Whitepaper

Going Above and Beyond

Using Advanced Techniques to Create Customized HTML Templates

August 3, 2010

Information in this document is subject to change without notice. Companies, names, and data used for example herein are fictitious unless otherwise noted. Some screen captures have been cropped and/or edited for emphasis or descriptive purposes.

L-Soft invites comments on its documentation. Please feel free to send your comments by email to: manuals@lsoft.com

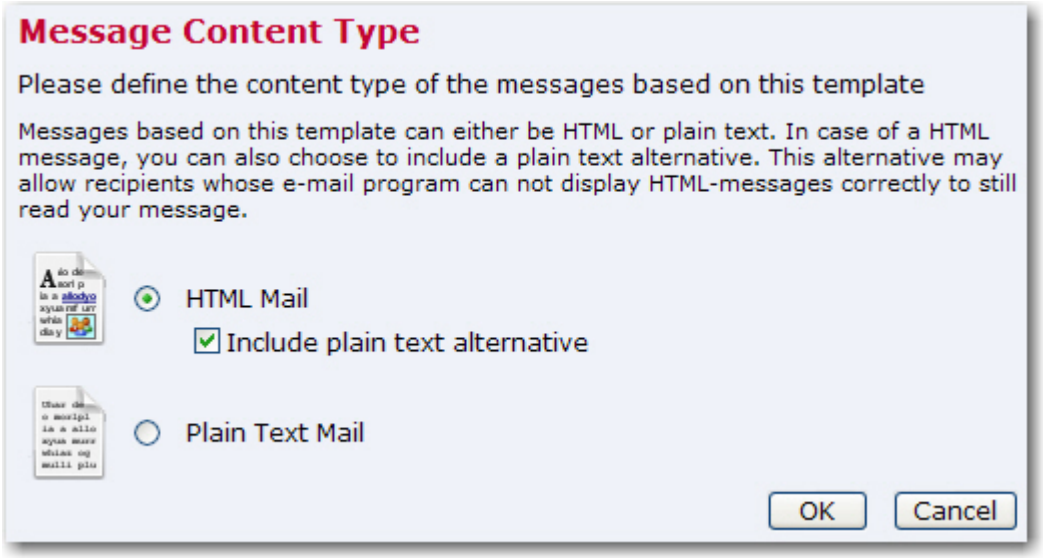
Introduction

For those more adventurous content managers who are familiar with HTML, then LISTSERV Maestro gives you the ability to expand your limits and design professional-looking HTML messages that go above and beyond the everyday standards. The advanced techniques explained in this document will give you the powers to not only enlighten but to astonish your subscribers; drawing them in and making their experience even more remarkable.

Tip: To learn more about the basic template concepts, such as how they work, giving users the right to create templates, and how to use templates to define a message, then you'll find clear, easy-to-follow instructions in the Message Templates: Getting Started whitepaper entitled *No HTML Coding Experience Necessary! Creating Professional Looking HTML Message without Coding* and the Message Templates: Creating Your Own Templates whitepaper entitled *Do-It-Yourself Templates: Using Your Own Content to Create Message Templates*.

Creating HTML Templates

To create an HTML template, click the **Utility** menu, select **New Content Templates**, and then select **Create Empty Template**. The Edit Content Template screen opens. Click on the **HTML** icon. The Message Content Type screen opens. Select the **HTML Mail** option, and then check the **Include plain text alternative** box. This ensures that recipients whose email client does not display HTML will default to the plain text alternative, which allows them to receive the message.



Click **[OK]** to save and return to the Edit Content Template screen. You are now ready to define your HTML template. The first step is to enable drop-in content. See the next section for details.

Enabling Drop-In Content

To use drop-in content in your template, you first need to enable it. Drop-in content is disabled by default for a newly created template. To enable drop-in content, click on the **Drop-Ins** icon in the icon bar on the Edit Content Template screen. The Placeholder Opening and Closing Tags screen opens.

On this screen, click the **Enable drop-in content** option, and then define the opening and closing tags (see below). Click **[OK]** to save your settings and return to the Edit Content Template screen.

Opening and Closing tags are used by LISTSERV Maestro to identify the names of the drop-in content. In content templates, the opening and closing tags used for drop-ins must be the same as used for the template placeholders. Therefore, changing the tags for drop-ins also changes the tags for the template placeholders and vice versa. Opening and closing tags can be any characters you like that do not contain any spaces. "{{" and "}}" are suggested as defaults, but you may change them.

To include drop-in content in your template, simply enter the name of the drop-in at the appropriate location, enclosed in the tags you have defined. For example, if your drop-in is called "sample" and you are using the suggested default tags, then you would include the text "{{sample}}" in the location where you want the drop-in content to appear in the final message. Whenever LISTSERV Maestro finds the opening tag followed by the closing tag, it will interpret all the characters in between first as a template placeholder name. If no placeholder with this name is found, then the name is instead interpreted as the name of a drop-in. LISTSERV Maestro will then look up the list of available drop-ins for a drop-in with this name. If the name is not found, delivery of the mail fails with an error message like "Drop-In with name 'sample' not found".

If drop-in content is disabled, any drop-in placeholder names that might appear in the message will not be replaced with the corresponding content. Instead, the placeholder names will appear verbatim in the body of the message.

For more information on drop-ins, see the *LISTSERV Maestro User's Manual*. You are now ready to define the template's content. See the next section for details.

Defining the HTML Template Content

The Edit Content Template screen lets you define the content for your HTML template.

The **Subject field** is the subject line of the message. It is mandatory and must be filled out.

The **HTML Preview tab** is where you can upload or download the HTML body of your message. Once defined, it also displays the HTML body of the message, with all drop-in elements resolved. If errors have been encountered while LISTSERV Maestro has tried to replace the drop-in placeholders with the drop-in content, then an error message is displayed and the un-replaced drop-in placeholders are highlighted in the HTML preview.

