



Adobe Graphics Server Command Reference

Version 2.0



ADOBE SYSTEMS INCORPORATED

Corporate Headquarters

345 Park Avenue
San Jose, CA 95110-2704
(408) 536-6000
<http://www.adobe.com>

Copyright 2002 Adobe Systems Incorporated. All rights reserved.

Adobe® Graphics Server 2.0 Command Reference for Solaris™ and Windows™

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe Systems Incorporated. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement. The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner. Any references to company names in sample templates are for demonstration purposes only and are not intended to refer to any actual organization.

Adobe, the Adobe logo, Adobe Caslon, Adobe Garamond, Adobe Jenson, Acrobat, Acrobat Reader, FrameMaker, Distiller, GoLive, Illustrator, ImageReady, InDesign, Kozuka Gothic, Kozuka Mincho, Minion, Myriad, Photoshop, PostScript, and XMP are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Apple, Mac, Macintosh, Power Macintosh and TrueType are trademarks of Apple Computer, Inc., registered in the United States and other countries. Java, Java Applet, Java Servlet and JavaScript are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Microsoft, Windows, Windows NT, and OpenType are either a registered trademark or a trademark of Microsoft Corporation in the United States and other countries. Solaris is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark of The Open Group. X Window System is a trademark of The Open Group. All other trademarks are the property of their respective owners.

© 1989 All Rights Reserved by Proximity Technology Inc. Proximity and Linguibase are registered trademarks of Proximity Technology Inc. The Proximity/Merriam-Webster Database © 1984-1990 Merriam-Webster Inc. © 1990 - All Rights Reserved. This product includes software developed by the Apache Software Foundation (<http://www.apache.org>). © 1998-2001 The Apache Group. All rights reserved. © 1995-2001 International Business Machines Corporation and others. All rights reserved. © 1990 David Koblas. Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted. This software is provided "as is" without express or implied warranty. © 1994 Anthony Dekker.

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR DOCUMENTATION.

The name and trademarks of copyright holders may NOT be used in advertising or publicity pertaining to the software without specific, written prior permission. Title to copyright in this software and any associated documentation will at all times remain with copyright holders. This product contains an implementation of the LZW algorithm licensed under U.S. Patent No. 4,558,302.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110-2704, USA.

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.



Contents

Preface	5
Chapter 1 Introduction to AGS Commands	7
File and Content Formats	7
Specifying Input and Output Content	8
Specifying URIs in Commands	9
Content References as Attribute Values	10
Command Summary	11
Command Files and General System Commands	11
File, Format, and Metadata Commands	12
Image Manipulation Commands	13
Chapter 2 AGS Commands Reference	15
addLayer	15
appendMetadata	17
applyClipPath	19
applyVariables	21
assignProfile	23
autoContrast	25
autoLevels	26
canvasSize	27
canvasSizeRelative	29
commands	31
convertPDFToRaster	33
convertProfile	36
convertPSToRaster	39
convertRasterToEPS	42
convertRasterToPDF	44
convertSVGTToPDF	45
convertSVGTToRaster	52
convertTo	53
createMetadata	55
crop	56
deleteLayer	58
exportMetadata	60
flatten	61
flip	63

imageInfo	64
imageSize	67
importMetadata	70
info.	72
loadContent	73
optimizeToSize	76
registerMetadataNamespace	78
removeMetadata	80
replacePixels	81
replaceText	84
rotate	86
saveContent	87
saveOptimized.	90
set	94
setFileFormat	99
setLayerPosition.	101
setMetadata	103
trim	105
unsharpMask	107
version.	109

Index.	111
-----------------------	------------



Preface

Introduction

Adobe® Graphics Server (AGS) is a dynamic publishing platform that extends the distinctive competence of Adobe in imaging and static and animated graphics to the server infrastructure. AGS allows the extension of current workflow for dynamic HTML (HyperText Markup Language) to include visually rich images and graphics.

Purpose

This guide contains reference information for the AGS command set that provides the capability to create and manipulate a variety of graphics and document formats. The commands are expressed in XML syntax. This guide describes the syntax, attributes, and usage of each AGS command.

NOTE: "AlterCast" was the name of the previous version of this product. References to the name "AlterCast" still appear in both the product and the documentation. Such references are valid and required for the product to function correctly.

Audience

The audience for this manual is programmers or other technical personnel familiar with XML (eXtensible Markup Language) concepts. Some familiarity with Illustrator®, Photoshop®, and ImageReady®, and the file formats these applications use such as SVG and PSD is also assumed.

