Instructions for Embedding a "Kudos" Display within Your Website

You may use either of two technologies for this embedment.

A. You may directly insert the underlying PHP code; or

B. You may insert some JavaScript that acts essentially as a wrapper for the PHP. (In other words, the JavaScript does the "talking" to the PHP for you, so all your own work has to do is "talk" to the JavaScript.) If this is your preference, skip to Page X.

If you are using WordPress (or any other web-development environment that cannot directly consume PHP), you'll want to use Method-B. If you prefer simplicity and ease in coding for the incorporation, you'll likely also want to use Method-B. If you are using a more conventional and standard web-development environment that easily consumes PHP,[1] and if you want to maximize efficiency of performance and/or you want greater power in maximally fine-tuning and customizing the displayed result, it's likely you'll prefer Method-A.

Aside from the choice of technological method, you should also consider content options. Though it's up to you, we have designed the underlying machinery with the thought you will likely want to use both of two elements:

1. A *spontaneous* display of up to 5 customer comments. By "spontaneous," we mean it's our intent for this display to show on your webpage without the browsing consumer doing anything to specifically request it. It is simply an automatic part of any webpage in which you design to place it. You may design to spontaneously include only a single such comment, or maybe three, or perhaps four or even five. The general idea is they are standard feature (in whatever quantity up to five that you specify) of any webpage you place them in.

2. A *requested* display of up to 100 customer comments. By "requested," we mean it's our intent for this display to be used when you've provided a button, next to the spontaneous display, that's labeled with something akin to "Show more . . . " The browsing consumer see the small sampling of comments, is interested, and so clicks on that button. Now you want to show them a larger quantity, so as to really drive home the sale.

The above represents our thinking of how you will likely want to use these tools, but it's up to you. The mechanisms by which your website will pull comments (and specify the quantity to be pulled) are the same in either instance. The website design (what elements in your website cause what pulls) is completely up to you. There is one significant structural distinction, vis-à-vis Rossware systems, and it has to do with billing.

[1] If you are using either of the Rossware-hosted web platforms, PHP compatibility is assured. This means, for you at least, Option-A is at least a *candidate* for consideration.

For billing purposes, we classify any pull that requests a comment quantity up to five (as you might use under Scenario-1 above) as a "*small view*." We classify any pull for more than five comments (as you might use under Scenario-2 above) as a "*large view*." The rate for large views is greater than the rate for small views (see http://rossware.net/MiniManuals/CyberOfficeRateSheet.pdf). We thinks makes sense. If your viewing prospect has explicitly requested to see comments, it's likely worth significantly more than when you are spontaneously showing such comments to each and every browser.

It would certainly be possible, BTW, for you to configure your website to show only five comments even when triggered by an explicit browser's request. If you did so, indeed, you'd only incur "small-view" fees. However, we think in such a case you'd be "penny wise and pound foolish." By design, the comments as shown are random (well, random among those that rise to the minimum score level you have select for inclusion). This means, if your browsing prospect was anxious to read a bunch of comments (as a person in that situation often is), and so repeatedly clicked to "Show More," there's a significant chance they'd see some of the same comments repeatedly, and therefore fail to expose themselves to as much content as is really wanted. If you instead provide them with, say, up to 50 comments on the basis of one request, they'll instantly have a wealth of persuasion in front of them, and with no repetition.

So, back to technology of choice, if you prefer Method-B, please skip to Page 9. Otherwise (i.e., for Method-A), simply proceed to the next page.

METHOD-A:
DOING A DIRECT-INSERT OF PHP INTO YOUR WEBPAGE.

[This section was written by our Josh Smith, and in some instances he reverts to first-person.]