To define the body from this tab, you can upload an HTML file. An uploaded HTML file may contain links to images on a remote server or local (inline) images embedded in the page. To upload an HTML page, click the **[Upload HTML]** button. This will start a small applet that allows

you to select an HTML file from your local disk for upload to the system. You will need to have Java enabled in your browser for the applet to work. Since the applet needs to access your local disk, you will need to grant rights for the applet to function correctly. The applet will load the selected HTML file and will search for all images and other binaries that are referenced in the HTML code. All binaries that are referenced locally (with a local file name and/or path) will be uploaded together with the HTML code.

Once the upload is complete, the applet displays a short summary, listing all locally referenced binaries that were uploaded, as well as all binaries that were referenced as links to files on the Internet (these are not uploaded). All binary references that are found, but could not be resolved, will also be listed.

Confirming the summary will allow the applet to upload the data (HTML and binaries) to the system. The system will display the uploaded page on the HTML tab.

By clicking the **[Download HTML]** button, a previously uploaded HTML file and its associated binaries can be downloaded to your computer in the form of a ZIP archive. This also applies to any HTML code that has been entered directly into the box on the HTML Code tab.

The HTML page is mandatory and must be filled out, either by uploading an HTML page or by typing the HTML code directly into the box on the HTML Code tab.

The **HTML Code tab** contains the source code of the HTML body of the message. To define the body, type HTML code directly in the box, cut and paste it from another application, or upload an HTML page. To edit the source code, simply edit the code shown on this tab, and then go back to the HTML Preview tab to view the changes.

To upload an HTML page, click the **[Upload HTML]** button. To download a previously uploaded HTML file and associated binaries in form of a ZIP archive, click the **[Download HTML]** button.

The HTML page is mandatory and must be filled out, either by uploading a HTML page or by entering the HTML code directly on the HTML Code tab.

The **Text Preview tab** displays the plain text alternative option for the HTML mail. For this to function properly, the plain text alternative and drop-in content must be enabled for the job.

The alternative text is a plain text body that is an "alternative" for the more elaborate HTML body of the message. Recipients whose email client does not display HTML may default to the plain text alternative allowing them to receive a message. To reach the highest number of recipients, it is generally a good idea to use an alternative text with each HTML message.

This tab is a preview of the alternative plain text body of the HTML email message, with all drop-in elements resolved. If errors have been encountered while LISTSERV Maestro has tried to replace the drop-in placeholders with the drop-in content, an error message is displayed and the un-replaced drop-in placeholders are highlighted in the alternative text preview.

You cannot edit the alternative text on this tab (see the Text tab), but you can upload a new alternative text by clicking the **[Upload Text Alternative]** button.
The **Text tab** lets you define the alternative text of your choice. Type the text directly in the box, cut and paste it from another application, or upload a text file by clicking the **[Upload Text Alternative]** button.

For this to function properly, the plain text alternative and drop-in content must be enabled for the email job.

Defining the Template Settings

The Template Settings tab on the Edit Content Template screen is available for both plain text and HTML templates.

The **General Settings** section contains two options:

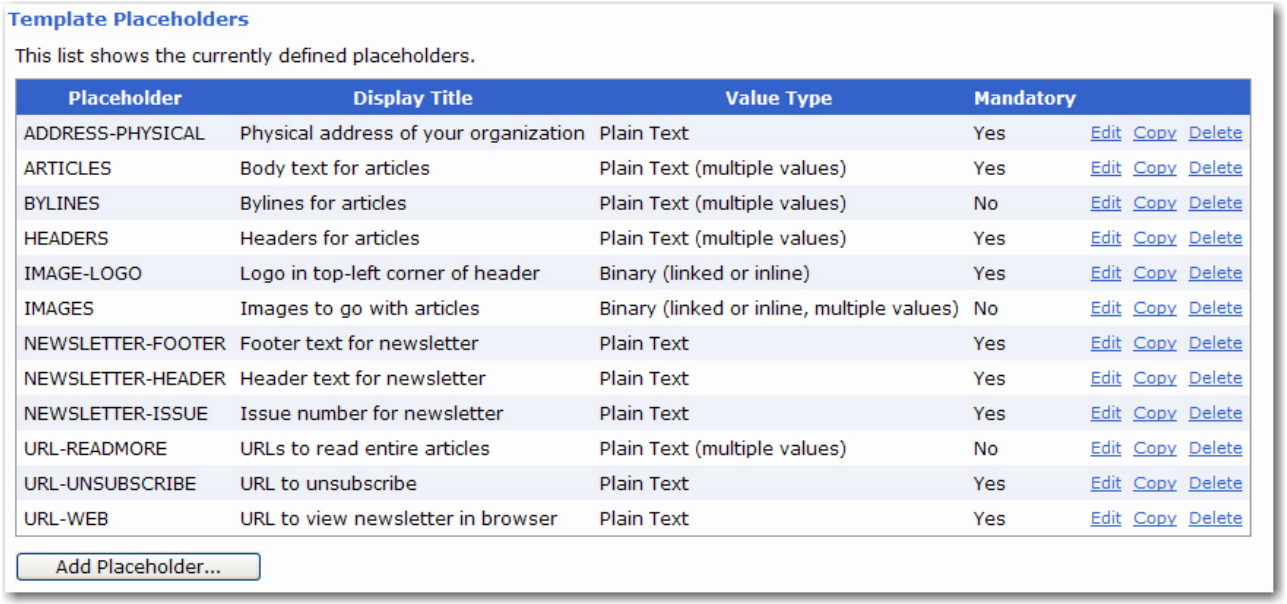
- **Allow Advanced Editing** – When a template is used during the content definition of a mail job, then the user who uses the template usually only has to "fill in the blanks" that are left by the template placeholders. The user does not have to bother with the actual message body, as this body is already defined by the template. Therefore, by default, when a template is used to define the message content, it is not even possible to modify the message body, except by filling out the placeholders.

  Under certain circumstances, it may be desirable to allow a user to not only fill out the placeholders but to also modify or edit the whole of the message body. This is what this option is for. If checked, then editing of the message body outside of the placeholders is allowed while the template is being used for content definition. If not checked, then only the placeholders can be filled out.

