rate defines how often full frames ("key frames") are saved in those files. A high value of key frame rate results in smaller file size because fewer full frames are stored. However, the higher the key frame rate, the harder it is to seek within a file.

Changing Audio Settings

The audio portion of an AVI file can be compressed. As with video compression, audio compression will dramatically reduce file size, by sacrificing some quality. For most codecs, another option is to select the sample frequency. This is the number of times per second (ranging from 8,000 to 48,000 times per second) that the encoder measures sound levels.

Flash

If Flash is used, there are several areas where optimization can be applied. Ensure backgrounds and layers are sized appropriately (i.e. avoid large background images, only a portion of which is used in the ad). Remove any hidden layers. Avoid using advanced, CPU intensive effects (e.g. fire, smoke, water/reflection). If ActionScript is used, avoid complex calculations.

2. Get Content Closer to the User

Geographic proximity between ad content and end users helps to reduce load time – the further away the ad servers, the longer the time it takes to transmit the data over the Internet. In addition, caching ad content and components close to the end user further reduces the effects of this geographic latency.

2.1. Use a Content Delivery Network (CDN)



Publisher

A *Content Delivery Network* or *CDN* is a system of geographically distributed servers that replicate (cache) origin server content at locations physically close to end users. A CDN is an efficient means to distribute the content of an ad by being optimally located relative to end users. CDN service providers can often support more exhaustive coverage, better load balancing, and scalability, than most in-house CDN solutions.

It is important to note that every static file associated with an ad can be hosted on a CDN. Although CDN hosting is commonly associated with images, script files can also be hosted by a CDN. If a file hosted by a CDN needs to be updated, a manual cache flush can be initiated, or the filename can be revised to avoid caching conflicts with previous versions. If multiple parties are editing the same file hosted on a CDN, it may be worth devising a revision management technique rather than opting not to use a CDN.

Do not assume that the CDN abides by all the recommendations in this document. Work with the CDN service provider to optimize the settings which give the best service to customers and best experience to end users. As one example, it is advisable to check the HTTP response headers to ensure they are set optimally. In particular, make note of the Cache-Control setting (see [2.2 Cache Content](#)). The same checks can be done for many of the other recommendations in this document which relate to Vendors. The Tools section ([Appendix B](#)) lists a few applications which will allow a user to view the HTTP header information of content.

2.2. Cache Content



Caching ad content and components closer to the end user improves ad load performance by reducing the effects of the relative geographic latency between the end user and the server hosting the ad content.

All ad content or components, especially those that are static or change infrequently, should be configured with the appropriate cache policy and expiration settings such that it can be stored (or not, in the case of truly unique requests) in the various data caches¹ that are available: on proxy servers, in geo-distributed CDNs, or locally on the end user's computer in the browser cache.

With the appropriate expiration and cache control settings, if a piece of ad content is needed again in the future, the information will already be available closer to the end user, greatly reducing or even eliminating network retrieval time, resulting in significantly faster response times. More specifically:

Expiration Dates

Static files such as JS, CSS, images, and XML can be cached by the browser for when the user loads the page again or for subsequent pages that utilize the same content files. The browser will look for an expiration date in the file's HTTP header to see if the file is still valid. However, even if a file is unchanged, if the expiration is not set properly, the browser will still make an unnecessary call to the server to determine if the file has been modified. The recommended practice is to set the *Expires* header associated with each static file to a date that is years into the future. For content that changes more often, setting the Expires header days or even hours ahead can still provide performance benefits².

Note: *If the content or component changes prior to the expiration date, the file name will need to be changed in order to force the browser and any intermediate caches to load the file and repopulate their cached copies.*

Cache-Control

Even dynamic content may be cacheable, and it can be controlled by using the appropriate *Cache-Control* header value. Commonly used settings include:

- **Public:** The HTTP response may be stored by any cache. For example, either the browser or a proxy server can cache the response. This allows sharing of content across different users who are using the same proxy server or CDN.
- **Private:** The response message is intended for a single client and must not be cached by any shared cache.
- **No-cache:** No cache in the entire request path should maintain a copy of the response

Note: *When using both Expires and Cache-Control settings, the Cache-Control setting will take precedence.*

¹ For more information on web caches, see <http://www.procata.com/cachetest>

² For more information on expiration settings, see <http://developer.yahoo.com/performance/rules.html#expires>

ETags

Related to cache settings, *Entity Tags (ETags)* tend to be a source of problems for otherwise cacheable content. ETags are intended to uniquely identify files using a numeric tag generated by the server; however, in clusters of servers, each server will often create a different number, which can then invalidate any caching. Also, the rules for honoring ETags are different for origin servers and proxies, which can also lead to missed opportunities for caching content. While ETags provide a flexible model for validating whether content has been modified at the source, it is usually best to avoid using ETags unless fully aware of the technical implications and precedence rules³.

Support for Expires, Cache-Control, and ETags is documented as part of the HTTP/1.1 RFC⁴: Please see the technical documentation for the Web server being used to determine how to set expiration dates and cache control options.