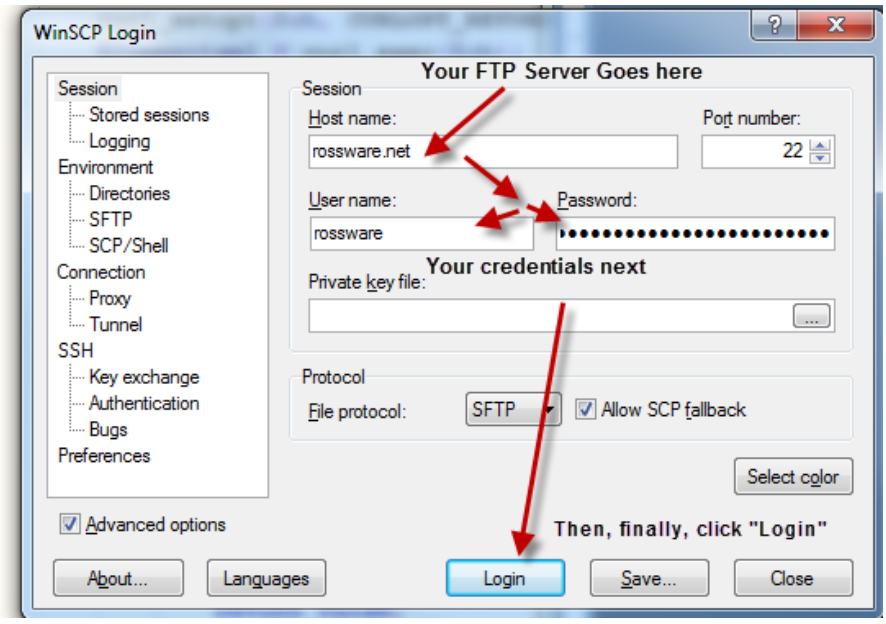
If you're already familiar with PHP, it's likely you do not even need specific instruction. You can just use this link to get the needed code (http://my.rossware.net/SurveyWidget_s.php), modify as desired, and insert appropriately.

If you are not familiar with PHP, we here provide a tutorial on how to do the PHP-direct insertion.

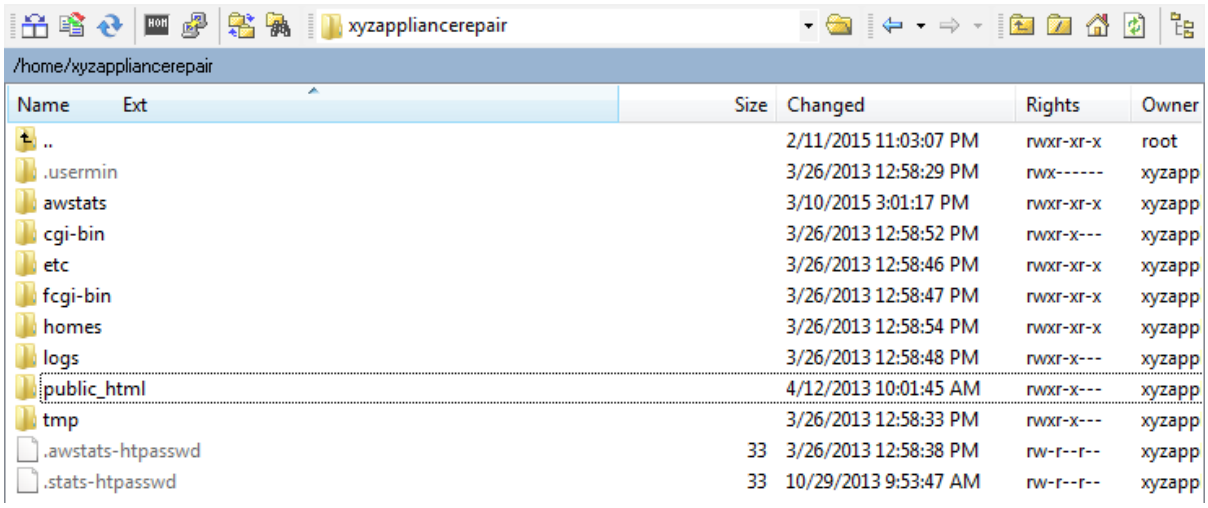
To begin, you will need to have a few tools handy. One is a web-editing tool. For instruction purposes here, I'm going to work with *WinSCP*. This is a free program that allows you to connect to your webhosting environment, and there modify files. If you decide to use a different web-editing tool, simply adapt these instructions to fit its methods.

I'm also using a text editor called *Notepad* ++, which is another program you can download and use for free (just Google it). You may use other text editors if you like, but I find this one is very effective.

