

SeeWhy Abandonment Tracker

Tagging Reference

Version 2.1

©2010 SeeWhy Inc. All rights reserved.

Table Of Contents

Tagging Reference Guide	1
Tagging with Headers and Footers	1
Tagging with Content Management Systems	3
Introduction	3
Drupal	4
Example PHP Tagging Strategy	4
Example PHP Code	6
Joomla	8
Adding JavaScript to Content	8
Other CMS	9
osCommerce	9
Mambo and WordPress	10
Multiple Tags	11

Tagging Reference Guide

Tagging with Headers and Footers

The process of tagging can be simplified by splitting the page code across the header, body and footer, as in the following examples.

This page code in the header is used to set up defaults. The example here sets Funnel Level, which must be set on every page that calls the tag JavaScript file. If you wanted to set other values for every page, e.g. one of the custom fields (cy.Custom1) you might also choose to do that here.

```
<!-- SeeWhy Abandonment Tracking Tag - header -->  
<img id="cy_image" width=1 height=1 border=0 alt="">  
<script src="WebEvent.js" type="text/javascript">  
</script>  
<script type="text/javascript">  
<!--  
cy.FunnelLevel="0";  
/-->  
</script>
```

Page code in the body would then only be required for key pages, allowing you to specify a Funnel Level and perhaps a User Id just where necessary.

```
<!-- SeeWhy Abandonment Tracking Tag - body -->  
<script type="text/javascript">  
<!--  
// set funnel level on key pages in the process  
cy.FunnelLevel="7";  
// set UserID - ideally on the first page after this is known  
//cy.UserId="%uid"  
/-->  
</script>
```

SeeWhy Abandonment Tracker

The page code that actually activates the tag can be included in the footer, as in this example. Or could have been included with the body code if you did not want to track every page.

```
<!-- SeeWhy Abandonment Tracking Tag - footer -->  
<script type="text/javascript">  
<!--  
cy_getImageSrc();  
//-->  
</script>
```

Tagging with Content Management Systems

Introduction

Content Management Systems (CMS) can make it more difficult to access the source code of your web pages where you need to add SeeWhy tag page code. However, they do offer facilities that not only make this possible, but also make the longer term maintenance of your tagging much simpler.

All Content Management Systems vary, but in general they separate the content, i.e. what visitors to your site see and read, from the HTML structure that delivers this as a web page. The final web page is usually created on demand from the stored content.

Tagging requires JavaScript page code to be present on each page you wish to track when it is served. The SeeWhy service also requires a funnel level for every event it receives, and so a mechanism to set this, along with any other optional variables you wish to use, must also be provided.

There are usually many ways of achieving this within a CMS, and this guide includes just some examples for the most popular systems, but the same principles will apply to all. Exactly which method you choose will depend on your requirements and the structure of your web site. It may be as simple as enabling JavaScript and then adding the page code directly to your content, or it may involve, for example, the use of PHP code and special features of the CMS.

Drupal

Example PHP Tagging Strategy

As well as the standard SeeWhy WebEvent.js, the following PHP tagging example requires two files that you will need to create and deploy on your site:

- *seewhytag.inc* - The code in this file generates JavaScript that will invoke the SeeWhy tag.
- *seewhytag-drupal.inc* - This file contains Drupal specific code that will package up your data and pass it to SeeWhy by calling the code in *seewhytag.inc*

You can create these files by cutting, pasting and modifying as required the PHP example code in next section of this guide.

Having created those files, the steps below show how 'FunnelLevel' can be set and passed through for each page. This is the minimum amount of data required for the SeeWhy service, but it is likely you will want to pass more information, such as a user ID for follow up emails, custom fields for context etc The example code can easily be extended to cover this, either using further custom CCK fields,, or other existing variables, such as a global user identifier.