- **Highlight Placeholders** – When a template is used to define the message content, then the user will have to provide content for all placeholders in the template. To make this process simpler for the user, LISTSERV Maestro highlights the currently selected placeholder in the preview.

  To select the background color for this highlighting, click the **Change** link and select a color. Make sure to use a color that has a good contrast in comparison to the foreground color used in the template.

The **Template Placeholders** section contains a table of all placeholders that are currently defined in the template.

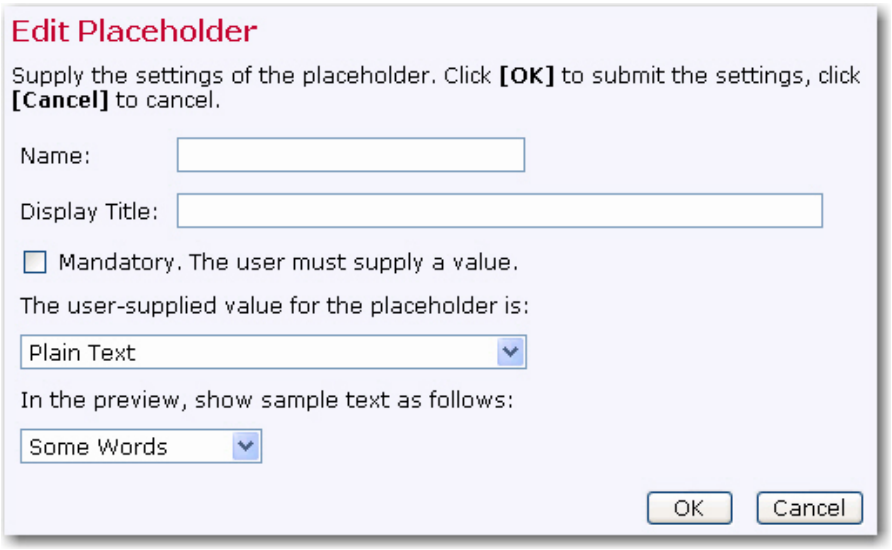


Placeholder	Display Title	Value Type	Mandatory	
ADDRESS-PHYSICAL	Physical address of your organization	Plain Text	Yes	Edit Copy Delete
ARTICLES	Body text for articles	Plain Text (multiple values)	Yes	Edit Copy Delete
BYLINES	Bylines for articles	Plain Text (multiple values)	No	Edit Copy Delete
HEADERS	Headers for articles	Plain Text (multiple values)	Yes	Edit Copy Delete
IMAGE-LOGO	Logo in top-left corner of header	Binary (linked or inline)	Yes	Edit Copy Delete
IMAGES	Images to go with articles	Binary (linked or inline, multiple values)	No	Edit Copy Delete
NEWSLETTER-FOOTER	Footer text for newsletter	Plain Text	Yes	Edit Copy Delete
NEWSLETTER-HEADER	Header text for newsletter	Plain Text	Yes	Edit Copy Delete
NEWSLETTER-ISSUE	Issue number for newsletter	Plain Text	Yes	Edit Copy Delete
URL-READMORE	URLs to read entire articles	Plain Text (multiple values)	No	Edit Copy Delete
URL-UNSUBSCRIBE	URL to unsubscribe	Plain Text	Yes	Edit Copy Delete
URL-WEB	URL to view newsletter in browser	Plain Text	Yes	Edit Copy Delete

Each placeholder is displayed with its name, display title, value type, mandatory/optional state, and the following actions:

- **Edit**: Click this link to edit this placeholder. The Edit Placeholder screen opens. From here, make changes to the placeholder and then press **[OK]** to save. For more information, see the *Editing Placeholders* section.
- **Copy**: Click this link to create a new placeholder as a copy of this one. A copy of this placeholder will appear in the list. Once the copy is created, you should rename the placeholder and make any other changes you'd like to make. To do this, click the **Edit** link. For more information on editing a placeholder, see the *Editing Placeholders* section.
- **Delete**: Click to delete this placeholder. You will be asked if you really want to delete the placeholder. Click **[OK]** to confirm.

The **[Add Placeholder...]** button at the bottom of the tab lets you create a new placeholder. Once you click on this button, the Edit Placeholder screen opens.



From this screen, you can enter the information for the new placeholder. For information on the Edit Placeholder screen, see the *Editing Placeholders* section. For other methods of creating a placeholder, see the *Creating Placeholders* section.

Using Template Placeholders

Template placeholders make content templates easy to use and truly customizable, as they define the "blanks" that the user will have to fill out. A template placeholder appears in the template body in the following form:

`{{#NAME}}`

where "{{" and "}}" are the opening and closing tags, and "NAME" is replaced with the name of the placeholder. The "#" character that prefixes the name must always appear for a template placeholder.

A placeholder with a given name can appear several times throughout the template body and may appear both in the HTML and the plain text part, if the placeholder's type allows this.