Reference Organization

This reference is organized as follows:

Chapter 1, "Introduction to AGS Commands"	Introduces basic AGS concepts and summarizes the AGS XML commands by functional category, with brief descriptions.
Chapter 2, "AGS Commands Reference"	Gives the syntax, complete descriptions, and code samples for the AGS XML commands.

Related Documentation

The following documents are available in addition to this reference:

- *Installation and Configuration Guide*: Contains installation instructions, configuration requirements, and administrative guidelines for AGS.
- *Adobe Graphics Server Developers Guide*: Provides architectural overviews and usage information for AGS developers. Explains how to construct and use command files, and how to manipulate images with AGS.

- *Adobe Graphics Server Command Quick Reference*: Provides brief syntax statements for the AGS XML command set.
- *Adobe Graphics Server API Reference*: Provides a reference for the interfaces used to invoke AGS, including the Java, Perl, and COM APIs, the Web service interface, and the command line interface, and how to handle the information received in response.
- *Adobe Graphics Server XML Grammars Reference*: Explains how AGS uses and supports XML formats and protocols. Includes reference information for formats and grammars that AGS uses, defines, or extends.

Document Conventions

This table describes the font conventions that the AGS SDK documentation uses.

Item	Use and examples
File names are shown in Courier font.	/psd/stoplight.psd
Code items within plain text are shown in Courier font.	The loadContent command The getName method
Code examples set off from plain text are shown in Courier font.	Syntax: <commands />
Document titles, new terms, and variables or other placeholders within code are designated by an <i>italic</i> font.	<i>Adobe Graphics Server Developers Guide</i> These translation methods are known as <i>rendering intents</i> because... in="contentName"

Documentation Problems

If you discover any errors in or have any problems with this documentation, please e-mail us at:

doc_problems@adobe.com

Please describe the error or problem as completely as possible and give us the document title and the page number or page range.

1

Introduction to AGS Commands

This chapter introduces the AGS XML command set by functional category, and provides a brief description of each command.

This chapter contains the following sections:

File and Content Formats	starting on page 7	Describes the relationship of file formats and AGS content types, and the mechanisms for specifying and referring to loaded content.
Command Summary	starting on page 11	Provides an overview of the entire command set, with brief descriptions.

File and Content Formats

When you load a file into AGS, it is kept as content of a particular type. Content types are internal AGS formats that correspond generally, but not exactly, to file formats. For example, the Raster content type is a PSD-like format that AGS uses to store and manipulate all image types. AGS transforms file content to an internal content type upon load, and transforms content types to various file formats upon save. The following table summarizes the content types and their relations to file formats:

Input file formats	Content type	Output file formats
PSD, TIFF, GIF, JPEG, PNG	Raster	normal save: PSD, TIFF optimized: GIF, JPEG, PNG8, PNG24, WBMP
SVG	SVG	normal save: SVG optimized: GIF, JPEG, PNG8, PNG24, WBMP
PDF	PDF	normal save: PDF optimized: Web-optimized PDF
PostScript [®] , EPS	PostScript	PostScript, EPS
XML	XML	XML
XMP	XMP	XMP

Commands generally accept a particular content type as input, and return a particular content type as output. Not all commands work on all content types.

There are commands to explicitly convert content types to other content types within AGS. The load, save, and convert command descriptions give explicit details about the format

transformations that AGS performs. For details of the file formats and extensions with which each content type is saved, see the [saveContent](#) and [saveOptimized](#) commands.

Specifying Input and Output Content

Almost all AGS commands operate on the current content. When you invoke AGS on the command line, the file that you specify with the `-source` option (if any) becomes the initial current content. When you load a file into AGS with the `loadContent` command, that file becomes the current content. Each command makes its result the current content, to be acted upon by the immediately subsequent command.

You can name content for future reference, by specifying the name in the optional `out` attribute. You can then refer to this *named content* in other commands, either to make it current, or to get data out of it to merge with the current content. You can use the optional `out` attribute for any command to specify a name for the command result. You can then refer to that named content in any subsequent command, even when it is no longer current. For a complete discussion of the concepts of content currency and naming, see the *Adobe Graphics Server Developers Guide*.

When you are working with multiple files, you normally name and store content in order to reference it later. Initially, you do this by specifying the name in the `out` attribute of `loadContent`. You can then refer to that name using the optional `in` attribute of any command, to make that content the input to the command. Most commands have the optional `in` and `out` attributes that refer to named content.