1. Install and enable the Drupal CCK module. Instructions for this can be found at drupal.org/project/cck
2. Use CCK to define a custom field called *seewhy_funnel_level*. This should be a text box backed by an integer field which should default to - 1.
3. Add this custom field to all the content types that you might be likely to tag, e.g. Page, Product etc.
4. Ensure that this custom field is not displayed on any content type.
5. Define a block with the PHP content of the form shown below:

```
<?php require "seewhytag-  
drupal.inc";add_seewhy_tag('field_seewhy_funnel_level','WebEvent.js');  
?>
```

6. The block above should be shown on all nodes.
7. Enter a value for the custom field `seewhy_funnel_level` greater than 0 for any pages/content you want to track. See the User Guide for full details of how to use Funnel Level. Pages where you do not set `seewhy_funnel_level` will not send data to SeeWhy.
8. Ensure that the `seewhytag.inc`, `seewhytag-drupal.inc` and `WebEvent.js` files are deployed in the root directory of your web site.

Now when visitors come to your web site data for the pages you are tracking will be sent to SeeWhy.

SeeWhy Abandonment Tracker

Example PHP Code

seewhytag.inc

To extend this function pass additional variables in and add extra print statements of the form used for FunnelLevel where shown.

```
<?php
// prints the seewhy tag to the output for the passed arguments
function print_seewhy_tag($scriptfile,$funnellevel) {
    print('<img id="cy_image" width=1 height=1 border=0 alt="">'. "\n");
    print('<script src="'. $scriptfile. '"
type="text/javascript"></script>'. "\n");
    print('<script type="text/javascript">'. "\n");
    print('cy.FunnelLevel="'. $funnellevel. '";'. "\n");
    // add extra tag parameters here
    print('cy_getImageSrc();'. "\n");
    print('</script>'. "\n");
}
?>
```

seewhytag-drupal.inc

```
<?php
// Allows content under Drupal, with the CCK module to be tagged for
the SeeWhy service

error_reporting(E_ALL);
require_once './includes/bootstrap.inc';
require 'seewhytag.inc';
drupal_bootstrap(DRUPAL_BOOTSTRAP_FULL);

// gets the nodeid for the passed url
// the url should be relative to the root of the website
// the url may be a real url or an alias
// if a valid node id cannot be found -1 will be returned
function get_nodeid_from_url($url) {
    // get the real node id if this is an alias
    $source = drupal_lookup_path('source',$url);
    if(isset($source) && $source!="") {
        $url = $source;
    }
    // split up the url and extract the node id (nid)
    $splitarr = split('/', $url);
    $type = $splitarr[0];
    if($type!='node') {
        return -1; // not a node
    }
    $nid = $splitarr[1];
    if(empty($nid) || $nid==' ' || !is_numeric($nid)) {
        return -1; // failed to get the nid
    }
    return $nid;
}
```

```

}

// gets the passed cck field for the node with the passed id
// the field is assume to be an integer and if not found, 0 is returned
function get_cck_integer_field($fieldname,$nid) {
    $tablename = 'content_'. $fieldname;
    $val=(Integer)0;
    // check if the table exists
    if(!db_table_exists($tablename)) {
        $msg = "ERROR: No cck table exists with name $tablename for the
cck field $fieldname".
        ". Please ensure you have the cck module installed and
enabled.";
        print($msg."\n");
        error_log($msg);
        return;
    }
    // it exists so get the first value for the passed node (if there is
one)
    $query = 'select '.$fieldname.'_value from '.$tablename.' where
nid=' . $nid;
    $results = db_query($query);
    if($result = db_fetch_array($results)) {
        $rawvalue = $result[$fieldname.'_value'];
        if(is_numeric($rawvalue)) {
            $val=(Integer)$rawvalue;
        }
    }
    return $val;
}

// adds the free seewhy tag for the current page if a funnel level is
defined
function add_seewhy_tag($fieldname,$scriptfile) {
    // extract the node id from the current url
    $nid = get_nodeid_from_url($_GET['q']);
    if($nid<0) {
        return;
    }
    // got the node id: get the funnel level if possible
    $funnellevel=(Integer)get_cck_integer_field($fieldname, $nid);
    if($funnellevel<0) {
        return;
    }
    // got a funnel level: print the tag
    print_seewhy_tag($scriptfile,$funnellevel);
}
?>

```

Joomla

Adding JavaScript to Content

The following steps will allow you to enter SeeWhy Tag page code directly into your Joomla content.