2.3. Re-use Connections


 A diagram consisting of two overlapping arrow-shaped boxes. The left box is blue and labeled 'Vendor', pointing right. The right box is pink and labeled 'Publisher', pointing left. They overlap in the center.

When an ad component is retrieved from a host server, a network connection must be established to that server. If other components or files will be retrieved from the same server (or, more accurately, the same Internet domain name), reusing the existing connection, instead of closing and re-opening it, can significantly improve performance. This is accomplished by enabling support for HTTP *Keep-Alives* on the server (or possibly a network load balancing device). It is also important for proxies and CDNs to enable support for Keep Alives in order to provide for fast downloading of content.

When Keep-Alives are enabled, a TCP network connection is opened and then used for several files, with only the one round-trip used to open the initial connection. Maintaining connections over the lifetime of a user's session also allows subsequent files to be loading over the network with an expanded TCP window, which provides for optimal data transmission over the network, especially for larger files.

Keep-Alive support is documented as part of the HTTP/1.1 RFC⁴. Please see the technical documentation for the Web server being used to determine how to enable HTTP Keep Alives.

3. Advanced Techniques

Other, advanced techniques can be employed to further reduce the impact of ad load time on overall page load time.

3.1. Make Ad Request Asynchronously


 A diagram consisting of two overlapping arrow-shaped boxes. The left box is blue and labeled 'Vendor', pointing right. The right box is pink and labeled 'Publisher', pointing left. They overlap in the center.

To improve overall download time for Web pages and to make the most efficiency use of available bandwidth, requests for ads and ad components should be made in parallel ("asynchronously") to other page content.

In an effort to allow the asynchronous loading of Web content and ad content, a number of publishers and rich media vendors have collaborated on a best practice for loading not only standard ad products (gif/jpg/flash) but also rich media ads. This effort became particularly critical with the advent of AJAX applications - the AJAX page model allows content to refresh without a page reload, which could result in a significant reduction in ad impressions. Rich Media, however, is typically served with JavaScript ad calls which do not support the reloading of ad content. Furthermore, rich media ad calls may execute document.write commands that disrupt page loading or cause script errors with AJAX page content.

The solution for AJAX based sites was the creation of a Friendly IFrame (FIF). The FIF has the ability to execute a JavaScript ad call in an IFRAME in the same domain as the page. Rich media vendors have added support to move out of the FIF and follow conventions that allow the ad to be "cleaned up" so as not to impact the Web browsing

³ For an example issue with ETags, see <http://support.microsoft.com/?id=922733>

⁴ For the HTTP/1.1 RFC, see <http://www.w3.org/Protocols/rfc2616/rfc2616.html>

experience. For more information on the FIF in AJAX or other asynchronous environments, please go to www.iab.net/ajaxrichmedia.

3.2. Polite Download

Polite techniques entail showing part of the ad before page content, and then loading the remainder after the content is fully loaded. Here we are advocating minimizing the total number of HTTP requests regardless of their order relative to the page content. Once the HTTP fetches have been reduced to the smallest number possible, polite techniques are often required for rich media ads which may pull in dynamic data (streaming, interactive games, etc.) after the initial payload.

For example, use a small pre-loader or wrapper SWF file, which calls a subsequent SWF or FLV video once the page has completed loading.

Related to this, it is also advisable to delay activation of a video ad on the page until a user begins to interact with the page (e.g. mouse movement, page scrolling, etc.).

Appendix A - Testing

In order to ensure the responsiveness and reliability of both interactive ad products and the Web pages and applications that host them, it is recommended that companies engage in some level of performance testing. While the guidelines listed below are not intended to provide a comprehensive process for conducting performance tests, they do represent the basic principles necessary for gathering valid and actionable test results.

1. Establish and test Web page and ad performance against specific and measurable goals

- a. An example of an explicit performance goal would be “all pages must load within 3000 ms as measured from the initial browser navigation request to the firing of the page FinalDocComplete event”

2. Test using a WAN simulator set to both typical and worst-case network conditions

- a. A WAN simulator is a software or hardware tool for simulating real-world network conditions experienced by end users, including request latency, bandwidth, and packet loss
 - i. An example of WAN simulator settings would be 300Kbps with 300 ms of Round-Trip Time (RTT) at 3% packet loss
- b. WAN simulator settings should be tuned to match the relative conditions for all servers in the test environment
 - i. For example, CDN servers are typically located closer to the end user and, as a result, the round-trip delay would be less.
- c. Worst-case network conditions are often as poor as 50 Kbps with 600 ms RTT at 15% packet loss

3. Test page views for both first time users (with un-cached content and cookies), and returning users (with cached content and cookies)

- a. Content caching can improve page download performance significantly for regular visitors to the site ([see section 2.2 Cache Content](#)), but first-time users will not have any cached content and improving their performance may require additional optimizations
- b. The presence or absence of user state, such as preference data stored in cookies, can also affect page download performance

4. Profile Web page download patterns and network roundtrips

- a. Web page content should be profiled for performance characteristics, data size, and adherence to best-practices
- b. The number of round trip should be minimized, since the effects of request latency are amplified by the number of round trips