Attribute	Description
<code>in</code>	<p>Use this attribute to operate on previously named content instead of the current content.</p> <ul style="list-style-type: none"> When present, specifies named content that becomes the input to the command. When not present, the command operates on the result of the immediately previous command (the current content).
<code>out</code>	<p>Use this attribute so that you can perform further operations on the result of this command by passing the name in any subsequent command's <code>in</code> attribute.</p> <ul style="list-style-type: none"> When present, specifies a name for the result of this command. When not present, the result is available only to the immediately subsequent command.

In almost all cases, a command makes its result the current content. If the previously current content was not named, it can no longer be referenced, and will be overwritten.

Some commands operate on two or more content sources. The content from a second source might be merged with the current content, for example, or used as a template for the current content. These commands take a `data` or `source` or `template` attribute that specifies a second source of content. These attributes, like the `in` and `out` attributes, take content names as their values. Unlike the `in` attribute, they never refer to current content, but must refer to named content. For example, you can supply the name of XML content as the value of the

`applyVariables` command's data attribute. The named XML content is used to replace variable values in the current SVG or Raster content.

Some commands (such as `exportMetadata`) generate or extract content of a different type from the input type and make it current, so that the output type is different from the input type. The description for each command gives the content type or types expected for input, and the content type that is current when the command completes execution. If the current content is not of the proper type when you call a command, AGS reports an error.

Specifying URIs in Commands

In commands that load input files, you can specify input located either on the file system or in an API Request object. Similarly, in commands that output result content to some place other than a content holder, you can direct the output either to a file name and location on the file system, or into an API Response object. In many places where you specify these input and output locations, you can use a relative URI; in a few cases, you must specify an absolute URI. For each situation where you must give a URI in a command, this reference tells you whether it can or must be absolute, whether it can be relative, and whether a relative URI has any restrictions placed on it.

All the interfaces establish two base URIs: one for input and one for output. Relative URIs in commands are resolved with respect to these base URIs. What these base URIs are, and how relative URIs are resolved with respect to them is explained in detail in the chapter “Invoking AGS” in the *Adobe Graphics Server Developers Guide*.

The following table summarizes the base URIs for each interface, and the conditions in which each occur. All base URIs that point to a location on the file system begin with `file:///` and identify a location locally visible to the AGS process executing the commands.

TABLE 1.1 *Base URIs for input and output*

Interface	Base URI		Conditions
Java API Perl API COM API Web service	Input:	Request object	Commands in Request object
		Location of command file	Commands in file on AGS server's file system
	Output:	Response object	Result location not set
		Result location	Result location is set
Command line	Input:	Location of command file	Command file supplied
		Current working directory ^a	Command file not supplied
	Output:	Result location	Result location set
		Current working directory	Result location not set

a. The directory in which the command line is invoked.

In most cases, you can override the base input URI in a given command by giving an absolute URI. Similarly, you can override both the base output URI and the result location in a given command by giving an absolute URI. All absolute file URIs must use the file: scheme.

Content References as Attribute Values

AGS defines a special reference function you can use to access named content and retrieve values from it, as long as that content is accessible with the XPath syntax that AGS supports. (See *Adobe Graphics Server XML Grammars Reference* for details of supported XPath syntax). You can use such a content reference anywhere as an attribute value. A content reference can be useful for retrieving values from loaded content at run time. This can be useful, for example, when you do not know ahead of time what those values will be.

The syntax for a content reference is as follows:

```
reference(adobe-content:contentName#XPath)
```

The #*XPath* portion of the specification is optional. If it is not supplied, the expression evaluates to the entire named content.

When AGS executes a command containing a content reference, it first evaluates the reference, then passes the result to the command as the attribute value. If the reference does not evaluate to the expected value type (a content name, a text string, or a color value, for example), the command reports an error.

For example, suppose you load and name simple XML content as follows:

```
<loadContent out="myXml">
  <A><B d="dog" /></A>
</loadContent>
```

- The following reference extracts the string value "dog" from the d attribute of the B element:

```
reference(adobe-content:myXml#/A/B/@d)
```

You could pass this reference for an attribute that expected a text-string value.

- The following reference, with no XPath, evaluates to the entire content (rooted at the A element):

```
reference(adobe-content:myXml)
```

If you passed this reference for an attribute that expected, for example, the name of named content, it would cause an error.

Command Summary

This section briefly summarizes the command set, dividing the commands into general categories, and presenting them in a logical (rather than alphabetical) order. Each brief description is linked to the complete description in the following chapter, where all commands are presented alphabetically.

