

7

Replacing Variables in SVG and PSD Files

This chapter describes the variable replacement facility in AGS. It contains the following sections:

Overview of Variable Binding and Replacement	starting on page 125
Automatic Variable Replacement	starting on page 127
Replacing Variables Explicitly	starting on page 128
Specifying Replacement Data	starting on page 130

Overview of Variable Binding and Replacement

Variables provide a convenient and portable way to replace elements of a file. You can define and replace variables in SVG and PSD source files.

When you define variable bindings, you create a definition in the file that associates a variable name with a particular element in the file. You can bind a variable to a major element such as an entire text flow in a PSD file or the contents of a graph in an SVG file, or to a very specific feature such as the fill attribute of a circle in an image. To replace the element, all you need to know is the name of the variable.

Variables give graphic designers the flexibility to make modifications to images without invalidating already existing code in AGS command sets. A designer can change the structure of a template file, and, as long as the variable names remain constant, there is no need to modify or update the associated AGS command set that uses that template.

AGS provides a very flexible mechanism for automatic variable replacement in various circumstances, and also provides a command that you can use to explicitly replace variables at any point in the execution of a command set.

To use the variable replacement facility in AGS, you must first create template SVG and PSD files. Use Adobe Illustrator 10 and Adobe ImageReady 7 (the Web-authoring application that ships with Photoshop 7) to define named variables and bind them to those portions of the file that you want to replace or fill with run-time data. You cannot create new variables or change their bindings with AGS.

- To create an SVG file that contains variable bindings, use the Variables palette in Adobe Illustrator. When you save the SVG file in Illustrator 10, you must export the variable bindings information by checking "Include Extended Syntax for Variable Data" in the "Save as SVG" dialog.

Illustrator 10 can create SVG file variables of type *visibility*, *graph*, *text*, and *image replacement*. AGS can recognize and replace all of these variable types.

- To create a PSD file that contains variable bindings using the Variables palette in ImageReady 7.

ImageReady 7 can create PSD file variables of type *visibility*, *text replacement*, and *image replacement*.

NOTE: In Adobe GoLive version 6.0, you can bind only *text* and *visibility* variable types.

AGS can replace an element bound to a variable with new data that you supply in XML format. AGS can replace variables automatically when you load a file, or you can use the `applyVariables` command to replace variables explicitly in the course of executing a command set.

To replace variables in a file, you load two separate source files into AGS: the replacement data, and the SVG or PSD file containing the variables. You can load either or both of the files as part of the AGS request (from the command line or API), a command file can load the files using the `loadContent` command, or you can combine these methods. You can also load the replacement data inline in the `loadContent` command.

You can use the same replacement data for several source files, or replace variables in the same source file with several sets of data. The automatic variable replacement behavior depends on how the files are loaded and how the replacement data is named. You can do any of the following:

- Replace variables completely automatically, passing both source files in the request, without specifying a command set.
- Replace variables automatically, passing in both source files, then continue processing with a command set.
- Provide only the replacement data file with the request. AGS uses that data to replace variables automatically as commands load different source files containing variables.
- Load the replacement data using the `loadContent` command, giving it the special content name `data`. AGS uses that data to replace variables automatically as commands load different source files containing variables.
- Load and name both the data and variable source files using the `loadContent` command, and replace variables explicitly using the `applyVariables` command.

Automatic Variable Replacement

You can cause AGS to replace variables in an SVG or PSD file automatically (without an explicit command) in a number of ways. Automatic replacement depends on how and when you load the SVG or PSD source file containing variables and the XML file containing the replacement data.

For AGS to perform automatic variable replacement, the replacement data must be loaded and named with the reserved name `data`, before the file containing the variables. You can load the XML variable replacement data in any of these ways:

- Use an explicit `loadContent` operation to load the file or inline data and name it `data` using the `out` attribute.
- On the command line, specify the data file in the `-data` option to the AGS command.
- When constructing a request with one of the AGS APIs, use the `addVariable` method to add each variable name and replacement value to the request. AGS collects all of the values into an XML content holder called “`data`” before executing the command set.

