APPLICATION NOTE SERIES

Information Technology Group
Computer Software & Systems
U.S Coast Guard Auxiliary

WOW II Pushbutton Website Platform: Smart Tags Rev 1.1

NOTE: This Application Note described an ADVANCED WOW feature and is intended primarily for Department and District webmasters. Except as noted, however, the capabilities described are available to all WOW sites. Fluency in HTML and software coding, although not required, are highly recommended.

What are Smart Tags?

"Smart Tags" in a WOW website are pieces of text that you can put into your Web page, on any announcement (post), that are replaced when the page is displayed with "dynamic" information.

For example let's say that you wish to create a custom page to describe your flotilla's Vessel Safety Check program, and at the bottom you wish to write:

To schedule a complimentary vessel safety check by Flotilla 114-12-04 at your boat, contact:

Tom W. Smith, FSO-VE
Cell: 650-555-1212
Email: tom.smith@cgauxnet.us

Example #1 – Static Content That Must Be Maintained

All WOW webmasters could create this in a heartbeat. But should they?

The problem with this is that the underlying information is "dynamic"; that is, it changes. It often changes when there is a change of watch, but frequently in between times, as well, due to the volunteer nature of our forces. And it is *guaranteed* to change when the CS officer is on a six-week vacation in the Uganda rain forest.

Now, WOW users know that content such as your staff list is dynamically updated by WOW from the current AuxOfficer directory, so you never have to worry about changing *that* content. Well, wouldn't it be nice if the above contact information could be written to update itself, as well?

It can. You would simply write, in place of the above text, the following:

To schedule a complimentary vessel safety check by Flotilla **{$_canonical_unit}** at your boat, contact:

{$_unit_officers.FSO_VE.fullname}, FSO-VE
Cell: **{$_unit_officers.FSO_VE.cphone}**
Email: **{$_unit_officers.FSO_VE.email}**

Example #2 – Dynamic Content with WOW Smart Tags

The `{$_unit_officers.FSO_VE.fullname}`, `{$_unit_officers.FSO_VE.cphone}`, and `{$_unit_officers.FSO_VE.email}` constructs, shown bold above, are called "smart tags". WOW replaces them with the correct information, in real time.

Let's deconstruct the first two tags; the others then will be obvious:

1. {…} sets off (encloses) WOW smart tags. *There must absolutely be no other curly brackets anywhere in your Web page's text or HTML* (see footnote).
2. `{$_canonical_unit}` is replaced by WOW with your unit's canonical unit number, here 114-12-04.
3. `{$_unit_officers.FSO_VE.fullname}` is replaced with the full name of the FSO-VE of your unit. Note that in the tag, the dash "-" is replaced with an underscore "_".

This latter tag is a species know as a "list" (or array). The actual tag is "$_unit_officers, which is a list of *all* relevant information about *all* officers in your unit. However, you don't want a list; you want a specific *officer* and a specific *fact* about that officer, in this case his full name.

To drill down to this specific fact, you simply add a dot "." after the tag name, and list the office you want, and a second dot followed by the *identifier* for the fact that you want. In this case, "fullname" is the identifier for the officer's full name, which WOW looks up in AuxOfficer, just before displaying the page.

Here are some examples of pulling data out of lists:

`{$_unit_officers.FSO_CS.hphone}`	The home phone number of the FSO-CS
`{$_unit_officers.VFC.lname}`	The last name of the Vice Flotilla Commander
`{_$dso_list.DSO_PE.085.fullname}`	The full name of the DSO-PE from District 085

A complete list of smart tags, including lists and available "fields" (facts), is given in Appendix I.

Control Tags

The observant reader will realize that although Example #1 *looks* dynamic, it is not. What happens if your FSO-VE resigns, and nobody replaces her (or him, in this example). What if the *next* FSO-VE doesn't have a cell phone? To be truly maintenance-free, your content will have to be able to modify itself as conditions change. This is what control smart tags are for.

There are *two* type of control tags supported by WOW: "if" tags, and "foreach" tags. "If" tags allow you to display content *if* something is true, and (optionally), alternative content if it is not. "Foreach" tags allow you to repeat content "for each" item in a list, while extracting information about each item in turn for display.

