The Image class enables you to display bitmapped images. The Image class is a viewer that uses the Image Data Model class to manage the actual image.

Note: The GRPHVIEW class, which is not documented, is a child of the Widget class.

Parent:
  sashelp.fsp.grphview.class

Class:
  SASHELP.FSP.IMAGE.CLASS

Methods specific to the Image class are described here. Inherited methods are described in the Object class and the Widget class.

Dictionary

_attach

Attaches an Image Data Model to an image object
Syntax

CALL NOTIFY (image-name, '_attach', model-id);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>model-id</td>
<td>N</td>
<td>the model identifier (object identifier) of the new model</td>
</tr>
</tbody>
</table>

Example

This example creates a new instance of the Image Data Model and attaches it to an image object named IMAGEVIEW:

```plaintext
/* load the image data model class */
image_data_class = loadclass('sashelp.fsp.imgdat.class');

/* create an instance of the image data model class */
call send (image_data_class, '_new', objid);

/* set the file name of the model */
call send (objid, '_readFilepath', '/images/clown.tif');

/* attach the model instance to the image object. */
/* IMAGEVIEW is the name of the image object */
/* specified in the Attributes window. */
call notify ('imageview', '_attach', objid);
```

_detach

Detaches a model from an image object

Syntax

CALL NOTIFY (image-name, '_detach');
**_getAttributes**

Returns values of image attributes

---

**Syntax**

CALL NOTIFY (image-name, '_getAttributes', list-id);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list-id</td>
<td>N</td>
<td>the identifier of an SCL list containing object attributes</td>
</tr>
</tbody>
</table>

---

**Table 74.1 _getAttributes List Items**

<table>
<thead>
<tr>
<th>List Item</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPT</td>
<td>C</td>
<td>a string no longer than 200 characters containing information about the image size and color map</td>
</tr>
<tr>
<td>FILENAME</td>
<td>C</td>
<td>image path string</td>
</tr>
<tr>
<td>FORMAT</td>
<td>C</td>
<td>original file format, such as GIF</td>
</tr>
<tr>
<td>WIDTH</td>
<td>N</td>
<td>image width in pixels</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>N</td>
<td>image height in pixels</td>
</tr>
<tr>
<td>DEPTH</td>
<td>N</td>
<td>image depth</td>
</tr>
<tr>
<td>TYPE</td>
<td>C</td>
<td>image type: 'CMAP', 'GRAY', 'RGBA'</td>
</tr>
<tr>
<td>NCOLORS</td>
<td>N</td>
<td>specifies the number of colors, if TYPE='CMAP'</td>
</tr>
<tr>
<td>RDEPTH</td>
<td>N</td>
<td>specifies the red depth, if TYPE='RGBA'</td>
</tr>
<tr>
<td>GDEPTH</td>
<td>N</td>
<td>specifies the green depth, if TYPE='RGBA'</td>
</tr>
<tr>
<td>BDEPTH</td>
<td>N</td>
<td>specifies the blue depth, if TYPE='RGBA'</td>
</tr>
<tr>
<td>ADEPTH</td>
<td>N</td>
<td>specifies the alpha depth, if TYPE='RGBA'</td>
</tr>
<tr>
<td>ONERROR_OPTIONS</td>
<td>C</td>
<td>a string containing the option flags currently set. These can be changed by using _onError.</td>
</tr>
</tbody>
</table>

---

**Details**

If the specified list is empty, it will be filled in with all of the image attributes. However, if the list is not empty, it is assumed that a list of valid named attributes will be filled in.