- [Command Files and General System Commands](#)
- [File, Format, and Metadata Commands](#)
- [Image Manipulation Commands](#)

Command Files and General System Commands

An AGS command file is an XML file that contains a set of processing instructions. You can pass a command file to AGS on the command line or in a request, or include a reference to a local command file in a request.

- The first line of a command file indicates the text encoding:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

AGS supports UTF-8 text encoding. It reads UTF-16 SVG files, but converts them to UTF-8, and writes them to UTF-8.

- The AGS [commands](#) element identifies and encloses all AGS commands in the file. AGS commands are case sensitive. Comments can be included in the following format:

```
<!-- comment format -->
```

The following commands apply to the AGS system. They do not operate directly on loaded content:

TABLE 1.2 *General system commands*

Command	Description
info	Returns information about AGS in XML format.
version	Returns the current AGS version information and stores it in XML format.
registerMetadataNamespace	Registers an XML namespace for use with metadata and requests a prefix for that namespace.

File, Format, and Metadata Commands

The following commands load and save content to and from files of various types, convert among different file formats, operate on file metadata, and replace variable data:

TABLE 1.3 *File, format, and metadata commands*

Command	Description
Load/Save	loadContent Loads a file into AGS for processing.
	saveContent Saves AGS content to the external file system or as a result in a response object.
	saveOptimized Explicitly saves SVG content or RGB Raster content in an optimized format (GIF, JPEG, PNG, or WBMP), or PDF content to Fast Web View PDF.
Conversion	convertTo Converts SVG content to Raster content, or marks Raster content to be saved as PSD or TIFF.
	convertSVGToPDF Converts SVG content to PDF content.
	convertSVGToRaster Converts SVG content to Raster content.
	convertPDFToRaster Converts a page from a PDF document into Raster content.
	convertPSToRaster Converts a PostScript document into Raster content.
	convertRasterToEPS Converts Raster content into EPS format.
	convertRasterToPDF Converts Raster content into PDF format.
	setFileFormat Marks Raster content to be saved as PSD or TIFF.
Metadata	exportMetadata Retrieves the metadata for the current content as XMP content.
	importMetadata Replaces document-level metadata with new metadata specified inline, or with data in named XMP content.
	setMetadata Creates or sets an unstructured metadata property, or sets a structured property item.
	createMetadata Creates a new metadata structured property and its first item.
	appendMetadata Adds a new value to a structured metadata property.
	removeMetadata Removes one or more metadata properties from the current content.
Data	applyVariables Replaces variable values in SVG or Raster content.
	set Creates or modifies elements or attributes in Raster, SVG, and XML content.
	imageInfo Retrieves image information from Raster content.

Image Manipulation Commands

The following commands operate on Raster content:

TABLE 1.4 *Image manipulation commands*

Command	Description
<code>replacePixels</code>	Replaces the pixels in a Raster image layer.
<code>replaceText</code>	Replaces text in a text layer in Raster content.
<code>addLayer</code>	Adds a pixel or text layer to Raster content.
<code>deleteLayer</code>	Deletes a layer from Raster content.
<code>setLayerPosition</code>	Positions a layer in Raster content.
<code>flatten</code>	Flattens a Raster image into a single layer.
<code>assignProfile</code>	Assigns an International Color Consortium (ICC) color profile to be associated with Raster content, without converting the image data.
<code>convertProfile</code>	Converts image data from the current (or a default) International Color Consortium (ICC) color profile to a new color profile.
<code>autoContrast</code>	Automatically adjusts contrast in an RGB image in Raster content.
<code>autoLevels</code>	Automatically adjusts the levels in an RGB image in Raster content.
<code>unsharpMask</code>	Sharpens edges in an RGB image in Raster content.
<code>imageSize</code>	Changes the size of an image.
<code>optimizeToSize</code>	Adjusts optimization settings to compress an RGB image to an approximate file size.
<code>canvasSize</code>	Sets the canvas size for an image.
<code>canvasSizeRelative</code>	Sets the canvas size of an image relative to the current canvas size.
<code>crop</code>	Reduces the canvas bounds of an image, retaining a given rectangle.
<code>trim</code>	Crops an RGB image by removing pixels from the edges based on their color.
<code>applyClipPath</code>	Applies a clipping path to Raster content.
<code>flip</code>	Reverses an image from top to bottom or from left to right.
<code>rotate</code>	Rotates an image.