If Tags

The "FSO-VE-who-has-resigned-and-there-is-no-replacement" case is handled easily with "if" tags. Let's say you decide that if the FSO-VE resigns, you'd like to have the VFC listed in her place. Moreover, if there is no VFC, then the Flotilla Commander (or interim flotilla commander, when elected), represents your final fallback position.

Example #3 shows you what to put into your announcement. The basic structure is:

```
{if <something true>} <content if true>
{elseif <something true>} <content if true>
{else} <content otherwise>
{/if}
```

The {elseif} and {else} tags are optional. (The "< and >" are not used; they are show here to set off the explanations.) The closing {/if} is always required. Using the "if" tags, you could have content that displays *if* something is true, and *have nothing display if it is not.*

```
To schedule a complimentary vessel safety check by Flotilla
{$_canonical_unit} at your boat, contact:

     {if $_unit_officers.FSO_VE}
          {$_unit_officers.FSO_VE.fullname}, FSO-VE
          Cell: {$_unit_officers.FSO_VE.cphone}
          Email: {$_unit_officers.FSO_VE.email}
     {elseif $_unit_officers.VFC}
          {$_unit_officers.VFC.fullname}, Vice Flotilla Commander
          Cell: {$_unit_officers.VFC.cphone}
          Email: {$_unit_officers.VFC.email}
     {else $_unit_officers.FC}
          {$_unit_officers.FC.fullname}, Flotilla Commander
          Cell: {$_unit_officers.FC.cphone}
          Email: {$_unit_officers.FC.email}
     {/if}
```

Example #3 – Controlling Content Display with "If" Tags

For example, to display a block of content if the person who is logged on to the Member Zone is actually an auxiliarist (as opposed to active duty or authorized contractors), write:

```
{if $_is_member} Sign up for member training this week.{/if}
```

The "$_is_member smart tag is TRUE if someone is logged on *and* is an Auxiliary member, and FALSE otherwise.

The "true conditions" in Example #3 are a little subtle, and bear some explanation. If you write:

```
{if $_unit_officers.FSO_VE}
```

the tag is actually interpreted "if in the list of unit officers there is actually an FSO-VE, then 'TRUE'". This interpretation is a wonderful characteristic of lists in WOW (well, if you are a geeky programmer…); in an "if" tag, if a member of the list is actually *missing*, then WOW reads the test as FALSE, and if the list item is *not missing* (and actually has some data associated with it), then WOW reads it as TRUE. In this case, the list is "$_unit_officers", and the specific *member*s of the list are items such as "$_unit_officer.FC, $_unit_officer.VFC,… $_unit_officer.FSO_SR", etc.

```
{if $_unit_officers.FSO_VE}
     {$_unit_officers.FSO_VE.fullname}, FSO-VE
          {if $_unit_officers.FSO_VE.cphone}
               Cell: $_unit_officers.FSO_VE.cphone}
          {else if $_unit_officers.FSO_VE.hphone}
               Home: $_unit_officers.FSO_VE.hphone}
          {elseif $_unit_officers.FSO_VE.wphone}
               Work: $_unit_officers.FSO_VE.hphone}
          {/if}
          {$_unit_officers.FSO_VE.email}
               Email: {$_unit_officers.FSO_VE.email}
          {/if}
{elseif $_unit_officers.VFC} …
     <more of the same>
{/if}
```

Example #4 – Making Phone Numbers and Email Dynamic

Note that each of the above list items, such as $_unit_officer.FSO_SR, are themselves lists, and contain *facts* (or fields) such as fname, lname, cphone, hphone, member_id (see PII warning), and so forth. So, if you are really maniacal (as we are), you would treat items like home, work, and cell phone numbers as dynamic content, as well. In fact, we would have never written the code in Example #3, but instead, would have written (for each of the officers) the code shown in Example 4. *This* code is truly dynamic. It displays the first phone number that exists for the officer, in order from cell to home to work – or else omits the phone number entirely – and displays the word "Email" only if there is an email address to be rendered. And obviously, the same code would be repeated for the VFC and FC from Example 3.

In fact, there is only one case where this would all fall apart. We leave it to you to figure out, and ask, "Why, then was this person appointed to this position?"

As a final note, Example 4 demonstrates that {if}…{/if} tags can be "nested" inside one another, like Russian Matryoshka dolls. There is no limit to the nesting. Indenting the code as was done in these examples makes the structure more obvious, and is a best practice.

