ADTECH

HTML BANNERS

2015-08-13

Checklist

1. Do not build a basic structure of a html-doucument since the banner almost always is being delivered on a site where this already exsists (exceptions are IFRAMES)

2. Pull all neccesary scripts within the index.html (root directory)

3. Pull all files into the index.html file (the root directory)

4. Avoid using a folder structure when creating the path to files in the index.html-file

5. Adserver variables like _ADPATH_ should be used to create a path to files uploaded in IQ, Adserver varible for counting clicks should be used

6. Use unique identifiers, classes, variables and functions per banner to avoif conflicts with elements on the site.

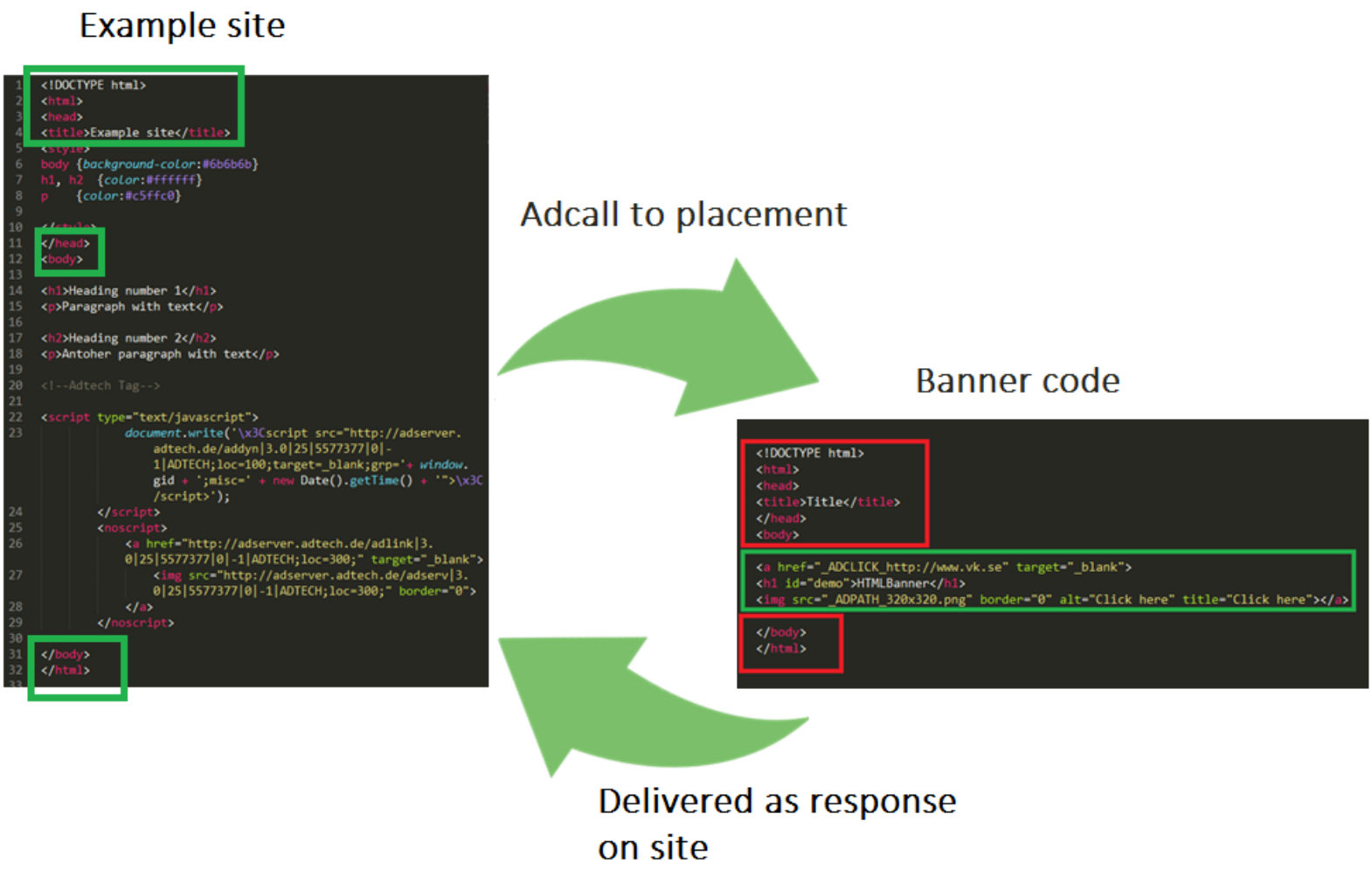
7. If external JS-liberaries are used the scriptloader might need to be used