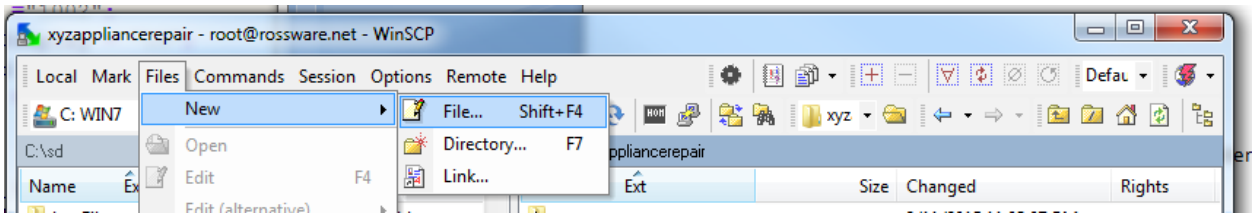
I'm going to connect via the sFTP protocol, as you'll see in this illustration. I've also entered my credentials (yours will be different, of course), and then clicked "login". You can optionally save these credentials for use next time.



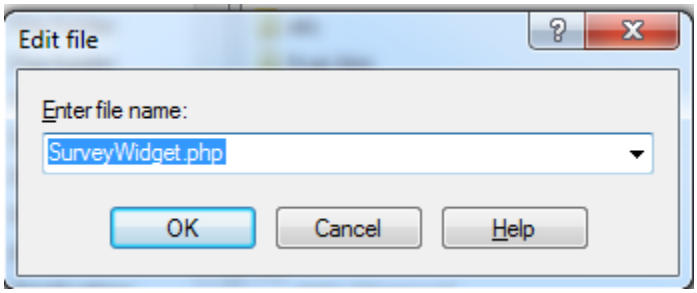
After you've logged in, you'll be placed in your default folder; it should look something like this.



Open up the public_html directory (or maybe you have a www directory) and then click **Files > New > File**



Then, name your file, and click OK to proceed.



Now that you have a file with which to work, you'll need to copy and paste a couple pieces of code. Let's start with the easy stuff.

```
<html>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.3/jquery.min.js"></script>
<link rel="stylesheet" type="text/css" href="http://my.rossware.net/lib/css/stars.css" />
<script type="text/javascript" src="http://my.rossware.net/lib/js/stars.js"></script>
```

These three lines of code (four lines if we count the HTML tag) do the following

1.) Load the jquery-minimal plugin from Google.
2.) Load the stars.css style sheet that stylizes the star display for your survey result
3.) Load the jquery script that correctly displays stars based on the survey result.

I would like to point out that you may download a copy of the CSS and JS files to your own web server and modify them however you please. Regardless, please understand that we offer no troubleshooting on any modifications you may choose to make.

Now for the fun part. Here is where we get into actual programming.

The next line in your PHP file should be the PHP open tag.

```
<?php
```

Then add in a couple configuration variables.

```
// Configuration parameters
$ClientID ="1001";
$NumberToDisplay = "1";
$Threshold = "9.5";
```

These variables should be fairly easy to understand, but I'll go over them to assure clarity:

$ClientID – This is your 4 digit SDM/SDCO login ID. Make sure to set this correctly.

$NumberToDisplay – This number indicates how many results you want to display when displaying the widget.

$Threshold – This is a big one. Everyone is bound to receive a negative review now and then, and while we may want to know if we're perceived badly, potential customers are better off seeing just the better reviews. We take the result of the 4 questions asked, and average them out. We can tell the WebService XML that we only want to see results that are equal to, or greater than, the number provided here. We've found that 9.5 gives a good mix of four and five stars.

Next we must create a function to deal with the XML results from CyberOffice.

```
function produce_XML_object_tree($raw_XML) {
	libxml_use_internal_errors(true);
	try {
		$xmlTree = new SimpleXMLElement($raw_XML);
	} catch (Exception $e) {
		// Something went wrong.
			$error_message = 'SimpleXMLElement threw an exception.';
		foreach(libxml_get_errors() as $error_line) {
			$error_message .= "\t" . $error_line->message;
		}
		trigger_error($error_message);
		return false;
	}
	return $xmlTree;
}
```

I know: the above looks complicated. Here's the gist of it:

1.) Define a function, and name it produce_XML_object_tree, accept one parameter
2.) Use the xml libraries internal error system
3.) (this line starts with $xmlTree) Define a new object, which is a new simple xml element that contains the raw XML from the parameter established in the function earlier
4.) If there's an error, catch it here
5.) Display each and every error message we come across
6.) If there's an error, log the error message to the web-server log
7.) (Still in error handle), if we've encountered an error, return false (the execution fails)
8.) Otherwise, return the $xmlTree object we defined.