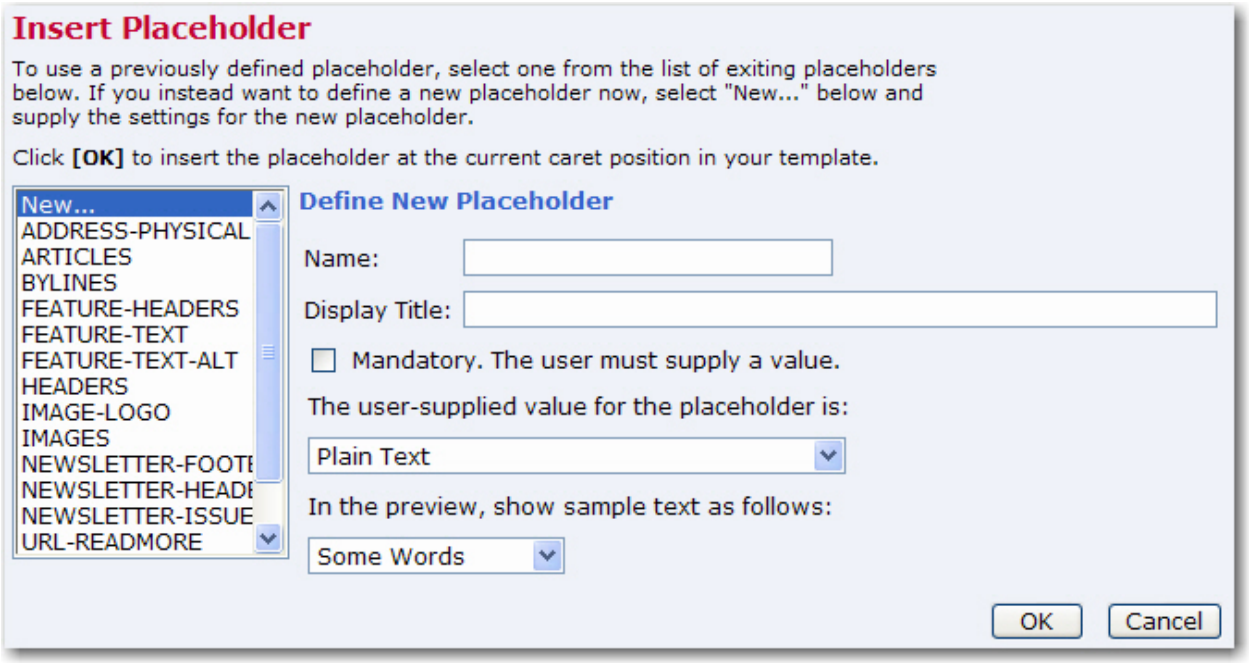
The opening and closing tags for the placeholders can be freely defined and can be any characters you like that do not contain any spaces. "{{" and "}}" are suggested as defaults, but you may change them by clicking the **Placeholders** icon in the icon bar.

Note: The opening and closing tags that are defined for the placeholders will also be used for drop-ins, if drop-ins are enabled for the template. It is important to choose opening and closing tags with care, making sure that they do not otherwise appear in the template. For example, if you decide to use parentheses '(' and ')' for opening and closing tags, any text that appears between any set of parentheses will be interpreted as the name of a template placeholder or drop-in. Since parentheses have a high probability for use in the normal text of a message, using them as tags is not recommended because they may cause an error in placeholder or drop-in name look up, leading to mail delivery failure.

Creating Placeholders

There are several ways to create a new placeholder:

- Click the **[Add Placeholder]** button on the Template Settings tab. For more information, see the *Defining the Template Settings* section.
- Click the **Copy** link associated with a placeholder on the Template Settings tab. For more information, see the *Defining the Template Settings* section.
- Click the **Click here to insert a placeholder** link at the top of the Text tab. This will open the Insert Placeholder screen, which allows you to create a new placeholder by selecting the **New** option in the left list and filling out the placeholder attributes in the right part. The `{{#NAME}}` tag of the new placeholder will be automatically inserted at the current cursor position in the current text input box.
- Press **CTRL+Space** while typing in the text box on the Text tab. This will also open the Insert Placeholder screen (see above).
- If you simply want to insert an already existing placeholder somewhere in the current text box, you can either simply type its `{{#NAME}}` tag or you can open the Insert Placeholder screen by clicking the **Click here to insert a placeholder** link or pressing **CTRL+Space**. In this screen, select an existing placeholder from the list on the left and click **[OK]**, which will automatically insert the placeholder's `{{#NAME}}` tag at the current cursor position.



Editing Placeholders

Placeholders have various attributes that determine their appearance and behavior. These attributes can be defined during placeholder creation or by clicking the **Edit** link of an existing placeholder in the list on the Template Settings tab:

- **Name** – The name of the placeholder. This is the name that you must use as a replacement for NAME in the {{#NAME}} tag. Only the following characters are allowed in a placeholder name: 'a'-'z', 'A'-'Z', '0'-'9', '_' and '-'.

- **Display Title** – This is the title or name of the placeholder that will be displayed to users that employ the template during content definition. Enter a title that makes it easy for the user to understand the purpose of the placeholder.

- **Mandatory** – Defines if the placeholder is mandatory or not. For a mandatory placeholder, the user must supply a non-empty content. For an optional placeholder, the user may decide to leave the placeholder empty.

- **Value Type** – Defines the placeholder's content type.

  The following **Single Value Placeholders** are available:

  - **Plain Text** – The placeholder's content will be interpreted literally, as plain text. If the placeholder is used in a HTML part, then any special HTML code characters in the content will be escaped before replacement. Therefore, it will not be possible to include any HTML formatting (like bold, color, font, etc.) in a placeholder of this type. Only the two plain text placeholder types can be used in a plain text template or in the alternative text of a HTML template.

  - **HTML** – The placeholder's content will be interpreted as HTML code. Therefore, if the content contains any HTML formatting code (HTML tags or attributes), then these will not be escaped and will have their normal formatting functionality. On the other hand, it is necessary for the user who employs this template during content definition to HTML-escape any special HTML characters that are not supposed to be interpreted as HTML code. Also, the user must remember not to enter any HTML tags that may disrupt the syntax or semantics of the surrounding HTML tags defined in the template. Placeholders of this type can only be used in the HTML part of a HTML template.

  - **Binary (linked or inline)** – This is a special placeholder that allows the template manager to define a binary (usually an image) in the HTML template of which the actual content is not yet known but will be supplied later by the user who employs the template during content definition.

    When doing so, the user who defines the content may decide if the binary data (the image file) is supposed to be linked remotely, from a URL at a web server, or if the binary data shall be bundled into the mail as an inline attachment.

    Binary placeholders can be used for images (JPEG, GIF, and PNG) or background sounds (MIDI).

    During content definition the placeholder will be replaced with a URL to the image (either a linked or inline URL), which is why this type of placeholder should be

used only in a context in the HTML code where such a URL makes sense. Normally, the placeholder would stand in for an image and would appear in the "`src`" attribute of an "`<img>`" tag, like this:

```
<img src="{{#SampleBinaryPlaceholder}}">
```

Other valid contexts would be in a style definition where an image URL is required or in a background image or sound attribute of an HTML tag.