You can pass in or load the data and source files in any combination. AGS automatically replaces variables in an SVG or PSD file whenever there is an XML content holder named `data` available. This happens in any of the following circumstances:

- You pass in both an SVG or PSD source file (using the `-source` option) and XML replacement data (using the `-data` option) on the command line. If you also specify a command set, AGS performs the variable replacement before executing any of the commands. If you do not specify a command set, AGS performs the replacement and saves the updated file with the name `Untitled`.
- You pass in XML replacement data on the command line or with the request, then in the command set use the `loadContent` command to load an SVG or PSD file. The variable replacement is performed immediately after the file containing the variables is loaded. The variable data is accessible from AGS as XML content named `data`, and will be used to replace variables for any subsequently loaded SVG or PSD files as well.
- In the command set, you load an XML data file (or inline data) containing the replacement values using the `loadContent` command, name it `data` using the `out` attribute, then use the `loadContent` command to load an SVG or PSD file. The variable replacement is performed immediately after the file containing the variables is loaded. The XML content named `data` will be used to replace variables for any subsequently loaded SVG or PSD files as well.

If there is no XML content named `data` when an SVG or PSD file is loaded, automatic variable replacement does not occur and any original variable values remain unchanged.

EXAMPLE 7.1 Automatic variable replacement on load

In this example, the variables in `sales.svg` are automatically replaced by the data values in `myData.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<commands>
  <loadContent source="myData.xml" out="data" />
  <loadContent source="sales.svg" />
  <saveContent name="salesWithMyData.svg" />
</commands>
```

EXAMPLE 7.2 Automatic variable replacement from the command line

This command replaces the variables in the SVG file `template.svg`, then saves it to a new file named "Untitled." It does not execute any other commands.

```
AGS -source template.svg -data myData.xml
```

Replacing Variables Explicitly

Use the `applyVariables` command to replace variable values explicitly at any point during the execution of the command set, after loading and naming both the replacement data and the file containing the variables.

If you load the replacement data using the `loadContent` command (rather than passing it in as part of the request) you must name it using the `out` attribute:

- If you use the reserved name `data`, it is the same as passing the replacement data on the command line or with the request; AGS automatically applies that data to any SVG or PSD file that you load. The name `data` is also the default data content name for the `applyVariables` command. If you have named the content data, you need not specify the `data` attribute in the command.
- If you use any name other than the reserved name `data`, you must pass the name in the `data` attribute of the `applyVariables` command in order to apply those values to SVG or PSD content.

It is not necessary to name the source file containing the variables, as long as it is the current content when the `applyVariables` command executes. However, if you name it with the `out` attribute of the `loadContent` command, you can make it current by specifying that name with the `in` attribute of the `applyVariables` command.

When you replace variables explicitly, you can load the files in any order, and perform other manipulations before or after the variable replacement. This differs from automatic variable replacement, where the data file must be loaded first, and the replacement is performed immediately after the file containing variables is loaded.

NOTE: When setting up the template file and data file loads, take automatic replacement into account to avoid replacing variable values twice with the same data.

For further information on the `applyVariables` command, see the *Adobe Graphics Server Command Reference*.

EXAMPLE 7.3 Loading multiple data sets for a single template file

This example loads a single SVG template file, then replaces the variables with three different sets of data, writing out a new GIF file each time.

```
<?xml version="1.0" encoding="UTF-8"?>
<commands>
  <!-- sales template and yearly sales data -->
  <!-- load common sales template -->
  <loadContent source="sales.svg" out="salesTemplate" />
  <!-- processing sales data for 2001 -->
  <loadContent source="data01.xml" out="sales2001" />
  <applyVariables in="salesTemplate" data="sales2001" />
  <saveOptimized name="sales2001" appendExtension="true" />
  <!-- processing sales data for 2002 -->
  <loadContent source="data02.xml" out="sales2002" />
  <applyVariables in="salesTemplate" data="sales2002" />
  <saveOptimized name="sales2002" appendExtension="true" />
  <!-- processing sales data for 2003 -->
  <loadContent source="data03.xml" out="sales2003" />
  <applyVariables in="salesTemplate" data="sales2003" />
  <saveOptimized name="sales2003" appendExtension="true" />
</commands>
```

EXAMPLE 7.4 Streaming in replacement data

In this example, a command set loads replacement data inline in a `loadContent` command, naming the XML content `newData`. It then loads the source PSD file containing the variables, and uses the `applyVariables` command to explicitly replace the variable values in the file with the new data.

This example replaces variables of types *image replacement* and *text replacement*.