Do not build the basic structure of a HTML-file

When booking a banner in IQ it should not have the basic structure of an html-document. This is due to the banner almost always being delivered on a website that already has these elements (exceptions are IFRAMES).

Below you can see a example of a basic HTML-structure, including the html, head, title, body elements and the ADTECH-placement. The banner code will be delivered as response on the site which already includes the basic structure of the HTML-document.

Colorcoded as red in the banner code example is not needed since the elements (colorcoded as green in the example site example)already exists on the site.



Pull all neccesary scripts within the index.html file

We recommend that all scripts are pulled from the index.html file (root directory), using external JS-libraries can sometimes cause issues when not executed in the right order in relation to the code used in the index.html-file. This can in some cases be handled by using the Scriptloader developed by Fredrik Borgren, please see separate section called *Scriptloader*.

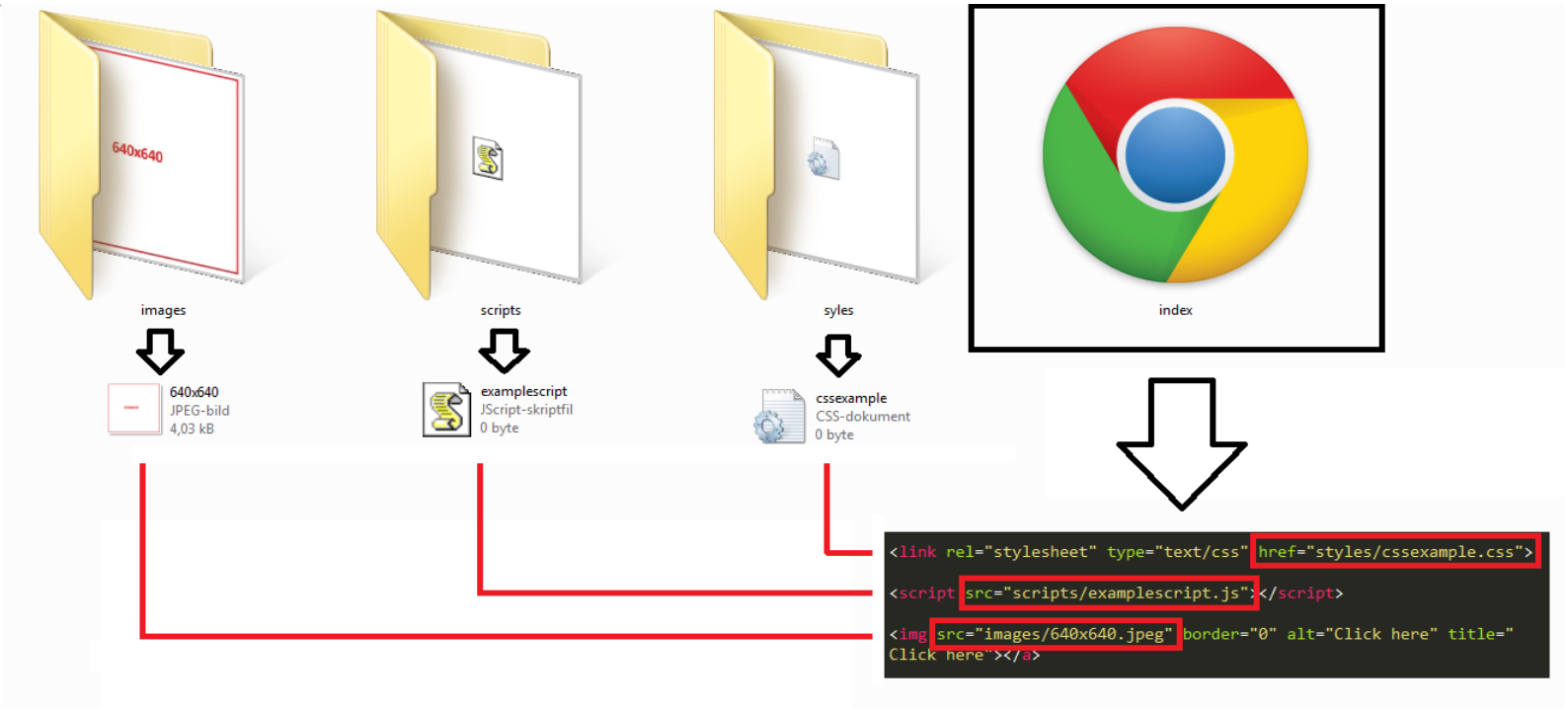
Pull all Files into the root directory and try not to use a folder structure

