# Chapter 3:

# Attribute Manager (MAT) Protocol

The Attribute Manager (MAT) protocol (protocol ID 16) consists of atoms that let you assign certain characteristics to objects on a form. Objects on a form can be used to perform a range of functions from displaying text to executing actions associated with a button. For example, you can use the **atom$mat_style_id** command to change the style of a text object.

## Form Display Objects

Objects that are used with online service forms include buttons, static text, and list boxes. Several form display objects can be nested on a single form and multiple hierarchies of objects are maintained.

Table 3-1 lists the online service form display objects and includes the object number, object name, and a description of the object:

*Table 3-1: Form Display Objects*

| Number | Object | Description |
|---|---|---|
| 0 | `org_group` | Organizational group—A group containing other objects. |
| 1 | `ind_group` | Independent group—A window. |
| 2 | `dms_list` | Dynamic, multiselect list—A list box where more than one item is selected. |
| 3 | `sms_list` | Static, multiselect list—A group containing check boxes. |

*Table 3-1: Form Display Objects (Continued)*

| Number | Object | Description |
|---|---|---|
| 4 | `dss_list` | Dynamic, single-select list—A list box, standard popup, or editable popup. |
| 5 | `sss_list` | Static, single-select list—A group containing radio buttons. |
| 6 | `trigger` | Trigger—An actionable object, such as an icon, button, list item, menu item, or check box. |
| 7 | `ornament` | Ornament—A static or decorative object, such as a picture or static text. |
| 8 | `view` | View—A view field used for display of larger amounts of text or a graphic with scroll bars if necessary. |
| 9 | `edit_view` | Editable View—An editable view field used as a text input field where members can enter text. |
| 12 | `range` | Range—Used as a graphic representation of a value, such as a bar gauge. |
| 13 | `select_range` | Selectable Range—Used as a user interface for entering numeric data, such as a spin gadget. |
| 17 | `tool_group` | Toolbar—Used on any form or frame window including tab controls. |
| 18 | `tab_group` | Tab Group—Used for tab control of a group of tabbed pages, frames, or forms (for example, list boxes with tabs). |
| 19 | `tab_page` | Tab Page—Used as a tabbed page, frame, or form such as a list box with a labeled tab. |

# Attribute Manager Protocol Atoms

The Attribute Manager (MAT) protocol functions and their associated atoms follow:

*Table 3-2: Attribute Manager Protocol Functions and Atoms*

| Function | Atoms | Page |
|---|---|---|
| Displaying art and graphics | atom$mat_art_animation_rate | 3-7 |
| | atom$mat_art_animation_seq | 3-8 |
| | atom$mat_art_frame | 3-10 |
| | atom$mat_art_id | 3-22 |
| | atom$mat_bool_background_pic | 3-29 |
| | atom$mat_bool_background_tile | 3-30 |
| | atom$mat_bool_graphic_view | 3-63 |
| | atom$mat_bool_invert | 3-71 |
| | atom$mat_bool_repeat_animation | 3-90 |
| Displaying progressive rendering hints | atom$mat_art_hint_font_size | 3-11 |
| | atom$mat_art_hint_height | 3-12 |
| | atom$mat_art_hint_select_placeholder | 3-14 |
| | atom$mat_art_hint_title | 3-15 |
| | atom$mat_art_hint_title_x | 3-16 |
| | atom$mat_art_hint_title_y | 3-20 |
| | atom$mat_art_hint_width | 3-21 |
| Displaying toolbars | atom$mat_bool_child_line_feed | 3-31 |
| | atom$mat_bool_child_movable | 3-33 |
| | atom$mat_bool_child_removable | 3-34 |
| | atom$mat_bool_children_movable | 3-35 |
| | atom$mat_bool_children_removable | 3-37 |
| | atom$mat_bool_customizable | 3-39 |
| | atom$mat_bool_detachable | 3-43 |
| | atom$mat_bool_detached | 3-45 |
| | atom$mat_bool_dock_horizontal | 3-66 |
| | atom$mat_bool_dock_vertical | 3-51 |
| | atom$mat_bool_menu | 3-76 |
| | atom$mat_bool_palette | 3-83 |
| | atom$mat_bool_palette_art | 3-84 |
| | atom$mat_bool_tool_group | 3-97 |
| Displaying windows: placement | atom$mat_form_icon | 3-123 |
| | atom$mat_position | 3-147 |
| | atom$mat_style_id | 3-168 |