```
<?xml version="1.0" encoding="UTF-8"?>
<commands>
  <loadContent out="newData">
    <data>
      <description>Great guitar sale</description>
      <image>guitar.psd</image>
      <price>$100</price>
      <percentage>Now 30% off!</percentage>
      <title>Guitar</title>
    </data>
  </loadContent>
  <loadContent source="product_template.psd" out="template" />
  <applyVariables in="template" data="newData" />
  <saveContent name="product_with_values" appendExtension="true" />
</commands>
```

Specifying Replacement Data

A replacement data file is an XML file containing variable replacement values inside a data element. Each replacement value is contained in a subelement named for the variable. For example:

```
<data>
  <myVariable1>value</myVariable1>
  <myVariable2>
    <element>
      <subelement attr="value" />
    </element>
  </myVariable2>
</data>
```

For details of what kind of replacement data to provide for each type of variable, see [“SVG Variable Replacement Syntax” on page 130](#) and [“PSD Variable Replacement Syntax” on page 135](#).

Variable Replacement Error Behavior

AGS expects the source SVG or PSD file to contain a variable definition to match every replacement value found in the data.

- If the data file contains a variable name that is not defined in the destination SVG or PSD file, AGS reports an error.
- If any variable value is missing from the data file, AGS reports a warning, and the variable value in the destination file remains unchanged.

The data file need not replace every variable defined in the destination file. Any variable that is not mentioned in the replacement data remains unchanged.

You are responsible for providing data in the correct format for the file format and variable type that you want to replace. AGS does not distinguish between SVG and PSD replacement data, or validate the replacement data. If you specify an existing variable in the replacement data, AGS replaces the bound element with the provided replacement value. If the replacement value is not in the correct format, the resulting file may be syntactically incorrect.

SVG Variable Replacement Syntax

You can set variables that reference standard elements of the SVG file, as well as variables that allow easy and convenient modification of the graphing, text flow, and image replacement Adobe foreign object extensions. In a source SVG file, variable definitions are stored in the metadata portion of the file. A set of variable bindings defines the attributes and elements that can be replaced with specific data.

Variable replacement data in an XML file is enclosed in a data element. The file contains an element for each variable to be replaced, where the name of the element corresponds to the

name of the variable. You set the variable names outside AGS, in Illustrator. The replacement value that you provide must be appropriate to the variable type:

- For a simple variable bound to an attribute, the replacement value is the new attribute value.
- The replacement value for a *visibility* variable is the new visibility value, true or false.
- The replacement value for an *image replacement* variable is the new embedded image data, or a file reference to the new referenced image file.
- For *text* and *graph* variables, the variable-name element contains the new element and its subelements with which to replace the original element, as shown in the examples below.

A graph variable can be replaced by new graph data in the full graph syntax, as shown in [“Replacing a graph variable” on page 133](#), or in an abbreviated syntax, as shown in [“Abbreviated graph syntax data” on page 134](#).

A text variable can be replaced at four different levels of the text flow hierarchy:

- If the replacement data value is a *flow* element, AGS replaces the *flow* element in the source content with the new *flow* element.
- If the replacement data value is text only, AGS replaces the text of the first *span* element of the first *p* element (or the text of the first *p*, if it has no *span*) of the *flow* element in the source content with the new text. It removes all other *span* and *p* elements.
- If the replacement data value is a *p* element, AGS replaces all the child elements of the *flow* element with the new *p* element.
- If the replacement data value is a *span* element, AGS replaces all the child elements of the first *p* element of the *flow* element with the new *span* element.

It is possible to generate replacement data files programmatically by queries to a database, or by wrapping a series of user inputs into an XML file in the correct format.

EXAMPLE 7.5 Basic variable replacement

The source SVG file contains a simple variable named `fillcolor1` that is bound to the `fill` attribute of a circle. The following replacement data specifies a value of `red` for the variable:

```
<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <fillcolor1>red</fillcolor1>
    </data>
  </loadContent>
  <loadContent source="replacement-target.svg"/>
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>
```

EXAMPLE 7.6 Replacing a visibility variable

The source SVG file contains a variable of type *visibility* with the name `showimage`, bound to the element `Image1`. In the SVG, this trait is actually controlled by the `display` attribute, which has a value of `inline` or `none`. This replacement data specifies a value of `true` for the variable `showimage`, which causes AGS to set the value of `Image1/@display` to `inline`.

```
<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <showimage>true</showimage>
    </data>
  </loadContent>
  <loadContent source="replacement-target.svg"/>
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>
```

EXAMPLE 7.7 Replacing an image variable with an image file reference

The source SVG file contains a variable of type *image replacement* with the name `newimage`, bound to the element `Image2`.