With the function defined, we move to heavy lifting. First we must download XML from the WebService. This is where the variables we created will come into play.

```
//The Next part should be one line.
$xml_feed_url = 'http://my.rossware.net/GetReviewsAsXML.php?ClientID='. $ClientID .'&limit='. $NumberToDisplay .'&avg=' .
$Threshold;
//The above is one line.
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL, $xml_feed_url);
        curl_setopt($ch, CURLOPT_HEADER, false);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        $xml = curl_exec($ch);
        curl_close($ch);
```

Please note, though the line starting with "$xml_feed_url" is displayed here on two lines, it actually is only a single line of genuine text.

The object defined as $ch is a curl handler; and is what actually does the work to go out to the Rossware server to download the XML response.

Next, we'll take the raw XML response, send it to our function, and create an object that has the xml object response. We'll also define a counter to track the number of survey entries we're going to parse through.

```
        $SurveyResults = produce_XML_object_tree($xml);
        $entry_count = 0;
```

Now all that's left is to loop through the results. This is where, if you choose to, you may change what's displayed and how it is displayed. I'll include a list of all possible variables at the end of this tutorial.

```
foreach ($SurveyResults->Review as $Review) {

$Comments = $Review->CstmrCmnts;
$Comments = preg_replace('/[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}/i','(phone hidden)',$Comments); // extract email
$Comments = preg_replace('/(?:(?:\+?1\s*(?:[.-]\s*)?)?(?:\(\s*([2-9]1[02-9]|[2-9][02-8]1|[2-9][02-8][02-9])\s*\)|([2-9]1[02-
9]|[2-9][02-8]1|[2-9][02-8][02-9]))\s*(?:[.-]\s*)?)?([2-9]1[02-9]|[2-9][02-9]1|[2-9][02-9]{2})\s*(?:[.-]\s*)?([0-
9]{4})(?:\s*(?:#|x\.?|ext\.?|extension)\s*(\d+))?/','(email hidden)',$Comments); // extract phonenumber
$Comments = stripslashes(html_entity_decode(str_replace("\r\n", " ", $Comments)));
```

```
echo '<div id="content" class="survey"><i><font style="size:9px;">' . date( 'm-d-Y', strtotime($Review->CmpltDate)) .
"</font></i>";
echo '<!--' . $Review->InvNmbr . ' -->';
echo '<dt><span class="stars">'. $Review->avg .'</span>' . $Comments .'</dt>';
echo "</div>\r\n";
$entry_count++;
echo "<p style=\"font-size:9px\">Check out even more <a href=\"MorePraise.php\">reviews</a>!</p>";
                }
if ($entry_count < 1) {
        echo '<dt>No results to display.</dt>';
}


                ?>
```

The lines that start with $Comments are actually used to filter comments; we strip out email addresses and telephone numbers so those are not displayed. The echo lines are what actually create each display box. Provided you've followed along, you should now have a great result that looks something like:

Configuring for a "Show More" Display Under Method-A

You can use essentially the same code, but with a little modification, to create a larger page with "more results" (see above highlighted in yellow). Here's the modified code block we use on our example page to display these results in a table form, with alternating background colors. We've set our $NumberToDisplay higher in our variables (50 in our demo site), and then changed our foreach block as seen below.

```
<?php
        foreach ($SurveyResults->Review as $Review) {

                if ($entry_count % 2 == 0) {
                        $bgcolor = "#E3F1FF";
                }else{
                        $bgcolor = "#FFFAEB";
                }

                $Comments = $Review->CstmrCmnts;
                $Comments = preg_replace('/[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}/i','(phone hidden)',$Comments); // extract email
                $Comments = preg_replace('/(?:(?:\+?1\s*(?:[.-]\s*)?)?(?:\(\s*([2-9]1[02-9]|[2-9][02-8]1|[2-9][02-8][02-9])\s*\)|([2-9]1[02-
9]|[2-9][02-8]1|[2-9][02-8][02-9]))\s*(?:[.-]\s*)?)?([2-9]1[02-9]|[2-9][02-9]1|[2-9][02-9]{2})\s*(?:[.-]\s*)?([0-
9]{4})(?:\s*(?:#|x\.?|ext\.?|extension)\s*(\d+))?/','(email hidden)',$Comments); // extract phonenumber
                $Comments = stripslashes(html_entity_decode(str_replace("\r\n", " ", $Comments)));


                echo '<div id="content" class="more"><table width="500" cellspacing=0><tr>';
                echo '<!--' . $Review->InvNmbr . ' -->';
                echo "<tr><td bgcolor=\"" . $bgcolor . "\" valign=\"top\" align=\"left\" width=\"40\"><i><font style=\"size:9px;\">" . date( 'm-
d-Y', strtotime($Review->CmpltDate)) . "</font></i></td><td bgcolor=\"" . $bgcolor . "\" align=\"center\"><span class=\"stars\">". $Review->avg .
'</span></td></tr>';
                echo "<tr><td bgcolor=\"" . $bgcolor . "\" colspan=\"2\" align=\"center\">" . $Comments . '</td>';
```

```
                echo "</tr></table></div>\r\n";

                $entry_count++;
}
if ($entry_count < 1) {
                echo '<dt>No results to display.</dt>';
}
```