1. Copy your WebEvent.js into the root directory of your web site
2. Login as the Joomla Administrator and go to Site->GlobalConfig.
3. Set the default WYSIWYG Editor to 'No Editor'. You will now be able to define JavaScript in your content.
4. Add the SeeWhy page code JavaScript to content on the pages you want to track, for example:

```
<img id="cy_image" width=1 height=1 border=0 alt="">  
<script src="WebEvent.js" type="text/javascript">  
</script>  
<script type="text/javascript">  
<!--  
cy.FunnelLevel="0";  
cy_getImageSrc();  
//-->  
</script>
```

4. Save the new content.

Of course you may alternatively choose a PHP based tagging approach as described for Drupal

Other CMS

osCommerce

osCommerce is an open source online shop program consisting principally of separate PHP pages for each stage of the checkout process. It is best tagged by modifying each of the pages you wish to track.

You will need to create a *seewhytag.inc* file, as detailed in the Drupal PHP example. You can create this file by cutting, pasting and modifying as required the PHP example code in the Drupal section of this guide.

Having created this file, the steps below show how 'FunnelLevel' can be set and passed through for the checkout confirmation page. This is the minimum amount of data required for the SeeWhy service, but it is likely you will want to pass more information, such as a user ID for follow up emails, custom fields for context etc. The example code can easily be extended to cover this, by changing the `print_seewhy_tag()` function and calls that are made to it accordingly.

1. Copy the *seewhytag.inc* and *WebEvent.js* into the root directory of your web site.
2. Open the *checkout_confirmation.php* file in the root directory using a text editor.
3. Append the following to the file, just before the 'load the selected shipping module' comment:

```
<?php require "seewhytag.inc";print_seewhy_tag('WebEvent.js',7);?>
```

4. Save the file.

You will need to repeat steps 2 through 4 for each page you wish to tag.

Mambo and WordPress

A similar approach can be used for both Mambo and WordPress. The steps below involve adding JavaScript directly to content on the pages you wish to tag.

1. Copy your WebEvent.js into the root directory of your web site.
2. Login to the CMS as user with rights to create content.
3. Add the SeeWhy page code JavaScript to content on the pages you want to track, for example:

```
<img id="cy_image" width=1 height=1 border=0 alt="">  
<script src="WebEvent.js" type="text/javascript">  
</script>  
<script type="text/javascript">  
<!--  
cy.FunnelLevel="0";  
cy_getImageSrc();  
/-->  
</script>
```

4. Save the new content.

Multiple Tags

You may have multiple SeeWhy tags either to track different processes in the same web site, or to track multiple web sites that you are managing.

In this case you will have received multiple mails from SeeWhy, one for each service request. Each will contain a JavaScript file that you will need to re-name. This file contains your unique customer service code, so it is important that you keep track of which file is used where, to ensure that if you need to update your service in any way that the correct file is replaced.

If in any doubt you can open the JavaScript file in a text editor and look at the value for CustomerCode, this is the unique service code. You will also see this code in the title of your daily email summary.

Giving each of your JavaScript files a name that includes this number, and modifying the src statements in your page code may make things easy to track. For example, if you have customer service codes 123456 and 789123, you could name the relevant JavaScript files WebEvent123456.js and WebEvent789123.js

The page code for service 123456 would look like this:

```
<!-- SeeWhy Abandonment Tracking Tag -->
<img id="cy_image" width=1 height=1 border=0 alt="">
<script src="WebEvent123456.js" type="text/javascript">
</script>
<script type="text/javascript">
<!--
cy.FunnelLevel="0";
cy_getImageSrc();
//-->
</script>
```

and for service 789123:

```
<!-- SeeWhy Abandonment Tracking Tag -->
<img id="cy_image" width=1 height=1 border=0 alt="">
<script src="WebEvent789123.js" type="text/javascript">
</script>
<script type="text/javascript">
<!--
cy.FunnelLevel="0";
cy_getImageSrc();
//-->
```

SeeWhy Abandonment Tracker

</script>

The line that changes is shown in bold.