*Table 3-2: Attribute Manager Protocol Functions and Atoms (Continued)*

| Function | Atoms | Page |
|---|---|---|
| Sizing | atom$mat_bool_resize_horizontal | 3-91 |
| | atom$mat_bool_resize_vertical | 3-92 |
| Navigation | atom$mat_bool_modal | 3-77 |
| | atom$mat_bool_non_closeable | 3-80 |
| | atom$mat_dirty_query | 3-114 |
| Displaying scrollbars | atom$mat_bool_horizontal_scroll | 3-66 |
| | atom$mat_bool_vertical_scroll | 3-99 |
| Displaying menus | atom$mat_bool_dropdown_button | 3-56 |
| | atom$mat_bool_popup_menu | 3-86 |
| | atom$mat_popup_pfc_path | 3-146 |
| | atom$mat_popup_relative_id | 3-145 |
| Displaying objects | atom$mat_bool_default | 3-40 |
| | atom$mat_bool_disabled | 3-47 |
| | atom$mat_bool_draw_focus | 3-54 |
| | atom$mat_bool_gradual_shadow | 3-62 |
| | atom$mat_bool_modified | 3-78 |
| | atom$mat_bool_no_border | 3-79 |
| | atom$mat_bool_permanent | 3-85 |
| | atom$mat_frame_style | 3-124 |
| Placement | atom$mat_bool_hidden | 3-64 |
| | atom$mat_bool_invisible | 3-72 |
| | atom$mat_bool_precise | 3-87 |
| | atom$mat_horizontal_spacing | 3-127 |
| | atom$mat_orientation | 3-141 |
| | atom$mat_precise_x | 3-151 |
| | atom$mat_precise_y | 3-153 |
| | atom$mat_text_on_picture_pos | 3-171 |
| Sizing | atom$mat_bool_expand_to_fit | 3-58 |
| | atom$mat_height | 3-126 |
| | atom$mat_precise_height | 3-149 |
| | atom$mat_precise_width | 3-150 |
| | atom$mat_size | 3-165 |
| Color | atom$mat_bool_background_flood | 3-28 |
| | atom$mat_bool_background_pic | 3-29 |
| | atom$mat_color_face | 3-104 |
| | atom$mat_color_frame_hilight | 3-105 |
| | atom$mat_color_frame_shadow | 3-106 |

*Table 3-2: Attribute Manager Protocol Functions and Atoms (Continued)*

| Function | Atoms | Page |
|---|---|---|
| Trigger color | atom$mat_color_bottom_edge | 3-103 |
| | atom$mat_color_selected | 3-107 |
| | atom$mat_color_top_edge | 3-110 |
| List-box items | atom$mat_bool_contiguous | 3-38 |
| | atom$mat_bool_default_send | 3-41 |
| | atom$mat_bool_list_icons | 3-75 |
| | atom$mat_sort_order | 3-166 |
| URL history lists | atom$mat_bool_ignore_url_list | 3-68 |
| Tabbed pages | atom$mat_bool_page_control | 3-81 |
| | atom$mat_tab_get_cur_sel | 3-169 |
| | atom$mat_tab_set_cur_sel | 3-170 |
| Text views and text | atom$mat_bool_double_space | 3-53 |
| | atom$mat_bool_drop_at_top | 3-55 |
| | atom$mat_bool_encode_unicode | 3-57 |
| | atom$mat_bool_exportable | 3-59 |
| | atom$mat_bool_first_script | 3-60 |
| | atom$mat_bool_force_scroll | 3-61 |
| | atom$mat_bool_importable | 3-69 |
| | atom$mat_bool_language_popup | 3-73 |
| | atom$mat_bool_list_allow_entry | 3-74 |
| | atom$mat_bool_protected_input | 3-89 |
| | atom$mat_bool_url_sink | 3-98 |
| | atom$mat_bool_writeable | 3-100 |
| | atom$mat_capacity | 3-102 |
| | atom$mat_color_text | 3-108 |
| | atom$mat_color_text_shadow | 3-109 |
| | atom$mat_field_script | 3-116 |
| | atom$mat_font_id | 3-117 |
| | atom$mat_font_sis | 3-119 |
| | atom$mat_font_size | 3-121 |
| | atom$mat_font_style | 3-122 |
| | atom$mat_link_content_to_rid | 3-132 |
| | atom$mat_paragraph | 3-144 |
| | atom$mat_ruler | 3-157 |
| | atom$mat_scroll_threshold | 3-160 |
| | atom$mat_sink | 3-164 |
| Range fields and spin gadgets | atom$mat_increment | 3-128 |
| | atom$mat_minimum | 3-137 |