As performance testing proceeds, please keep a few points in mind:

1. Consider alternatives for improving page performance as it relates to ads, such as running fewer or smaller ads, or changing the page design to load ads sooner or asynchronously
2. Discuss with business partners any trade-offs between monetization and performance
 - a. For example, it is important to weigh the effect running more or fewer (or larger or more complex) ads per page will have on revenue against abandonment rates and slow download performance, which can have a negative impact on customer satisfaction and retention
3. It may be valuable to run additional tests simulating various alternatives in order to provide points of comparison for determining the most appropriate performance optimizations
 - a. For example, if regional latency to the ad selection service is a concern, simulating the network conditions as if the ad system were deployed to different regional data centers could provide valuable data for choosing between different server hosting options

In addition to performance tests, it is also recommended to verify the correct rendering of ad products. Some test cases to consider would include:

- Ads should render correctly on rollover, refresh, resize and zoom
- Expanding Ads should not displace, ghost or reorder justifications
- Expanding Ads should not adversely affect toolbars, menus, buttons and other controls
- Ad should not leave any remnant HTML or JavaScript on refresh of an AJAX page
- Any conflicts between audio, visual, other ads, flash, etc should be managed
- If an IFrame is created by the ad code being used make sure the domain is set correctly
- Document.write and Document.close should be called in the correct sequence
- Timing, data fields in ads, Z-Index and tab order should work together
- Ads should render correctly across various combinations of browsers and operating systems

Appendix B – Tools and Resources

Performance Analysis

AOL PageTest

An IE (v6, v7 & v8) plug-in that provides graphical depiction (waterfall diagram) of component loading, screen shots, and detailed reports, including details for requests, responses and the HTTP headers. It is available in both standalone (pagetest.sourceforge.net), and hosted (www.webpagetest.org) versions. This is a free and open source tool.

YSlow

A plug-in for Firefox that analyzes Web pages and explains why they are slow based on Yahoo's rules for high performance Web sites. <http://developer.yahoo.com/yslow/>. This is a free tool.

HTTP Analysis

HttpWatch

HttpWatch is an HTTP viewer and debugger that will integrate with Internet Explorer to provide seamless HTTP and HTTPS monitoring without leaving the browser window. <http://httpwatch.com/>.

Fiddler

Fiddler is an HTTP debugging proxy. It seamlessly captures HTTP traffic and logs it for the user to review. It can also be used to "fiddle" with HTTP traffic as it is being sent. By default, traffic from Microsoft's WinINET HTTP(S) stack is automatically directed through Fiddler at runtime, but any browser or application can be configured to route traffic through Fiddler. <http://www.fiddler2.com/>. This is a free tool.

Charles Proxy

Charles is an HTTP proxy / HTTP monitor / Reverse Proxy that enables a developer to view all of the HTTP traffic between their machine and the Internet. This includes requests, responses and the HTTP headers (which contain the cookies and caching information). <http://www.charlesproxy.com/>.

Wireshark (formerly known as Ethereal)

Wireshark is a packet sniffer used for network troubleshooting and analysis. Wireshark can decode many networking protocols, thus is able to display the encapsulated data fields and their meanings. <http://www.wireshark.org/>. This is a free and open source tool.

Image Optimization

For JPEG: <http://www.iijg.org/>

For GIF (Converts GIF to PNG): <http://www.imagemagick.org/>

For PNG: <http://pmt.sourceforge.net/pngcrush/>

JavaScript Optimization and Best Practices

JavaScript Best Practices: <http://javascript.crockford.com/code.html>

JSMIn

A tool which minifies JavaScript for optimal performance: <http://www.crockford.com/javascript/jsmin.html>

JSLint

JavaScript verification with an "adsafe" option: <http://www.jshint.com/>

Checklists

Agency/Creative Shop Checklist



- Reduce components** *(Section 1.1)*
- Reduce HTTP requests including redirects** *(Section 1.2)*
- Reduce size by compressing and minifying components** *(Section 1.3)*
- Optimize cookie usage** *(Section 1.3)*
- Optimize Images⁵** *(Section 1.3)*
- Do not Scale Images⁶** *(Section 1.3)*
- Ensure creative conforms to IAB guidelines⁷** *(See IAB Guidelines at www.iab.net)*
- Use Polite Download where possible** *(Section 3.2)*

Ad Vendor Checklist



- Use a CDN** *(Section 2.1)*
- Reduce HTTP Requests including redirects** *(Section 1.2)*
- Reduce DNS Lookups** *(Section 1.2)*
- Make use of Caching / Expires Headers** *(Section 2.2)*
- Compress Content** *(Section 1.3)*
- Make JavaScript and CSS External** *(Section 1.3)*
- Optimize Cookie Usage** *(Section 1.3)*
- Enable HTTP Keep-Alives** *(Section 2.3)*
- Limit 4th and 5th Party Serving**

⁵ http://developer.yahoo.com/performance/rules.html#opt_images

⁶ http://developer.yahoo.com/performance/rules.html#no_scale

⁷ http://www.iab.net/iab_products_and_industry_services/1421/1443/Ad_Unit