```
<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <newimage>images/ducky.png</newimage>
    </data>
  </loadContent>
  <loadContent source="replacement-target.svg"/>
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>
```

This replacement data specifies a URI as the value of the `newimage` variable, which causes AGS to set the value of `/Image2/foreignObject/imageReplacement/@xlink:href` to that URI. When the image is displayed, the relative path will be interpreted with respect to the SVG file location.

EXAMPLE 7.8 Replacing an image variable with embedded image data

The source SVG file contains a variable of type *image replacement* with the name `newimage`. This example replaces the image with base-64 encoded image data, which the command writes to an optimized format and embeds in the file.

```
<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <newimage>data:image/gif;base64,
        R0lGODlhBQAFATAAABsbG09PTyH5BAAAAAA
        LAAAAAFAAUAAAIHjB9woItXAAA7
      </newimage>
    </data>
  </loadContent>
  <loadContent source="replacement-target.svg"/>
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>
```


EXAMPLE 7.9 Replacing a graph variable

The source SVG file contains a variable of type *graph* with the name `newsalesdata`, bound to the graph object named `Sales1`.

This replacement data specifies data and values subelements as values of the `newsalesdata` variable. AGS uses that information to replace the values of

`/Sales1/foreignObject/graphdata` and `/Sales1/foreignObject/graph/values`.

```
<data>
  <newsalesdata>
    <data numDataColumns="4">
      <propertyRow key="name">
        <value/>
        <value>Seattle</value>
        <value>San Jose</value>
        <value>Minneapolis</value>
      </propertyRow>
      <values>
        <row>
          <value key="name">Sun</value>
          <value>0</value>
          <value>1</value>
          <value>2</value>
        </row>
        <row>
          <value key="name">Mon</value>
          <value>2</value>
          <value>0</value>
          <value>1</value>
        </row>
        <row>
          <value key="name">Tue</value>
          <value>-3</value>
          <value>0</value>
          <value>2</value>
        </row>
        <row>
          <value key="name">Wed</value>
          <value>0</value>
          <value>0</value>
          <value>1</value>
        </row>
        <row>
          <value key="name">Thu</value>
          <value>1</value>
          <value>0.5</value>
          <value>1.5</value>
        </row>
        <row>
          <value key="name">Fri</value>
          <value>4</value>
          <value>1</value>
          <value>5</value>
        </row>
        <row>
          <value key="name">Sat</value>
          <value>3</value>
          <value>0</value>
          <value>1</value>
        </row>
      </values>
    </data>
  </newsalesdata>
</data>
```

```

        </row>
      </values>
    </data>
  </newsalesdata>
</data>

```

For further information on graphing, see [Chapter 8, “Creating Dynamic Graphs in SVG Files.”](#)

EXAMPLE 7.10 Abbreviated graph syntax data

Here is an example of abbreviated data file syntax, shown for the variable `newsalesdata` from the previous example. Note that there must be a comma preceding the first line of data, which reserves a spot for the name of a row title. Semicolons delimit the values.

```

<newsalesdata>,Seattle, San Jose, Minneapolis;Sun,0,1,2;Mon,2.0,1;
Tue,-3,0,2;Wed,0,0,1;Thu,1,0.5,1.5;Fri,4,1,5;Sat,3,0,1</newsalesdata>

```

EXAMPLE 7.11 Replacing text using a text variable

The source SVG file contains a variable of type *text* with the name `newcaption`, bound to the text object named `Caption1`.

```

<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <newcaption>Hello, AGS!</newcaption>
    </data>
  </loadContent>
  <loadContent source="replacement-target.svg"/>
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>

```

This replacement data specifies the value of the `newcaption` variable. AGS sets the value of `"/Caption1/foreignObject/flowDef/flow"` (or its first `<p>` or `` subelement) to that value.

EXAMPLE 7.12 Replacing text attributes using a text variable

The source SVG file contains a variable of type *text* with the name `newSamples`, bound to the text object named `SampleColors`.