Foreach Tags

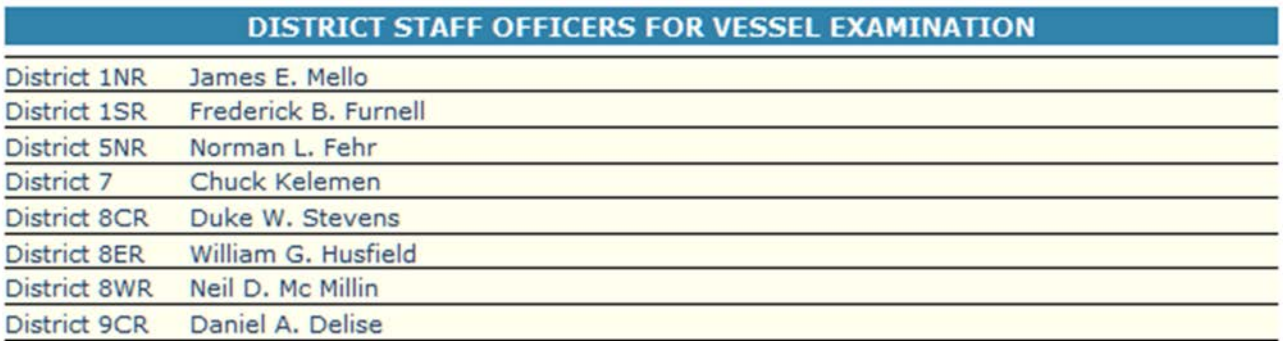
Often, you want to present a list. Two lists are built into WOW: a list of officers in the unit (flotillas, divisions, districts, departments, and groups), and a list of DSOs in the entire organization, organized by office (CS, IS, OP, etc.) and then by District.

The "Staff" page which is automatically added to all WOW sites creates a list of staff officers, so the staff list smart tags are more useful as sources of information about individual offices, as in the above example. However, many department websites show a list of DSOs around the country that service their discipline, and for these sites being able to print out a dynamic list is very useful.

You can see an example of such a list on the V-Department's WOW site, at:

http://wow.uscgaux.info/content.php?unit=V-DEPT&category=FIND-YOUR-DSOS

It is worthwhile visiting this page before continuing here, to get some context. The page in question consists of three (3) announcements: one containing the map, once containing a list of DSO-VEs for the country, and one containing a list of DSO-PVs. Try logging on, and note that the display changes to include email and phone information about each officer.



DISTRICT STAFF OFFICERS FOR VESSEL EXAMINATION	
District 1NR	James E. Mello
District 1SR	Frederick B. Furnell
District 5NR	Norman L. Fehr
District 7	Chuck Kelemen
District 8CR	Duke W. Stevens
District 8ER	William G. Husfield
District 8WR	Neil D. Mc Millin
District 9CR	Daniel A. Delise

Figure 1 – Snippet from V-Department WOW Site Showing Part of National DSO List

The formatting of the two lists was tricky, and is not a subject of this Application Note. However, the dynamic lists were each produced with a single pair of "foreach" tags, and some "if" tags, used as demonstrated above, to conditionally display content if it was actually available.

The foreach tag works like this: if you have a list, you can "loop" through the list and write a single block of code that gets repeatedly applied to each item in the list. So, if, for example, I have a list of all DSO-VEs, WOW can loop through them one at a time, and use the same WOW content with smart tags to display each one.

Here's the basic structure: if the smart tag "$_dso_list" is a list of all DSOs in the country, organized by office, then the smart tag "$_dso_list.VE" is a list of all DSOs-VE, organized by District.

The foreach tag is written:

```
{foreach $_dso_list.VE as $dso}
… <content that is repeated each row>
{/foreach}
```

Anything you put between `{foreach…}` and `{/foreach}` will be displayed repeatedly, as the list of DSO-VEs is stepped through (or "looped over"), district by district. And the magic is that the foreach creates a *new* smart tag on the fly representing the list of information available about each DSO. That's what the "…`as $dso`" is all about. The $dso is a smart tag – named as you see fit – that, each time the loop is repeated, takes the value of that item in the list – in this case, a list of facts about each DSO by district. This special tag we'll call the "list element".

So, *inside* the foreach "loop", we can write:

```
{$dso.fname} to get the DSO's first name
{$dso.unit} to get the DSO's unit (district)
{$dso.email} for the DSO' email address, etc.
```

Note that if we had written the foreach loop as `{foreach $dso_list as $greeble}`, then to pull out the email address for each DSO, we would write:

```
{$greeble.email}
```

which would have produced the *identical* output as the third statement in the preceding example.