*Table 3-2: Attribute Manager Protocol Functions and Atoms (Continued)*

| Function | Atoms | Page |
|---|---|---|
| | atom$mat_maximum | 3-136 |
| Help bubbles | atom$mat_context_help | 3-113 |
| Defining shortcut keys | atom$mat_command_key | 3-111 |
| | atom$mat_shortcut_key | 3-163 |
| Transmitting field input data | atom$mat_secure_form | 3-162 |
| Spacing | atom$mat_left_spacing | 3-131 |
| | atom$mat_top_spacing | 3-174 |
| | atom$mat_right_spacing | 3-156 |
| | atom$mat_bottom_spacing | 3-101 |
| | atom$mat_spacing | 3-167 |
| Factory ID | atom$mat_factory_id | 3-115 |

The MAT protocol atoms are described in alphabetical order in the rest of this chapter.

# atom$mat_art_animation_rate

## 98 ($62)

**atom$mat_art_animation_rate** defines the rate at which animation occurs.

## Syntax

```
atom$mat_art_animation_rate <x>
```

| | |
|---|---|
| `<x>` | Specifies the interval of time in milliseconds that each animation frame appears. A value of **0** is the default, which specifies that the animation does not occur. |

## Object(s) Handled

The objects handled by this atom are `org_group` and `ind_group`.

## Return Value

None.

## Example

The following example defines the animation rate to 250 milliseconds:

```
atom$mat_start_object <trigger>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-2201>
atom$mat_precise_x <370>
atom$mat_precise_y <105>
atom$mat_art_animation_rate <250>
atom$mat_art_animation_seq <1 5 0 1>
atom$mat_repeat_animation <yes>
.
.
.
atom$mat_end_object
```

## atom$mat_art_animation_seq

### 99 ($63)

**atom$mat_art_animation_seq** determines which frames are displayed and the duration for which they are displayed. Animation frames are contained in the art specified using **atom$mat_art_id**.

### Syntax

```
atom$mat_art_animation_seq <f1 d1 [f2 d2]...[fn dn]>
```

| | |
|---|---|
| `<f1 d1>` | Specifies a 2-byte value that indicates a frame to display and the duration for which to display the frame.<br><br>The first byte is the frame number and the second byte is the duration in multiples of the rate specified using **atom$mat_animation_rate**. |
| `[f2 d2]...[fn dn]` | Specifies other optional 2-byte values to indicate other frames and durations that can be added in series. |

If no argument is specified for this atom, the default sequence occurs starting with frame 0 and ending with the last frame.

### Object(s) Handled

The objects handled by this atom are `org_group` and `ind_group`.

### Return Value

None.

### Example

The following example defines the animation sequence of two frames. Frame 1 occurs first and is set to a 1.25 second duration. Frame 0 occurs next for 250 milliseconds:

```
atom$mat_start_object <trigger>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-2201>
atom$mat_precise_x <370>
```

```
atom$mat_precise_y <105>
atom$mat_art_animation_rate <250>
atom$mat_art_animation_seq <1 5 0 1>
atom$mat_repeat_animation <yes>
.
.
.
atom$mat_end_object
```

## atom$mat_art_frame

**97 ($61)**

**atom$mat_art_frame** defines the frame number where animation begins or the frame number to display when not animating (for example, the three-icon window that appears when someone is connecting to the service). If this atom is not used, animation begins, by default, at the first frame (frame 0).

### Syntax

```
atom$mat_art_frame <x>
```

| | |
|---|---|
| `<x>` | Specifies the frame number. Values are **1** or greater. |

### Object(s) Handled

The objects handled by this atom are `org_group` and `ind_group`.

### Return Value

None.