```

<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <newSamples>
        <flow xmlns="http://ns.adobe.com/Flows/1.0/">
          <p>Sample 1: </p>
          <p><span fill="red">Red</span></p>
        </flow>
      </newSamples>
    </data>
  </loadContent>
  <loadContent source="replacement-target.svg"/>
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>

```

This replacement data provides an entire flow element, with subelements and attributes that define the text style, as well as the text itself. When AGS replaces the element, the text in the SVG content changes to the two paragraphs “Sample 1:” and “Red” while the fill color of the text “Red” changes to the color red:

Sample1:

Red

For further information on text flows, see [Chapter 6, “Changing Text in Images.”](#)

PSD Variable Replacement Syntax

Unlike SVG, PSD is a binary format. When you load a PSD file into AGS, you can access and manipulate certain features using an abbreviated XPath syntax with the `set` command. AGS translates the elements and attributes that you set in this way to and from their PSD-format equivalents. Only a limited set of features are accessible through the `set` command.

You can use ImageReady 7 to create and bind variables of the following types:

- *visibility*: The variable can be bound to an image, a layer, or a layer set. The replacement value is `true` or `false`.
- *text replacement*: The variable can be bound to the text flow in a text layer. The replacement value is simple text, which replaces all of the text in the layer. Text attributes such as the font are unchanged. For more information on text replacement methods and text flows, see [Chapter 6, “Changing Text in Images.”](#)
- *pixel replacement*: The variable can be bound to an image in an image layer. The replacement value is a new image file specification, where the file is accessible to the AGS server process, or the new image in base-64 encoded format.

When AGS replaces an image value for a pixel-replacement variable, it uses the same mechanism as the `replacePixels` command, but with position and scaling settings that match those set in the Variables dialog box in ImageReady.

EXAMPLE 7.13 Replacing a visibility variable

The source PSD file contains a variable of type *visibility* with the name `showimage`. This replacement data sets the value to `true`.

```
<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <showimage>true</showimage>
    </data>
  </loadContent>
  <loadContent source="replacement-target.psd" />
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>
```

EXAMPLE 7.14 Replacing a text variable

The source PSD file contains a variable of type *text replacement* with the name `caption`. This replacement data sets the value to "new title".

```
<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <caption>"new title"</caption>
    </data>
  </loadContent>
  <loadContent source="replacement-target.psd" />
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>
```

EXAMPLE 7.15 Replacing a pixel variable using a file reference

The source PSD file contains a variable of type *pixel replacement* with the name `product`. This replacement data sets the value to the contents of the referenced JPEG file. AGS resolves the relative path to that file with respect to the base input URI.

```
<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <product>images/guitar.jpg</product>
    </data>
  </loadContent>
  <loadContent source="replacement-target.psd" />
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>
```

EXAMPLE 7.16 Replacing a pixel variable with an embedded image

The source PSD file contains a variable of type *pixel replacement* with the name `product`. This example loads a replacement image in base-64 encoded format, giving it the reserved name `data`. When it loads the source PSD file, the pixel value is replaced automatically with the new image data. The example then saves the file with the new image value to an optimized format (as determined by its optimization settings).

```
<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <product>data:image/gif;base64,
R0lGODlhGAACALMAAP/u7v+7u/8REF93d/8iIv/d3f9mZv9VVf+
qqv9ERP+IiP+Zmf/MzP8zM/8AAP///yH5BAAAAAALAAAAAYABwAAA
TJ8MLJq70YA0aT+5/SgaEycJIHCgAVkI6yDMWkfoZFkGJlOyiKgme
5JS4AAag3+iwmrYlhWfwIhIEJg+oLUQg5m7frqEkWH/MDHasemsyHmp5y
ZCVbq7DtI1CmIE8SBXwdcQGIDQ5HEwd0DwdRFWxBAY+CFkkOYWeZGQy
IFJIZpKWmpCY2HacHVxIOFLCmCQQFA4oACQ2ECQ05uxYBAwkvDzEKy
LAXCQmPCgQE8bIyXIKBwFqUggIB7cEAA0NAAQNvg0JQafr7BcRADv=
      </product>
    </data>
  </loadContent>
  <loadContent source="replacement-target.psd" />
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>
```

The Automatic Text Variable

For any PSD file with one or more text layers, AGS defines a PSD variable named `defaultTextLayer`. This allows you to automatically replace the text in the topmost text layer upon loading the file into AGS, without having to first define such a variable outside AGS. (The topmost layer is the one with the highest index number.) For more information, see the *Adobe Graphics Server Command Reference*.

EXAMPLE 7.17 *Replacing the default text variable*

To replace the text in the topmost text layer, specify the new text value for the default variable named `defaultTextVariable` in the XML replacement data:

```
<?xml version="1.0" encoding="UTF-8"?>
<commands resultOverwrite="true">
  <loadContent out="data">
    <data>
      <defaultTextVariable>"new text"</defaultTextVariable>
    </data>
  </loadContent>
  <loadContent source="replacement-target.psd" />
  <saveOptimized name="var-replacement" appendExtension="true" />
</commands>
```