This placeholder type can only be used in the HTML part of an HTML template.

- **Binary (inline)** – Same as the Binary (linked or inline) type above, only with this type, the binary will always be an inline binary. This means that the user who employs the template during content definition will have no choice about this.

- **Binary (linked)** – Same as the Binary (linked or inline) type above, only with this type, the binary will always be a linked binary, i.e. the user who employs the template during content definition will have no choice about this.

The following **Multiple Values Placeholders** are available:

- **Plain Text (multiple values)** – Same as the Plain Text type above, but the user who employs the template during content definition has the option of supplying multiple values for the placeholder. The placeholder will be replaced with the combined multiple values (linked to each other). This is useful when using the multi-placeholder repeater and/or the special placeholder prefix and suffix because the prefix and/or suffix will appear once before/after each of the multiple values before they are concatenated to build the combined replacement value.

  From the **values in preview** drop-down menu, select how many dummy values shall be displayed for this placeholder during preview (when no values have been supplied yet).

  Only the two plain text placeholder types (see also above) are allowed to be used in a plain text template or in the alternative text of a HTML template.

- **HTML (multiple values)** – Same as the HTML type above, but the user who employs the template during content definition has the option of supplying multiple values for the placeholder. The placeholder will be replaced with the combined multiple values (linked to each other). This is useful when using the multi-placeholder repeater and/or the special placeholder prefix and suffix because the prefix and/or suffix will appear once before/after each of the multiple values before they are concatenated to build the combined replacement value.

  From the **values in preview** drop-down menu, select how many dummy values shall be displayed for this placeholder during preview (when no values have been supplied yet).

  This placeholder type can only be used in the HTML part of an HTML template.

- **Binary (linked or inline, multiple values), Binary (linked, multiple values), Binary (inline, multiple values)** – Same as the corresponding Binary types above, but the user who employs the template during content definition has the

option of supplying multiple binaries for the placeholder. The placeholder will be replaced with a list of the URLs of all supplied binaries (both linked and inline). This is useful when using the multi-placeholder repeater and/or the special placeholder prefix and suffix because the prefix and/or suffix will appear once before/after each of the multiple URLs before they are linked to build the combined replacement value, which allows you to enclose each of the URLs with its own `<img>` tag to actually display the image that is represented by the binary.

From the **values in preview** drop-down menu, select how many dummy images shall be displayed for this placeholder during preview (when no binaries have been supplied yet).

This placeholder type can only be used in the HTML part of an HTML template.

- **Sample Preview** – There are several situations where the template is supposed to be displayed while the placeholders are not yet filled out with any content. In these situations, LISTSERV Maestro fills out the placeholders with dummy content as a stand-in for the actual content that is yet to be provided. This makes it easier to assess the overall appearance of the template while the placeholders are still empty. The kind of dummy content that is used for this can be defined by the content manager, depending on the type of the placeholder:

  - **Dummy Text** – For Plain Text and HTML placeholders there are several choices to roughly define which kind of dummy text shall be displayed. The choices are:

    Some / Many Words
    Some / Many Sentences
    Some / Many Paragraphs
    Some / Many Digits
    Some / Many Letters
    Sample URL.

    Notes: The dummy text should be selected according to the expected kind of content that will be provided by the user for the given placeholder. For example, if the placeholder appears in a context where it will be filled out with a number (like a ZIP-code or phone number), then Some Digits would be the closest match. If the placeholder will be filled out with a lot of text, then Many Paragraphs is probably better.

    The Sample URL option is a special choice that will result in a replacement that looks like a dummy "http://" URL. This type is appropriate if the expected content will most likely be a URL (for example, in a context like "Click this URL {{#Placeholder}} to go to...").

  - **Dummy Images** – For placeholders of the binary types, the following choices are available to display dummy images:

    No Sample Image
    Small Square (20x20 Pixel)
    Medium Square (100x100 Pixel)
    Large Square (500x500 Pixel)
    Small Vertical Oblong (10x20 Pixel)
    Medium Vertical Oblong (50x100 Pixel)

Large Vertical Oblong (250x500 Pixel)
Small Horizontal Oblong (20x10 Pixel)
Medium Horizontal Oblong (100x50 Pixel)
Large Horizontal Oblong (500x250 Pixel)

> **Note:** The sizes of the dummy images will only be observed if the <img> tag itself does not have any "width" or "height" attributes. If these attributes are present, then they override the sizes of the selected dummy image and the image will be stretched or shrunk accordingly.

- **Disabled Placeholder Highlighting in HTML** – Only available for Plain Text or HTML placeholders (for binary placeholders, highlighting is always disabled).

  In some situations where the template is being displayed to the user, LISTSERV Maestro attempts to highlight the currently selected placeholder with a high contrast background color, making it easier for the user to see which placeholder is selected.

  This highlighting is achieved by embedding additional HTML code into the previewed template (embedded during the preview only). In some situations, this additionally embedded HTML code disrupts the actual HTML code of the template that surrounds the placeholder, which causes the template to appear "broken".

  In these cases, it is best to disable the highlighting of the placeholder. This can be done with this option. Or more precisely, if this option is checked, then any appearances of the placeholder in the HTML part of the template will no longer be highlighted. Appearances in the plain text part (if any) will always be highlighted.

Special Prefix & Suffix Rules for Template Placeholders

When using optional placeholders in a template, it is sometimes desirable that some content defined in the template will only appear if the placeholder is actually filled out and will not appear if the placeholder is left empty.