**Example**

This example displays an SCL list of information about an image object:

```call notify (imageview, '_readFilepath',
    filename, 'format=tiff');```
list=makelist();
call notify ('imageview', '_getAttributes',
    list);
call putlist (list, 'Image attributes
    are: ');  

When the program runs, the following lines are written to the LOG window:

Image attributes are: (DESCRIPT='187x209 8-bit
    CMAP, 8 colormap entries'
FILENAME='01010Q/usr/local/images/color/
    misc/canoe.tif' WIDTH=187
HEIGHT=209 TYPE='CMAP' DEPTH=8 NCOLORS=8 )[4]

This example displays specific attributes of an image:

call notify ('imageview', '_readFilepath',
    filename, 'format=tiff');
list=makelist();
list=setnitemc (list, '', 'format');
list=setnitemn (list, 0, 'width');
list=setnitemn (list, 0, 'height');
call notify ('imageview', '_getAttributes',
    list);
call putlist (list, 'Image attributes
    are: ');

When the program runs, the following line is written to the LOG window:

Image attributes are:(FORMAT='TIFF' WIDTH=187
    HEIGHT=209)[317]

_getDataId

Returns the object identifier of the image data model

Syntax

CALL NOTIFY (image-name, '_getDataId', model-id);

Argument | Type | Description
----------|------|----------------------------------
model-id | N    | the identifier assigned to the image data model

Details

To execute Image Data Model class methods on the image, you must first obtain the object identifier of the image data model that is being used by the image object. For more information on Image Data Model class methods, see that class.
Example

This example gets the image data model identifier from the image object and mirrors the image. You can use the data object that is returned as the object identifier for the image data model methods.

call notify ('imageview', '_getDataId', model-id);
call send (model-id, '_mirror');
call notify ('imageview', '_update');

_getInfo

Returns the identifier for an SCL list containing descriptive information about the last point selected on the image

Syntax

CALL NOTIFY (image-name, '_getInfo', list-id);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list-id</td>
<td>N</td>
<td>the identifier of an SCL list containing the descriptive information about the last point selected on the image.</td>
</tr>
</tbody>
</table>

Table 74.2 _getInfo List Items

<table>
<thead>
<tr>
<th>List Item</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>N</td>
<td>x coordinate of the selection, in pixels</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>y coordinate of the selection, in pixels</td>
</tr>
<tr>
<td>ORIGIN_X</td>
<td>N</td>
<td>object's x origin</td>
</tr>
<tr>
<td>ORIGIN_Y</td>
<td>N</td>
<td>object's y origin</td>
</tr>
<tr>
<td>DSWIDTH</td>
<td>N</td>
<td>object's width</td>
</tr>
<tr>
<td>DSHEIGHT</td>
<td>N</td>
<td>object's height</td>
</tr>
<tr>
<td>REG_ULX</td>
<td>N</td>
<td>region's upper left x coordinate</td>
</tr>
<tr>
<td>REG_ULY</td>
<td>N</td>
<td>region's upper left y coordinate</td>
</tr>
<tr>
<td>REG_LRX</td>
<td>N</td>
<td>region's lower right x coordinate</td>
</tr>
<tr>
<td>REG_LRY</td>
<td>N</td>
<td>region's lower right y coordinate</td>
</tr>
</tbody>
</table>

Example

This example gets the coordinates of the last selection for this object:
list=makelist();
call notify ('imageview', '_getInfo',
    list);
call putlist (list, 'Point of last
    selection: ');

When this code executes, the following lines are written to the LOG window:

Point of last selection: (X=67 Y=62 ORIGIN_X=0
    ORIGIN_Y=0 DSWIDTH=140 DSHEIGHT=140
    REG_ULX=145 REG_ULY=152
    REG_LRX=285 REG_LRY=292) [2581]

__print

Prints the current image to a file or to a hardcopy device using the default system printer

__ syntax

CALL NOTIFY (image-name, '_print');

__ details

On UNIX systems, the _print method works only if host printing is enabled. To verify
or change the setting, from a SAS windowing environment window select

   Tools ▶ Options ▶ Preferences

and the DMS tab; then turn on Use Host Printing. Once host printing is enabled,
the _print method sends the image to the file or device selected in the Printer Setup
dialog.

   For more information on using the Printer Setup dialog to specify a destination for
the image, see the help topics for using SAS with your operating system.

   On PC systems, host printing is enabled by default and can be turned off only by
selecting Use forms from the Print Setup dialog under the File menu.

__ see also

   __printSetup" _printSetup" on page 1416

__printSetup

Specifies image printing options

__ syntax

CALL NOTIFY (image-name, '_printSetup', print-list);
Argument Type Description

print-list N specifies the identifier of an SCL list containing the image printing options in Table 74.3 on page 1417.

Table 74.3 _printSetup List Items

<table>
<thead>
<tr>
<th>List Item</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'WIDTH'</td>
<td>N</td>
<td>specifies the width of the printing area in pixels.</td>
</tr>
<tr>
<td>'HEIGHT'</td>
<td>N</td>
<td>specifies the height of the printing area in pixels.</td>
</tr>
</tbody>
</table>

Details

The _printSetup method does not actually initiate printing; it simply saves options for later use by the _print method.

If the width or height specifications are too large for the device, an error occurs.

See Also

_print

_read

Reads an image from an external file, SAS catalog, or device and displays it in an image object

Syntax

CALL NOTIFY (image-name, '_read', pathname=<, 'attributes'>)CALL NOTIFY (image-name, '_read', devicename, 'DEVICE=CAMERA|SCANNER<attributes>);

Argument Type Description

pathname C the name of the external file containing the image or the path string returned by the LNAMEMK function

devicename C the name of a camera or scanner:

'KODAKDC40' Kodak DC 40 camera (available only under the Windows 95 operating system)

'HPSCAN' HP Scanjet scanners (available only under Windows 95, Windows 32S, and HP/UX operating systems)
### Example

This example reads an image stored in a SAS catalog by using the path string defined with an LNAMEMK function:

```
fullpath=lnamemk(5,'sashelp.imagapp.gfkids',
   'format=cat');
call notify ('imageview', '_read_',
   fullpath);
```

This example reads an image from an external file:

```
call notify ('imageview', '_read_', '/usr
   /images/color/sign.tif');
```

This example scans a document:

```
call notify ('imageview', '_read_', 'hpscan',
   'device=scanner dpi=100');
```

### _readCatalog

**Reads an IMAGE entry from a catalog and displays it in an image object**

**Syntax**

CALL NOTIFY (image-name, '_readCatalog', image-entry);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>image-entry</td>
<td>C</td>
<td>the name of the IMAGE entry to read, specified as a three- or four-level entry name: libref.catalog.entry.&lt;IMAGE&gt;</td>
</tr>
</tbody>
</table>

### Example

This statement reads an image that is stored in a SAS catalog:

```
fullpath=lnamemk(5,'sashelp.imagapp.gfkids',
   'format=cat');
call notify ('imageview', '_read_','
   fullpath);
```
call notify ('imageview', '_readCatalog',
    'sashelp.imagapp.gfkids');

__readClipboard

Reads an image from the host clipboard and displays it in an image object

Syntax

CALL NOTIFY (image-name, '_readClipboard');

Details

On some hosts, you can read the clipboard only after you use the WRITE_CLIPBOARD
command.

__readDirpathMember

Reads an image from an external file in a directory that is specified with a pathname

Syntax

CALL NOTIFY (image-name, '_readDirpathMember', dirpath, member<, attributes>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirpath</td>
<td>C</td>
<td>the path to the directory containing the member</td>
</tr>
<tr>
<td>member</td>
<td>C</td>
<td>the name of the file to be read</td>
</tr>
<tr>
<td>attributes</td>
<td>C</td>
<td>file-specific attributes.</td>
</tr>
</tbody>
</table>

Example

This statement reads an image by using a pathname to identify the directory:

call notify ('imageview',
    '_readDirpathMember',
    '/usr/images/color',
    'sign.tif');
_readDirrefMember

Reads an image from an external file in a directory that is identified by a fileref.

Syntax

CALL NOTIFY (image-name, '_readDirrefMember', dirref, member<, attributes>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirref</td>
<td>C</td>
<td>the fileref of the directory containing the member</td>
</tr>
<tr>
<td>member</td>
<td>C</td>
<td>the name of the file to be read</td>
</tr>
<tr>
<td>attributes</td>
<td>C</td>
<td>file-specific attributes.</td>
</tr>
</tbody>
</table>

Example

This statement reads an image using a fileref to identify the directory:

```plaintext
filename ('imgdir', '/usr/images/color');
call notify ('imageview',
  '_readDirrefMember', 'imgdir',
  'sign.tif');
```

_readFilepath

Reads an image from an external file that is specified with a pathname and displays it in an image object.

Syntax

CALL NOTIFY (image-name, '_readFilepath', pathname<, attributes>);
Example

This statement reads an image that is stored in an external file:

```call notify ('imageview', '_readFilepath',
'/usr/images/color/sign.tif');
```

_readFileref

Reads an image from an external file that is identified by a fileref and displays it in an image object.

Syntax

```CALL NOTIFY (image-name, '_readFileref', fileref<, attributes>);
```

Example

This example reads an image that is stored in external file that is identified by a fileref:

```filename ('imgfref', '/usr/images/color/sign.tif');
call notify ('imageview', '_readFileref', 'imgfref');
```