Putting this all together, Example #5 shows what you would put into the WOW announcement to render this entire list. All formatting has been omitted. The tag "`$dso@last`" becomes TRUE only if the foreach is executing the *last* item in the list. Compare this to what you see on the screen, at the above link, and it all will become clear!

The formatting of all of this into the columns shown on the actual WOW page is done with judicious (and advanced) use of WOW's Cute editor, or by simply

```
{foreach $_dso_list.VE as $dso}
  District {$dso.district} {$dso.fullname}
  {if $_logged_on}
    {$dso.email}
  {/if}
  {if $_logged_on}
    {if $dso.cphone}C: {$dso.cphone}{/if}
    {if $dso.hphone}H: {$dso.hphone}{/if}
    {if $dso.wphone}W: {$dso.wphone}{/if}
  {/if}
  {if $dso@last}
  <put in extra blank lines after the last
  iteration>
  {/if}
{/foreach}
```

Example #5 – Using Foreach to Output a List of DSOs-VE (Formatting Omitted)

switching to HTML mode (if you know HTML) and formatting manually. For the sake of completeness, the HTML that makes up the VE list on that page is shown in its entirety in Appendix #2.

Using Smart Tags in Practice

To create a page using smart tags, you must proceed as follows:

1. Create an announcement on the desired page, in the HTML mode, that consists *solely* of the following XML tag:

   ```
   <wow:smart_tag>
   ```

   This announcement should be captioned “Smart Tag Flag”, and should be Order 0. That is, it should be set to be the *first* announcement on the page. This tells WOW to look for smart tags in all announcement on the page.
2. Create and format your entire page using *real* data (that is real or dummy names, email addresses, etc.) and tweak the page formatting until it displays exactly as you desire. Note that if you are outputting any lists, you only need to dummy up one “row” of that list (e.g., one DSO in the above example). If you have any “conditional” content, include it.
3. Now, put “if” tags around any conditional content, and code in the control tag that you need to test for “true”, such as `{if $_logged_on}` … `{/if}`. NB: if you want to test for FALSE, simply put an exclamation point in front of the tag, as in `{if ! $_logged_on}`. This is read “if NOT logged on”.
4. Finally, put {foreach…} … {/foreach} tags around any content that must repeat (usually in rows), and choose a name for the “list element” tag you will use inside your foreach loop (such as $dso in the above examples)
5. Now, go through your entire page, and *replace* the dummy “real” data with the smart tags that will render the same information dynamically.

   For example, if you dummied up the following (dummy material in **bold**):

   ```
   District 017 DSO-CS Dobie Gillis  dobie@paradise.island.com
   ```

   then you would carefully swap out the dummy data for smart tags as follows:

   ```
   District {$dso.unit} DSO-CS {$dso.fullname}  {$dso.$email}
   ```
6. SAVE your announcement, and “Return to Unit Page” to test the page.

Coding Errors

When you are writing content with smart tags, you are writing code. If you make a single mistake, such as leaving out a curly bracket, forgetting the $-sign that starts the tag name, omitting the “closing” {/if} for any {if} tag, or the closing {/foreach} to match an opening {foreach…}, you will get a *voluminous, intimidating* error message.

Don’t be frightened by it. It simply means you have a typo in one of your smart tags. Look at the first part of the error message for a clue like “No closing {/if}” and then use that insight while scrutinizing your code. You’ll eventually find the character you mistyped or left out, and suddenly your page will render in all its glory.

Other Best Practices

Making "Legacy" Pages Dynamic

If you are intending to make dynamic a page that was originally created in HTML using a tool like FrontPage or Expression Web, and you imported the HTML directly into WOW when you ditched your AIRS or other conventional site, you may have legacy HTML code in there that contains curly brackets, and it will clobber the smart tag engine.

The way to tell is to simply follow step 1, above, putting in the Smart Tag Flag in the first announcement, and then "Return to Unit Page" to see what happens. If you get a ghastly error message, you'll have to rebuild the page using the trick of cleaning out all HTML (use the "broom" icon in the Cute Editor, and select the "Remove all HTML tags" option), and then formatting your page from scratch. That's OK, you shouldn't have copied that ancient HTML over in the first place because your page, frankly, probably looks like a ransom note.