For example, consider an HTML template that is supposed to render a bulleted list with up to five bullets, where each list item (except for the first) is optional, i.e. the user who employs the template during content definition can decide how many bullets there are, simply by filling out some and leaving empty the rest, like this:

```
<ul>
      <li>{{#Placeholder_1}}
      <li>{{#Placeholder_2}}
      <li>{{#Placeholder_3}}
      <li>{{#Placeholder_4}}
      <li>{{#Placeholder_5}}
</ul>
```

This would actually not work because if some placeholders are left empty, then their bullets (rendered by the `<li>` tag) would still be visible, like this (three placeholders filled out, two left empty):

- Content of first placeholder
- Content of second placeholder
- Content of third placeholder
-
-

It would be better if the preceding <li> tag would simply not appear at all, leaving the placeholder empty. To facilitate this, template placeholders in LISTSERV Maestro use a special syntax that allows you to define a prefix and a suffix to the content that will not appear if the user does not enter a content at all (the prefix and suffix are only included if the user actually supplies a non-empty content). An example of this is:

```
{{#NAME prefix#VALUE#suffix}}
```

where #NAME is replaced with the actual placeholder name and where you replace "prefix" with the desired prefix text and "suffix" with the desired suffix text, using the following rules:

- The string #VALUE# (including the "#" enclosing characters) must appear between prefix and suffix (if any are present).
- Both prefix and/or suffix may be empty (may be left out).
- Both prefix and suffix may contain linebreaks. The linebreaks will become part of the prefix/suffix.
- Any characters after "#NAME" and before "#VALUE#" will be part of the prefix, including all linebreaks (even linebreaks that appear immediately before "#VALUE#"), except for any whitespace that appears immediately after the word "#NAME" and that appears on the same line as the word "#NAME". Such whitespace is ignored. Also, if the first line contains only whitespace after the word "#NAME", then the linebreak at the end of this line will also be ignored and will not be part of the prefix. (See below for examples.)
- Any characters after "#VALUE#" and before the closing tag "}}" (or whatever closing tag is applicable in your template) will be part of the suffix, including all linebreaks, even linebreaks that appear immediately after "#VALUE#" or before the closing tag "}}". (See below for examples.)

If the user does not supply placeholder content, then the whole placeholder (starting with the opening tag and ending with the closing tag) will be replaced with an empty string (i.e. the prefix and suffix will not appear in this case).

If the user supplies a non-empty content, then the whole placeholder (again starting with the opening tag and ending with the closing tag) will be replaced with the text that is built by linking the prefix, content, and suffix. This means that, in the "prefix#VALUE#suffix" string, the "#VALUE#" part (including the enclosing "#" characters) is replaced with the user content, and the resulting string (including prefix and suffix) is used to replace the placeholder.

The following examples are placeholders with a prefix and/or suffix. In the examples, linbreaks are shown as "¶" for better readability, and the prefix is marked with a green background, while the suffix is marked with a yellow background.

Simple prefix and suffix

```
{{#Placeholder Prefix Text#Value#Suffix Text}}
```

Prefix and suffix with linebreaks

Note that the whitespace before the first prefix character is not part of the prefix itself, since all

whitespace between "#NAME" and the prefix is ignored, while the whitespace in the first suffix line (after "#VALUE#") is part of the suffix:

```
{{#Placeholder       Prefix First Line¶
Prefix Second Line#Value#     Suffix First Line¶
Suffix Second Line}}
```

Prefix and suffix with linebreaks before the first lines and after the last lines

Notes: that the whitespace and the linebreak on the first line is not part of the prefix, because if the first line contains only whitespace and a linebreak after "#NAME", then both the whitespace and the linebreak is ignored. In comparision, the whitespace and the linebreak immediately after "#VALUE#" is part of the suffix, so the suffix has actually three lines (where the first line is empty except for whitespace).

Also, that the prefix ends with a linebreak, so the content will start on a new line after the prefix. Similarly, the suffix ends with a linebreak, so whatever comes after the placeholder will start on a new line after the suffix:

```
{{#Placeholder¶
Prefix First Line¶
Prefix Second Line¶
#Value# ¶
Suffix Second Line¶
Suffix Third Line¶
}}
```

So the example of above could be written as follows instead:

```
<ul>
    {{#Placeholder_1 <li>#VALUE#}}
    {{#Placeholder_2 <li>#VALUE#}}
    {{#Placeholder_3 <li>#VALUE#}}
    {{#Placeholder_4 <li>#VALUE#}}
    {{#Placeholder_5 <li>#VALUE#}}
</ul>
```

After replacement would then be rendered as desired (three placeholders filled out, two left empty):

- Content of first placeholder
- Content of second placeholder
- Content of third placeholder

#TITLE# in a Binary Placeholder Prefix/Suffix

If the placeholder is a placeholder of one of the Binary types, then the special `#TITLE#` attribute may be used anywhere in the prefix and/or suffix of that placeholder (it may even appear several times).

If the `#TITLE#` attribute is used, then when the placeholder is filled out during content definition, the user will have the option of not only providing the binary source for the placeholder but also a title text. This title text will then be used to replace the `#TITLE#` attribute wherever it appears in the prefix or suffix.

The main use for this attribute is to provide the text for the "title" and/or "alt" attribute of the <img> tag that the binary placeholder is used to generate, for example like this (prefix and suffix color coded like above):

```
{{#BinaryPlaceholder <img src="#VALUE#" title="#TITLE#" alt="#TITLE#">}}
```

Note: Observe how in the example the <img> tag is opened in the placeholder prefix and closed in the placeholder suffix, and that the #TITLE# attribute appears in the suffix as a stand-in for the yet to be supplied "title" and "alt" attributes.

"Multiple Values" Placeholders with Prefix/Suffix

The prefix and suffix have an especially important role when used together with placeholders that may have multiple values – Plain Text (multiple values) or HTML (multiple values). The multiple values of these placeholders are linked to each other, and the resulting value is used to replace the placeholder. This by itself is not very useful, as the user could just as well have supplied the multiple values already as one combined text in a normal single value placeholder. However, multiple value placeholders become useful when a prefix and/or suffix are employed because the prefix/suffix is added individually to each of the supplied multiple values, and only then are the multiple values linked together to build the final replacement value. This means that if a prefix and/or suffix is defined, and the user later supplies several values for the placeholder, then not only will the prefix appear before and the suffix after the supplied value, but the prefix and/or suffix also appears between the various multiple values. This very useful if the template manager wants to define a list of items without already knowing how many items there are going to be.

Consider the bullet list example above. As the example was defined, the template manager had set up a bulleted list with up to five bullets. Since the bullet definition itself (the <li> tag) was defined in the prefix, any empty bullets were effectively avoided should the user decide not to provide values for some of the placeholders.