All files should be pulled in the index.html-file (the root directory) since adserver variables like _ADPATH_ should only be used in the index.html file.

Also we recommend that a folder structure is not used when building a banner that should be booked in IQ.

A lot of times the banner-project is structured as the following example:

Here the external files are pathed from other folders than the index.html-file

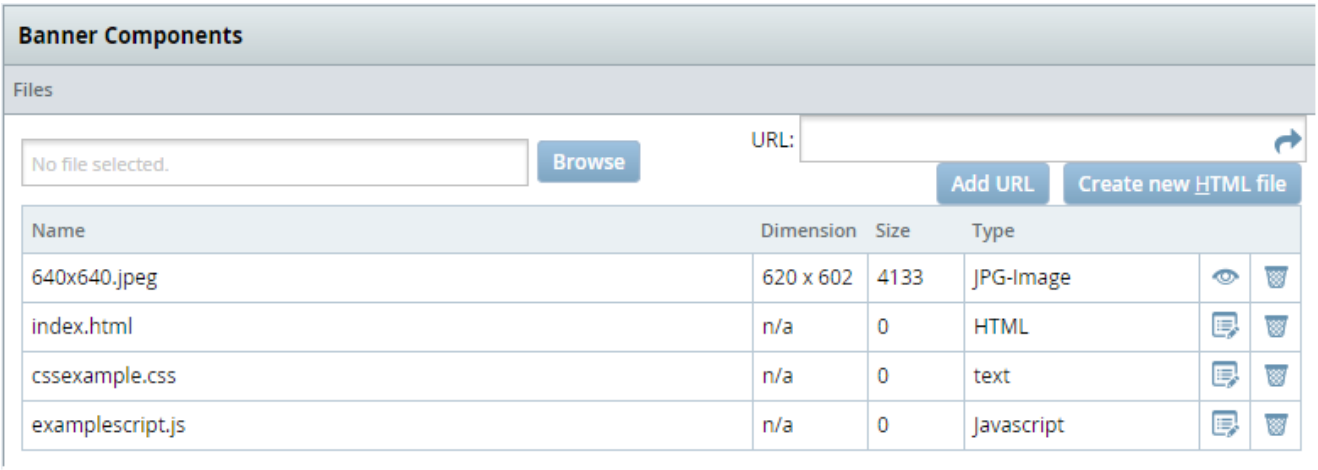


For example the below code will try to search for/get the images that is called “640x640.jpeg” (colorcoded as green) from a folder called “images” (colorcoded as red)

```
<img src="images/640x640.jpeg" border=”0” alt=”Click here” title=”click here”>
```

Uploading this into IQ can be an issue since the pathing to the images can be incorrect, due to them not actually being in different folders when uploaded into the campaign.

Example on how the files will look when uploaded in IQ



In the example you can see the files are not in different folders when uploaded in IQ, this means the image pathing in our earlier example would not work.