2

AGS Commands Reference

This chapter is a reference for the AGS XML command set. It lists commands alphabetically, gives example syntax, and describes each attribute.

addLayer

Adds a pixel or text layer to Raster content.

Content Types

Input: Raster

Output: Raster

Syntax

```
<addLayer in="contentName" out="contentName"
  type="pixel|text" position="above|below"
  layer="XPath" name="name" />
```

Attribute	Description
in	Optional. The named content to use as input to the operation.
out	Optional. A name to assign to the result of the operation.
type	Optional. The type of layer to add. The value can be: <ul style="list-style-type: none">pixel (default)text
position	Optional. Where to place the new layer with respect to the existing layer specified by the layer attribute. The value can be: <ul style="list-style-type: none">above (default)below
layer	Optional. The XPath for the layer to which the new layer will be adjacent, in the following format: /psd/layer[@name='name' num]" Default is the topmost layer.
name	Optional. The name of the layer you are adding. The default layer name is "Untitled Layer."

Details

Layers that you add this way have additional properties that you cannot configure with this command, and are therefore initialized with default settings.

- An added pixel layer results in the same defaults as those found in Photoshop. After you add a pixel layer, you can replace the pixels with the `replacePixels` command.
- An added text layer has the same defaults as ImageReady. The default font (Myriad® Pro Regular) for a text layer is specified by AGS. After you add a text layer, you can replace the text with the `replaceText` command.

You can use this command to add a pixel or text layer to content that was originally in another format. Content that was originally loaded from a GIF, JPEG, TIFF, or PNG file or that was converted from the PDF, PostScript, or SVG format has a single, unnamed layer in the Raster content. If you add layer information to a file that was loaded from another format, that information is preserved only if you save the content in the PSD format. If the content is marked to be saved as TIFF, or if you save it to an optimized format, the layers are flattened before the file is saved. For more information, see the `saveContent`, `saveOptimized`, and `setFileFormat` commands.

You can refer to PSD layers and layer sets in Raster content by either name or number. If a layer is a member of a layer set, you must include the layer set in the XPath layer reference. For example:

<code>/psd[layer[@name='grid']]</code>	Refers to a layer by name.
<code>/psd[layer[1]]</code>	Refers to the bottommost layer by number.
<code>/psd[layer[last()]]</code>	Refers to the topmost layer.
<code>/psd[layerSet[1]/@visibility</code>	Refers to the visibility of the entire bottommost layer set.
<code>/psd[layerSet[last()]/@visibility</code>	Refers to the visibility of the entire topmost layer set.
<code>/psd[layerSet[@name='myGroup']/layer[1]]</code>	Refers to the first layer of a layer set named <code>myGroup</code> .

Examples

This example adds a new pixel layer named "Green" below the layer named "Red" for the image file in the current Raster content:

```
<addLayer type="pixel" position="below" layer="/psd[layer[@name='Red']]"
  name="Green" />
```

This example adds a text layer to the top of existing layers, allowing position and layer to default:

```
<addLayer type="text" name="myTextLayerName" />
```

See Also

- [“deleteLayer” on page 58](#)
- [“replacePixels” on page 81](#)
- [“replaceText” on page 84](#)
- [“setLayerPosition” on page 101](#)

appendMetadata

Adds a new value to a structured metadata property.

Content Types

Input: Raster, SVG, PDF

Output: Raster, SVG, PDF

Syntax

```
<appendMetadata in="contentName" out="contentName"
  namespace="namespace"
  path="before|after propertyNameAndItem"
  value="value" />
```

Attribute	Description
in	Optional. The named content to use as input to the operation.
out	Optional. A name to assign to the result of the operation.
namespace	Required. The full name of the namespace in which the property resides.
path	Required. The XPath to the structured-container property name and item at which to add the new value. The path must be preceded by the keyword <i>before</i> or <i>after</i> , to indicate whether to add the new item before or after the named item, and a single space; for example "before name/*1". Use the pattern <i>name/*[1]</i> to specify the first item, or <i>name/*[last()]</i> to specify the last item.
value	Required. The value of the new item in the property.

Details

Structured metadata properties contain multiple values, or items. This command creates a new item value for the structured XMP metadata property specified by *namespace* and *path*. The *path* must specify a currently existing structured property item. To create a new structured property with an initial value, use the *createMetadata* command. To modify the value of an existing item, use the *setMetadata* command.