So far so good. The only problem is that the maximum number of bullets was predefined as five by the template manager. If the user later would want to add a sixth bullet, then he would have to ask the template manager to change the template. With a multiple values placeholder, this problem is avoided. In this case, the definition of the bulleted list in the template would look something like this:

```
<ul>
    {{#MultiPlaceholder <li>#VALUE#}}
</ul>
```

And, the "MultiPlaceholder" placeholder would have to be defined as HTML (multiple values). Then, it would be up to the user to decide how many values to supply for this placeholder.

If no value is supplied (only allowed if the placeholder is not mandatory), then the placeholder will be replaced with an empty string and the "<li>" tag will not appear since it is defined in the prefix, which is not rendered for an empty optional placeholder.

If only a single value is supplied, then the placeholder behaves just like a normal placeholder: The value is used to replace the "#VALUE#" part, and the result, including the prefix "<li>" would be used to replace the placeholder, and the resulting HTML code would look something like this:

```
<ul>
     <li>Single value here...
</ul>
```

But if several values are supplied, then the multi-behavior comes into play. Each of the values would be used individually to replace the "#VALUE#" part, and the prefix "<li>" would be added to each of the values. Then, the result for all values would be linked together and used to replace the placeholder. Therefore, the resulting HTML code would look something like this (sample with four values supplied, linebreaks added for readability):

```
<ul>
    <li>First value here...
    <li>Second value here...
    <li>Third value here...
    <li>Fourth value here...
</ul>
```

The length of this bulleted list would depend only on how many values the user supplies when using the template during a content definition.

#INDEX# in a Multiple Values Placeholder Prefix/Suffix

Another useful feature for multiple values placeholders is the "#INDEX#" attribute that can be used in the prefix and/or suffix. Any appearance of this attribute will be replaced with the index number of the multi-value. For example, assume the following template definition, where "MultiPlaceholder" is of the "HTML (multiple values)" type:

```
{{#MultiPlaceholder <p>#INDEX#. Paragraph: #VALUE#</p>}}
```

Now assume that the user fills out this placeholder with four values. Then, the resulting HTML code would look something like this (linebreaks added for readability):

```
<p>1. Paragraph: First value here...</p>
<p>2. Paragraph: Second value here... </p>
<p>3. Paragraph: Third value here... </p>
<p>4. Paragraph: Fourth value here... </p>
```

The above is an example of how the #INDEX# attribute can be used to generate a numbered list where a normal ordered list using the <li> tag would not be appropriate, or in a plain text, where this tag can of course not be used.

Another useful case where the #INDEX# attribute can be employed is to generate a unique HTML-ID or unique links and unique link anchor names:

```
<!-- Example with multi-placeholder that generates a unique HTML-ID, per multi-value: -->
<table>
       {{#TableRow <tr><td id="ValueCell-#INDEX#">#VALUE#</td></tr>}}
</table>

<!-- multi-placeholder that generates a unique link anchor, per multi-value: -->
{{#ParagraphWithAnchor <a name="Paragraph-#INDEX#"><p>#VALUE#</p></a>}}

<!-- multi-placeholder that generates a unique link to the above anchor, per multi-value: -->
{{#LinkToAnchor <a href="#Paragraph-#INDEX#">#VALUE#</a><br>}}
```

Assuming that "TableRow", "ParagraphWithAnchor", and "LinkToAnchor" are defined as multiple value placeholders, and assuming that the user supplies five values for "TableRow", three values for "ParagraphWithAnchor" and matching three values for "LinkToAnchor", then the resulting HTML would look something like this (linebreaks added for readability):

```
<!-- Example with multi-placeholder that generates a unique HTML-ID, per multi-value: -->
<table>
        <tr><td id="ValueCell-1">First cell value here...</td></tr>
        <tr><td id="ValueCell-2">Second cell value here...</td></tr>
        <tr><td id="ValueCell-3">Third cell value here...</td></tr>
        <tr><td id="ValueCell-4">Fourth cell value here...</td></tr>
        <tr><td id="ValueCell-5">Fifth cell value here...</td></tr>
</table>

<!-- multi-placeholder that generates a unique link anchor, per multi-value: -->
<a name="Paragraph-1"><p>First paragraph value here...</p></a>
<a name="Paragraph-2"><p>Second paragraph value here...</p></a>
<a name="Paragraph-3"><p>Third paragraph value here...</p></a>

<!-- multi-placeholder that generates a unique link to the above anchor, per multi-value: -->
<a href="#Paragraph-1">Click here to go to the first paragraph...</a><br>
<a href="#Paragraph-2">Click here to go to the second paragraph...</a><br>
<a href="#Paragraph-3">Click here to go to the third paragraph...</a><br>
```

Note: Observe how each table cell has a unique ID and each paragraph a unique anchor name and how the links at the bottom point to one of these unique anchors.

Of course, if two placeholders are used in this way (one to generate link anchors and one to generate the matching links to those anchors), then this will only work if the user who uses the template during content definition makes sure that both placeholders have the same number of values, and that the ordering of the values in both placeholders matches; otherwise, the links will point to the wrong anchors, or there will be links without anchors or anchors without links pointing to them.

Note: Any appearance of the #INDEX# attribute in the prefix/suffix of a normal (non-multiple value) placeholder will not be replaced. This attribute shows its special behavior only in the prefix/suffix of a multiple value placeholder.