Tables and Foreach Loops

Because of a bug in the licensed Cute Editor used in WOW, you cannot create a table, and then use a foreach tag set to control the contents of each row, even though this would be a best practice. The Cute Editor inexplicably rewrites the code and renders the foreach tag useless. This bug was reported to the software vendor. You will have to use nested <div> blocks, as is shown in Appendix 2, which is what all the cool programmers are doing these days, anyways ☺.

Meta Tags in Foreach Loops

You saw the example in the foreach tag section of "testing" for the last iteration of the foreach loop, using the list element tag plus "@last" (e.g., `{if $dso@last}`).

You might want to use this, for example, to draw a thin line between each row of your output, but a *thick* line before the first and after the last iteration.

Here it is in HTML:

```
{foreach $_unit_officers as $officer}
     {if $officer@first}
          <div style=”border-top:2px solid black;width=”100%”>
     {else}
          <div style=”border-top:1px solid black;width=”100%”>
     {/if}
     </div>
     <!–Content here for this row -->
     {if $officer@last}
          <div style=”border-bottom:2px solid black;width=”100%”>
     {else}
          <div style=”border-bottom:1px solid black;width=”100%”>
     {/if}</div>
{/foreach}
```

There are other such "meta" tags available, such as the current iteration number. These will be documented in a later version of this Application Note.

Personally Identifiable Information (PII)

WOW smart tags include virtually all the information about a member that can be accessed in AuxOfficer (AuxDirectory), for the *member who is logged on.* You can use these tags to "personalize" a page (e.g., Hello, Bill. Welcome back!), or to pre-fill a registration form, etc. Since these tags are NULL (empty) when the member is not logged on, be sure any such personalization code checks the $_if_logged_on tag, so you don't get awkward, broken content like: "Hello, . Welcome back!".

The logged-on-member tags do not present a problem with Personally Identifiable Information, since the member is seeing his or her own information, and only when logged on. However, there is PII available on other members in the DSO lists, and in the unit officer lists.

Without the specific permission of any member, you may *not* display the following items on any page unless it is protected in the Member Zone, or the content is revealed using smart tags only if the site visitor is logged on as a member: email address, telephone numbers, home address, spouse's name, and member ID.

Support

No technical support will be provided for the use of WOW Smart Tags. If you have a real need to use them, and are struggling, find someone in your unit or division who is or was a programmer, and ask for help. In contrast, if you have been successful using them, then provide a writeup to the DVC-UC or DVC-CN, and we'll make it an Application Note.

Change Log

Revision 1.1 – 5 August 2012: Added Address1, Address2, City, State, Zipcode to logged-on member smart tags.

Appendix 1 – WOW Smart Tag List

Tags About The Person Logged On

$_member_fname	First Name
$_member_middle	Middle Name or Initial
$_member_lname	Last Name
$_member_name	Full Name (Title Case)
$_member_email	Email Address (email1)
$_member_dist	District
$_member_div	Division
$_member_flot	Flotilla
$_member_hphone	Home Phone
$_member_cphone	Cell Phone
$_member_wphone	Work Phone
$_member_addr1	Address Line 1
$_member_addr2	Address Line 2
$_member_city	City
$_member_state	State
$_member_zipcode	ZipCode
$_member_id	Member ID
$_member_offices	Member's Offices
$_member_quals	Qualifications List
$_status	Member Status (BQ, AX, etc.)
$_is_member	True or False

Tags About the Site's Unit/Department

$page_title	Title of the Current Page
$category	Page Category or Slug
$_unit_level	DIST, DIV, FLOT, DEPT, GROUP
$_canonical_unit	E.g., 114-12-04
$_fullname	Full Name of Unit
$_formalname	Formal Name of Unit
$_unit_city	Unit City
$_unit_state	Unit State
$_service_area	Unit Service Area for WOW
$_unit_officers	Unit Officers (list)
$_is_department	True if Department or Group

Global Tags (Department/Group Sites Only)

$_dso_list	All Auxiliary DSOs (list)

Unit Officers List Breakdown

Structure: $_unit_officers.<office>

Examples: $_unit_officers.FSO_VE
$_unit_officers.SO_OP
$_unit_officers.DSO_CS

Underscore replaces dash in office codes.