If the structured property is of the type *sequence*, it contains an ordered sequence of items, and it is important to know where in the sequence the new item is to be added. For *alternative* type, the first item in the sequence is defined as the default value. For the *bag* type, the order is not significant. (For details of the structured property types, see the *addMetadata* command.) The ordering element of *path* (*before* or *after*) is required, and new items are created in the specified order regardless of the structured property type.

- To add the new item before the first existing item, use the pattern "before *name/*[1]*".
- To add the new item to the end of the item list, use the pattern "after *name/*[last()]*".

Example

This example appends a new item with the value, `Keyword2`, after the first item in the metadata property named `Keywords`, in the current content's metadata:

```
<appendMetadata namespace="http://ns.adobe.com/xap/1.0/"  
  path="after Keywords/*[1]" value="Keyword2" />
```

See Also

- [“createMetadata” on page 55](#)
- [“exportMetadata” on page 60](#)
- [“importMetadata” on page 70](#)
- [“registerMetadataNamespace” on page 78](#)
- [“removeMetadata” on page 80](#)
- [“setMetadata” on page 103](#)

*applyClipPath***applyClipPath**

Applies a clipping path to Raster content.

Content Types

Input: Raster

Output: Raster

Syntax

```
<applyClipPath in="contentName" out="contentName"
  target="layer" clipPath="retain|discard" />
```

Attribute	Description
in	Optional. The named content to use as input to the operation.
out	Optional. A name to assign to the result of the operation.
target	Required. The XPath to the layer to modify, in the following format: /psd/layer[@name='name' num]"
clipPath	Optional. Whether to retain the clipping path in the content after it is applied. retain: Retain the clipping path in the result content. discard (default): Remove the clipping path after the information has been transferred to the target layer.

Details

The clipping path in an image is used to define a cropping region when printing. This command applies the clipping path's cropping region on the image by converting it to a vector mask. This command is useful in updating legacy print images for Web use.

The command applies to a single layer. To crop the entire image (as the clipping path would do), you must flatten it before converting the clip path.

Clipping paths are not present in images that were originally in GIF or PNG format. If the Raster content does not contain a clipping path, the command reports a warning and does nothing to the content.

Examples

This example loads a PSD image, flattens it, and applies the clipping path. Finally, it saves the content in optimized format:

```
<commands>
  <loadContent source="clipPath.psd" />
  <flatten/>
  <applyClipPath target="/psd/layer[1]" />
  <saveOptimized name="clipPathApplied" appendExtension="true" />
</commands>
```

This example applies the clipping path to multiple layers, retaining the clipping path until the last conversion:

```
<commands>
  <loadContent source="clipPath.psd" />
  <applyClipPath target="/psd/layer[1]" clipPath="retain" />
  <applyClipPath target="/psd/layer[2]" clipPath="retain" />
  <applyClipPath target="/psd/layer[3]" />
  <saveOptimized name="clipPathApplied" appendExtension="true" />
</commands>
```

See Also

- [“flatten” on page 61](#)
- [“crop” on page 56](#)
- [“trim” on page 105](#)

applyVariables

Replaces variable values in SVG or Raster content.

Content Types

Input: SVG, Raster, and XML for data

Output: SVG, Raster

Syntax

```
<applyVariables in="contentName" out="contentName" data="contentName" />
```

Attribute	Description
in	Optional. The named content to use as input to the operation.
out	Optional. A name to assign to the result of the operation.
data	Optional. Named XML content to use for the variable replacement. When not specified, the command looks for content named data.

Details

This command explicitly replaces the variables in the input content with data values from previously loaded and named XML content. If you do not specify a name for the replacement data, the command looks for data content named data.

- Use Illustrator version 10 to create and bind variables in an SVG file with the **Variables** palette. For SVG files, you can examine and manipulate the variable definition in the source file. For details of the format, see *Adobe Graphics Server XML Grammars Reference*.
- Use Photoshop 7.0 to create and bind variables in a PSD file with the **Variables** dialog box from the **Image** menu.

All replacement data values are enclosed in data tags, and each value is enclosed in tags that give the variable name. For example:

```
<data>  
  <myTextVar>"new text value"</myTextVar>  
  <myVisibilityVar>true</myVisibilityVar>  
</data>
```

The format for each value depends on the type of the variable. For details of how to define replacement data, see the *Adobe Graphics Server Developers Guide*.

You can use this command multiple times to apply a sequence of data values to a common template file. Remember to name or save the result after each replacement.

Automatic Variable Replacement

When you load any PSD or SVG file, AGS attempts to replace variables automatically. It looks for content named data, and if that named content exists, AGS automatically applies variables