Multi-Placeholder Repeater

As described above, using a placeholder with multiple values, usually together with the placeholder prefix/suffix, is very useful for creating a list of user supplied items, where the number of items is not known in advance. However, this alone does not cover a situation like the following – assume that the template is meant for a newsletter, where each article in the newsletter is supposed to appear with a heading, a sub-heading, and a body, all with different styles:

Article Heading
Article Sub-Heading
Article Body goes here....

where the matching HTML code looks like this:

```
<div style="font-weight:bold; font-style:italic; margin-bottom:10px">Article Heading</div>
<div style="font-size: 8pt; font-style:italic; margin-bottom:5px">Article Sub-Heading</div>
<div style="font-family: serif;">Article Body goes here....</div>
```

Of course, one solution would be to have a placeholder called "article" and make this placeholder a multi-value placeholder so that there could be several articles. That would mean

each of the values would constitute one article. However, this would mean that the heading, sub-heading, and body of each article would all have to be supplied in the article's value, which would mean that the special styles for the heading and subheading would have to be supplied in the value by the user who uses the template to define the message content. When this happens, the definition of the heading, sub-heading, and body styles would be out of the hands of the template manager and would become the responsibility of the user who uses the template to define the message content, which is probably not what is desired.
The other solution one could think of would be to have three placeholders – one called "heading", one called "sub-heading", and one called "body", and all three would be defined as multiple values placeholders and each could contain the correct styles in the prefix/suffix. The user who uses the template to define the message content would simply have to supply the correct headings, sub-headings, and bodies of all desired articles, making sure to supply them in the correct order in all three placeholders and to supply an equal number of values for all. The styles for each would then be supplied automatically by the prefix/suffix mechanism. But, there would still be a problem because the correct way to put these three placeholders into the template code would be like this:

```
{{#heading <div style="font-weight:bold; font-style:italic; margin-bottom:10px">#VALUE#</div>}}
{{#sub-heading <div style="font-size: 8pt; font-style:italic; margin-bottom:5px">#VALUE#</div>}}
{{#body <div style="font-family: serif;">#VALUE#</div>}}
```

But, the result after replacement (assuming that three values were supplied for each of the three multi-value placeholders) would look like this:

First Heading Value

Second Heading Value

Third Heading Value

First Sub-Heading Value

Second Sub-Heading Value

Third Sub-Heading Value

First Article Body Value
Second Article Body Value
Third Article Body Value

because each of the three multiple values placeholders would simply iterate over all its values, followed by the values of the next placeholder, and so on, which is not the desired result. Instead, what should happen is that the values of the three multiple values placeholders would be repeated in an alternating fashion. The first heading, followed by the first sub-heading, followed by the first body, only then followed by the second heading, followed by the second sub-heading, and so on.

LISTSERV Maestro does provide a mechanism for this special behavior of the multiple values placeholders in form of the multi-placeholder repeater. The syntax of the repeater looks similar to the syntax of a placeholder, even though the repeater is not really a placeholder by itself. The simple form without a separator looks like this:

```
{{#multi:repeat BODY}}
```

The full form with a separator looks like this:

```
{{#multi:repeat BODY{{#multi:separator}}SEPARATOR}}
```

With the following replacements:

- **Opening and Closing tags** – The "{{" and "}}" of above must be replaced with the correct template placeholder opening and closing tags of your template.

- **Repeater Body** – The word "BODY" must be replaced with the body text that shall actually be repeated by this repeater (i.e. the repeater will be replaced with one or more instances of this body text).

- **Repeater Separator** – The word "SEPARATOR" must be replaced with the separator text that shall appear between two instances of the body text. Such a separator is optional (use the simple form described above if you do not want to specify a separator - more about this below).

The body text and the separator text of the repeater can both contain any kind of text, including template placeholders and even further nested repeater tags. Actually, the repeater only really makes sense if the body contains at least one template placeholder, which also is defined as a multiple values placeholder. And to be really useful, a repeater usually contains at least two multiple value placeholders. All occurrences of multiple values placeholders in the repeater body are examined. For each such multi-value placeholder, the number of values (as supplied by the user) is determined. The maximum number of values over all multi-value placeholders then defines the repeat count. This means that the multi-value placeholder in the repeater body with the most values defines the repeat count of the repeater.

The repeater is then replaced with as many instances of the repeater body as defined by the repeat count, all linked together into one long value. During each such instance of the repeater body, each multi-value placeholder that occurs in the body will display only one value from its multi-value list, which is the value that matches the current repeat instance. That is, in the first body instance, all multi-value placeholders will display only their first values. In the second instance, each will display only its second value (if any), and so on. If a separator is defined, then between two instances of the repeater body, one instance of the separator text is inserted. The separator appears only between two instances, it does not appear before the first instance or after the last one.

Note: When using a multi-placeholder repeater in a HTML context, then linebreaks usually do not matter so you can use them to format the repeater in a way that works best for you. However, when using a repeater in a plain text context, then any linebreaks that appear in the BODY or SEPARATOR blocks actually do matter, as they become part of the block they belong to. This is especially true for linebreaks that appear just before or after the {{multi:separator}} tag or just before the closing "}}" tag of the repeater.

To illustrate the repeater, we use the same example as above with the article with heading, sub-heading, and body, where we also want to have a horizontal line between two articles (but not before or after the first and last articles). The same three placeholders "heading", "sub-heading", and "body" as used above are required, and all three must be defined as multiple values placeholders. The correct code looks like this:

```
{{#multi:repeat
     {{#heading <div style="font-weight:bold; font-style:italic;
      margin-bottom:10px">#VALUE#</div>}}
     {{#sub-heading <div style="font-size: 8pt; font-style:italic;
      margin-bottom:5px">#VALUE#</div>}}
     {{#body <div style="font-family: serif;">#VALUE#</div>}}
     {{#multi:separator}}
     <hr>
    }}
```

And, the result after replacement (assuming again that three values were supplied for each of the three multi-value placeholders) would look like this, just as desired:

First Heading Value

First Sub-Heading Value

First Article Body Value

Second Heading Value

Second Sub-Heading Value

Second Article Body Value

Third Heading Value

Third Sub-Heading Value

Third Article Body Value

References

LISTSERV Maestro User's Manual

LISTSERV Maestro Administrator's Manual

http://www.lsoft.com/resources/manuals.asp

The Message Templates: Getting Started whitepaper entitled *No HTML Coding Experience Necessary! Creating Professional Looking HTML Message without Coding*

The Message Templates: Creating Your Own Templates whitepaper entitled *Do-It-Yourself Templates: Using Your Own Content to Create Message Templates*

http://www.lsoft.com/resources/whitepaper.asp