Note: *Sometimes if the banner is uploaded from a .zip file the files can be shown with names like "foldername/imagename.jpg", if this happens we recommend that the files are uploaded one by one. Please read more about this under "troubleshooting" → "Examples of image pathing errors" → "exception"*

To see how the pathing to a file looks in the html-file you can press "edit" on the html-file in "banner edit" → press ctrl+f → and search the document for the filename. If there is a "/" in front of the name it can mean it is trying to path the file from a folder-structure, the "/" and folder-name should be removed and replaced with _ADPATH_, please see more examples on how to use ad server variables in the next section.

Commonly used Ad server varibles in HTML-banners

- Ad server variables should only be used within the index.html-file (root directory), the adserver will not replace adserver varibles used within JS-files, additional HTML-files or any other file types
- _ADPATH_ - is used for the correct pathing to files uploaded in IQ – it will be replaced with the path of the file, most cases it will be automatically detected during the banner conversion, please see example on how _ADPATH_ can be used below under “Where _ADPATH_ should be used”
- _ADPATH_ should not be used when the file is not uploaded in IQ, please see example below, under “where _ADPATH_ should not be used”
- _ADCLICK_ - Is used to count clicks (_ADCLICKESC_ and_ADCLICKDEC_ can also be used, please read more about our click specific ad server varibles in the common Adtech documentation here)

Example on how the varibles can be used

```
<!--Where _ADPATH_ should be used-->
<link rel="stylesheet" type="text/css" href="_ADPATH_cssexample.css">

<script src="_ADPATH_examplescript.js"></script>

<a href="_ADCLICK_http://www.aol.com" target="_blank"><img src="_ADPATH_300x250_blau.jpg" width="300"

<!--Where _ADPATH_ should not be used-->
<script src="http://example/script/externalscriptfile.js"></script>
```

Note: *Some variables will be added automatically under the banner conversion, to see what happens with the code you can use the “unlock code”-button in “banner edit” and open the .html-file to check what has been added.*

1. *If you see a file that is uploaded in IQ without _ADPATH_ added, please add “_ADPATH_”*
2. *If you see a link to a file that is not uploaded in IQ with _ADPATH_ in front of it, please remove “_ADPATH_”*
3. *Above example can be used as reference, but please note that it is only an example and that the code in your banner may differ*

Use unique identifiers, classes, varibles and function names per banner

When building a HTML-banner in we recommend that you use unique identifers and classes. This is because the banner will be delivered onto a site containing other HTML-elements, CSS and JavaScript code – this can sometimes result in conflicts between the code in the banner and the code on the site.

An example of this could be setting a CSS-style to a body-element in the banner → this is then delivered onto the site also containing a body element. When the banner is delivered on the site this CSS-style can in some cases also be set on the body-element on the site and therefore cause a conflict – therefore we recommend unique identifiers to be used in the banner.

Example of unique identifiers

- Class – should be used to identify more than one element, selected by ”.” in CSS
- ID – should be used to identify one element, selected by ”#” in CSS

Example of how the ID and class can be used

```
<div id="idexample"></div>

<p class="classexample"></p>

<style type="text/css">
#idexample {
    background-color: #ccc;
}

.classexample {
    color: red;
    font-weight: bold;
}
</style>
```

The example CSS-code will only be set on the elements that have the given ID (green) or the given class (blue)

Note: *we always recommend that Banners are tested IP-targeted live on the site to make sure there are no conflicts with for example styles on the site and styles coming from the banner.*

Scriptloader

We recommend that all scripts are used in the index.html (root diretory); if the banner is using an external .js-file there can sometimes be issues. When the banner is using a external JS-file and code related to that file is used in the index.html the external file needs to be loaded before the code is executed. Sometimes this is not executed in the right order, but can be handled by using the scriptloader.

Please see example below

Here we have a function declared in an external JS-file called **code.js**

#code.js, myFunction declared in external jacaScript file:

myFunction = function() {

//JavaScript code goes here

};

We then want to use this function in the index.html

#index.html, rootdirectory calls myFunction declared in code.js

myFunction();

In this case code.js needs to be loaded before myFunction(); is executed in the index.html – if it is not the function will not have been defined before it's executed. When using external JS-files in banners this can often happen when loaded onto the site. This can often be handeled by the scriptloader, please also see separate .pdf for the scriptloader.