Each of the above is a list itself, structured as follows:

$_unit_officers.XXX_YY.<fact>

Examples: $_unit_officers.DSO_HR.fullname
$_unit_officers.SO_OP.email1
$_unit_officers.FSO_MT.cphone

Allowable values for the <fact> fields are shown on the following page.

Website Meta Tags

$META_title	Page Title from Browser Perspective
$META_description	Unit Specific Generic Page Description
$META_abstract	Unit Specific Generic Page Abstract
$META_unit_name	Unit Name, as in "Flotilla 12-4, District 11SR"
$META_author	Same as Unit Name
$META_webmaster_name	Not Used
$META_webmaster_email	Usually "NOT AVAILABLE"
$META_revised_date	Always today's date

Meta tags hold the HTML meta tags sent to the browser in the <head> area. They can be seen on any WOW page by telling your browser to "View Source".

DSO List Breakdown

Structure: $_dso_list.<office>

Examples: $_dso_list.VE
$_dso_list.OP
$_dso_list.MT

Each of the above is a list itself, structured as follows:

$_dso_list.XX.<district>

Examples: $_dso_list.VE.014
$_dso_list.VE.081
$_dso_list.VE.140

Each of the above is a list itself, structured as follows:

$_dso_list.XX.nnn.<fact>

Examples: $_dso_list.VE.014.fullname
$_dso_list.VE. 014.email1
$_dso_list.VE. 014.cphone

Values for <fact> are shown to the right.

DSO List/Unit Officer List Facts (Fields)

unit	Canonical Unit Number (e.g. 081-25-03)
level	NATL,DIST,DIV or FLOT
odr	Sort order (low to high)
office	DSO-CS, DVC-CH, etc.
memberID	7-digit member ID
email1	Primary email address
email2	Secondary email address
fname	Officers' first name
middle	Officer's middle initial/name
lname	Officer's last name
reportlvl	Level office reports to (e.g., DIST)
reportoff	Office code of supervisor (e.g. DCOS)
descrip	Office Description (e.g. Operations)
hphone	Officer's home phone number
cphone	Officer's cell phone number
wphone	Officer's work phone number
fullname	Officers Full Name (First Last)

Opt In/Opt Out Flags from AuxOfficer:

disptel	Display phone number(s) True/False
dispemail	Display email True/False
celltext	Officer can receive text on cellphone
necodes	Reserved
dispcodes	Reserved

Appendix 2 – Complete HTML Example of Department Site DSO List With Smart Tags

The results of this code may be seen at :

http://wow.uscgaux.info/content.php?unit=V-DEPT&category=find-your-dsos

Smart tags are shown in **red**.

```
<div align="center"><span style="width: 595px; font-weight: bold;"
class="turquoise">DISTRICT STAFF OFFICERS FOR VESSEL EXAMINATION</span></div>
   <div style="margin-top: 5px; background-color: #ffffee;">
   {foreach $_dso_list.VE as $dso}
      <div style="clear: right; border-top-color: black; border-top-width:
      1px;border-top-style: solid;">
      <div style="width: 83px; display: inline-block;font-size:11px;">District
      {$dso.district}</div>
      <div style="width: 150px; display: inline-block;font-
      size:11px;">{$dso.fullname}</div>
      {if $_logged_on}
         <div style="width: 225px; overflow: hidden; display: inline-block;font-
         size:11px;" >
         <a style="font-size:11px;"
         href="mailto:{$dso.email1}">{$dso.email1}</a></div>
      {/if}
      {if $_logged_on}
         <!-- Phone Block -->
         <div style="width: 125px; float: right; display: inline-block;">
         {if $dso.cphone}
            <div style="float: right; display: inline-block;font-size:11px;" >
            C: {$dso.cphone}</div>
         {/if}
         {if $dso.hphone}
            <div style="float: right; display: inline-block;font-size:11px;" >
            H: {$dso.hphone}</div>
         {/if}
         {if $dso.wphone}
            <div style="float: right; display: inline-block;font-size:11px;" >
            W: {$dso.wphone}</div>
         {/if}
         </div>
         <!-- end of phone block -->
      {/if}
      </div>
      {if $dso@last}
         <div style="clear: right; border-top-color: black; border-top-width: 1px;
         border-top-style: solid;">&nbsp;</div>
      {/if}
   {/foreach}
</div>
```

####

Revision 1.0 July 23, 2012