METHOD-B:
EMBEDDING INTO YOUR WEBPAGE VIA JAVA SCRIPT.

[This method was created by our Stephen Cox.]

This simple instruction set relies on javascript completely, if you don't have javascript it won't work!

The four components of the javascript implementation are relatively simple:

1. 1. Link to a style sheet that makes everything pretty:

   <link rel="stylesheet" type="text/css" href="http://my.rossware.net/lib/css/stars.css" />

   This will generally go in the top of your webpage document, usually within the <head></head> tags.

2. Link to the necessary javascript to load the kudos data:

   <script type="text/javascript" src="http://rossware.net/api/kudos/script/script.js"></script>

   This will also go in the top of your page

3. Using your existing page, you will want to place a div somewhere on your page with an ID you can reference later. This can be any element on your page, the innerHTML will be replaced with the kudos data:

   This can go anywhere in your page, typically within your <body></body> tags.

4. Implement the code that loads your kudos:

   ```
   <script type="text/javascript">
   var clientID = 2075;
   var threshold = 9.5;
   var quantity = 5;
   var elementID = 'kudos';
   var linkToMore
   ='MorePraise.php\\';loadKudos(clientID,threshold,quantity,elementID,linkToMore);
   </script>
   ```

   Which can, of course, be simplified into:

   ```
   <script type="text/javascript>"loadKudos('2075',9.5,5,'kudos','MorePraise.php\\');</script>
   ```

This will go anywhere in your page after the element with the ID you name (in this case 'kudos') has been created. Change to linkToMore = null; if you don't have a webpage with additional reviews.

If the generic version is not what you want, you are completely free to download the two loaded files and change them in any way you wish. These are:

http://my.rossware.net/lib/css/stars.css and
http://rossware.net/api/kudos/script/script.js

Just change the href in the css/stylesheet link to your new file, and change the src in the javascript tag to your new file!

Here is a direct example:

```
<link rel="stylesheet" type="text/css" href="http://my.rossware.net/lib/css/stars.css" />
<script type="text/javascript" src="http://rossware.net/api/kudos/script/script.js"></script>
<div id="kudos"></div>

<script type="text/javascript">
        var clientID = 2075;
        var threshold = 9.5;
        var quantity = 5;
        var elementID = 'kudos';
        var linkToMore = 'MorePraise.php\\';
        loadKudos(clientID,threshold,quantity,elementID,linkToMore);
</script>
```

To change the above from a “small view” to a “large view” request (i.e., with the latter being more appropriate for a deliberately requested “show more” kind of scenario), simply change the “var quantity” argument to something greater than 5 (typically, perhaps, 50 or so).

Appendix: XML Result Variables

$Review->ItemID -- This is the Internal Use ItemID for our SurveySystem. There's no need to show this to a customer, ever.

$Review->TechCode – This is the tech code for the tech who completed the job.

$Review->CallTaker – Person who took the call. XX would refer to one of the utilities.

$Review->InvNmbr – InvoiceNumber associated with this job

$Review->Mfg – Appliance manufacturer

$Review->InviteDate – Date the survey was setn

$Review->q1 – Q1 through Q4 are the actual results of the survey question

$Review->q2

$Review->q3

$Review->q4

$Review->avg – The average of the above 4 answers.

$Review->CstmrCmnts – Comments as left by the client. Make sure you filter these as they may contain phone numbers or other personal info.

$Review->CmpltDate – Date the survey was completed

$Review->IsCmplt – Has the survey been completed? 1 for yes, 0 for no.