Please see example on how the scriptloader can look when used below

How to implement the scriptloader

1. You can create a new script-block in the index.html-file (root directory)

```
<script type="text/javascript">

</script>
```

2. Add the original or minified version of the function found in the scriptloader .pdf under "sources"

```
<script type="text/javascript">
// Original:

            window.adtech = window.adtech || {}
            window.adtech.load = window.adtech.load || {
                js: function( url, callback ) {
                    var a = document.createElement("script");
                    a.src = "" + url;
                    a.onreadystatechange = a.onload = function() {
                        var c = a.readyState;
                        callback.done || c && !/loaded|complete/.test( c ) || ( callback.done = true, callback() )
                    };
                    document.getElementsByTagName("head")[ 0 ].appendChild( a )
                }
            };
</script>
```

3. After that add the code found under "usage" in the scriptloader .pdf, dependant on how many libraries should be used.

   Single library:

```
adtech.load.js('http://example.com/library.js', function() {
console.log('done!');
});
```

   Multiple libraries (two):

```
adtech.load.js('http://example.com/library1.js', function() {
adtech.load.js('http://example.com/library2.js', function() {
console.log('done!');
})
});
```

   Note: more than two libraries can be used

4. Change the example url to the name of the external JS-file or URL to the file, please see green marked below

```
<script type="text/javascript">
// Original:

            window.adtech = window.adtech || {}
            window.adtech.load = window.adtech.load || {
                js: function( url, callback ) {
                    var a = document.createElement("script");
                    a.src = "" + url;
                    a.onreadystatechange = a.onload = function() {
                        var c = a.readyState;
                        callback.done || c && !/loaded|complete/.test( c ) || ( callback.done = true, callback() )
                    };
                    document.getElementsByTagName("head")[ 0 ].appendChild( a )
                }
            };

            adtech.load.js('http://example.com/library.js', function() {
            console.log('done!');
            });
</script>
```

5. Move all the javaScript code in the index.html-file in the correct order, top →down, in the –html-doucument to where it now says “console.log(‘Done!’);”, please see green marked below:

```
<script type="text/javascript">
// Original:

            window.adtech = window.adtech || {}
            window.adtech.load = window.adtech.load || {
                js: function( url, callback ) {
                    var a = document.createElement("script");
                    a.src = "" + url;
                    a.onreadystatechange = a.onload = function() {
                        var c = a.readyState;
                        callback.done || c && !/loaded|complete/.test( c ) || ( callback.done = true, callback() )
                    };
                    document.getElementsByTagName("head")[ 0 ].appendChild( a )
                }
            };

            adtech.load.js('http://example.com/library.js', function() {
            console.log('done!');
            });
</script>
```

Basic steps of troubleshooting

1. Does the banner work as expected outside of Adtech? – Please always make sure the banners is working correctly outside Adtech
2. Test the banner live with IP-targeting:
    a. Is the banner shown? If not, is the files in the banner correct? – meaning not using a folder structure and that _ADPATH_ is used correctly
    b. Are there any conflicts with the site? If there is – what is happening? – is there CSS or other code in the banner that applies to elements that also exist on the site?
    c. Is there functionality that should be attached to the banner that is not working? – If so, does the banner use an external JS-library? If yes, test using the scriptloader

Console errors

It is often a good idea to check the console to see if there are any errors from the banner, the console can in Chrome for example be opened by pressing f12.

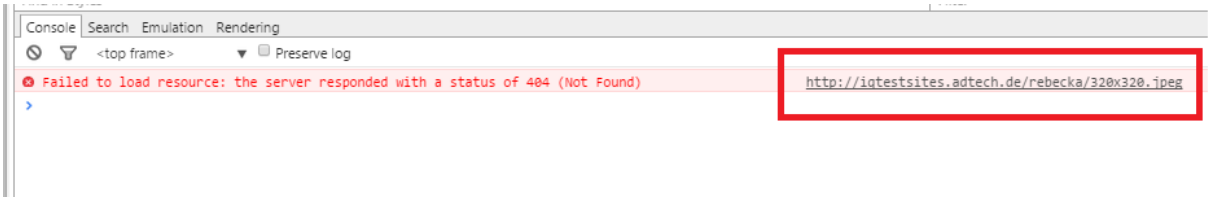
Examples of image pathing errors

Example 1

This is the code in the banner, please see the image pathing maked with red:

<img src="320x320.jpeg" border="0" alt="Click here" title="Click here">

Since the image is uploaded in IQ it should have the _ADPATH_ varible infront of it. Not having this will result in a not found error, please see screenshot from the console below:



Correct code would be:

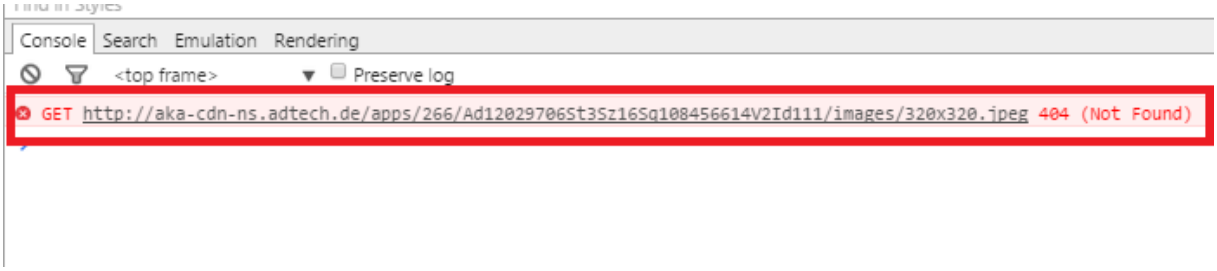
<img src="_ADPATH_320x320.jpeg" border="0" alt="Click here" title="Click here">

Example 2

Here the banner code has _ADPATH_ added but also a folder structure in the path to the image, see colormarked in red below:

<img src="_ADPATH_images/320x320.jpeg" border="0" alt="Click here" title="Click here">

The image is uploaded in IQ without a folder structure which will result in a not found error

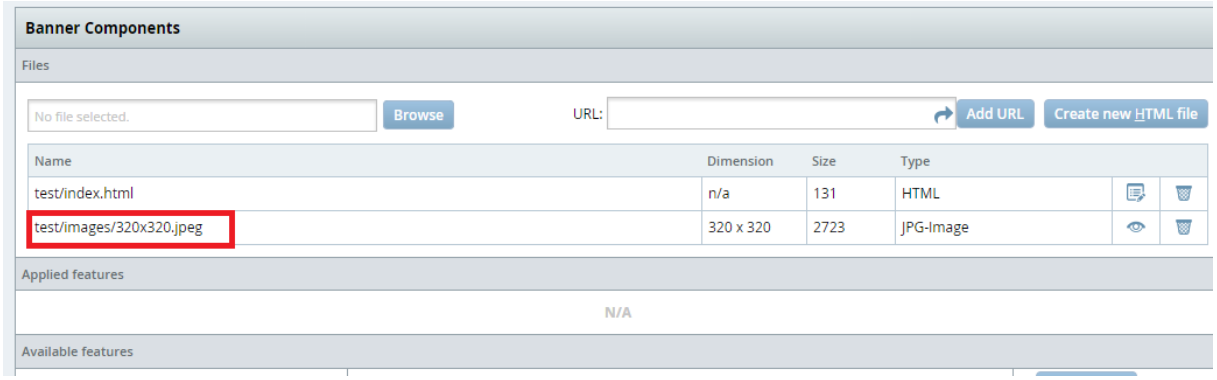


Correct code would be:

<img src="_ADPATH_320x320.jpeg" border="0" alt="Click here" title="Click here">

Exception

We recommend a folder stucture should not be used, if it is and the file is uploaded into IQ via a .zip-file it can be uploaded with this structure, it can look like this in IQ front-end:



The following pathing to the file can work:

<img src="_ADPATH_test/images/320x320.jpeg" border="0" alt="Click here" title="Click here">

Example of a function that is not defined, often a indication that the scriptloader can be used