### Example

The following example starts frame animation at frame number 1:

```
atom$man_start_object <trigger>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-2201>
atom$mat_art_animation_rate <250>
atom$mat_art_frame <1>
.
.
.
atom$man_end_object
```

# atom$mat_art_hint_font_size

## 128 ($80)

**atom$mat_art_hint_font_size** specifies the point size of the text for the placeholder (hint) artwork that appears during the progressive rendering of data on demand (DOD) artwork. This atom overrides any existing defaults set with other atoms, such as **atom$mat_font_sis**, and is always used in conjunction with atom$mat_art_hint_title on page 3-15. This atom is sent from the host and is useful for fitting text in a limited space inside the artwork.

### Syntax

```
atom$mat_art_hint_font_size <x>
```

| | |
|---|---|
| `<x>` | The point size of the text for the placeholder artwork. |

### Object(s) Handled

The objects handled by this atom are `trigger`, `ornament`, `org_group`, and `ind_group`.

### Return Value

None.

### Example

The following example defines the point size of the text for the placeholder artwork to 8:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-1234>
atom$mat_art_hint_width <x>
atom$mat_art_hint_height <y>
atom$mat_art_hint_title <“Travel Information”>
atom$mat_art_hint_font_size <8>
```

## atom$mat_art_hint_height

### 122 ($7A)

**atom$mat_art_hint_height** specifies the height of the placeholder (hint) artwork that appears during the progressive rendering of data on demand (DOD) artwork. This atom is sent by the host and is always used in conjunction with atom$mat_art_hint_width on page 3-21.

### Syntax

```
atom$mat_art_hint_height <y>
```

| | |
|---|---|
| `<y>` | The height of the artwork to be downloaded. Values are **1** or greater. A value of **1** is equal to the height of one pixel. |

### Object(s) Handled

The objects handled by this atom are `trigger`, `org_group`, and `ind_group`.

### Return Value

None.

### Example

The following example defines the height of the artwork to be downloaded as 20 pixels:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_hint_width <32>
atom$mat_art_hint_height <20>
atom$mat_art_hint_title <“Travel Information”>
atom$mat_art_id <1-0-1234>
```

# atom$mat_art_hint_placeholder_id

## 124 ($7C)

**atom$mat_art_hint_placeholder_id** sets the graphic ID (GID) of the art used as a placeholder before the real art is sent through a DOD.

### Syntax

```
atom$mat_art_hint_placeholder_id <x>
```

| | |
|---|---|
| `<x>` | The graphic ID (GID) of the art used as a placeholder. |

### Object(s) Handled

The objects handled by this atom are `trigger`, `ornament`, `org_group`, and `ind_group`.

### Return Value

None.

### Example

The following example sets the GID of the art used as a placeholder before the real art is sent through a DOD:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_hint_width <x>
atom$mat_art_hint_height <y>
atom$mat_art_hint_title “Travel Information”
atom$mat_art_hint_placeholder_id <1>
```

## atom$mat_art_hint_select_placeholder

**132 ($84)**

**atom$mat_art_select_placeholder** selects placeholder (hint) artwork from a list of canned placeholders that are defined within a table in the UDO (update disk operation). Placeholder artwork appears during the progressive rendering of data on demand (DOD) artwork. This atom is sent from the host.

### Syntax

```
atom$mat_art_hint_select_placeholder <placeholder_id>
```

| | |
|---|---|
| `<placeholder_id>` | The ID of the canned placeholder artwork. The default value is **0**, which specifies a default placeholder. |

### Object(s) Handled

The objects handled by this atom are `trigger`, `ornament`, `org_group`, and `ind_group`.

### Return Value

None.

### Example

The following example selects the default placeholder artwork for display during a progressive rendering, DOD operation:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_hint_width <x>
atom$mat_art_hint_height <y>
atom$mat_art_hint_title “Travel Information”
atom$mat_art_hint_select_placeholder <0>
atom$mat_art_id <1-0-1234>
```

# atom$mat_art_hint_title
## 123 ($7B)

**atom$mat_art_hint_title** defines the hint text associated with the placeholder (hint) artwork that appears during the progressive rendering of data on demand (DOD) artwork. The text wraps but must fit the dimensions specified with **atom$mat_art_hint_width** and **atom$mat_art_hint_height**; otherwise, the text does not appear. This atom is sent by the host.

### Syntax

```
atom$mat_art_hint_title <title>
```

| | |
|---|---|
| `<title>` | The hint text of the artwork to be downloaded. |

### Object(s) Handled

The objects handled by this atom are `trigger`, `ornament`, `org_group`, and `ind_group`.

### Return Value

None.

### Example

The following example defines "Travel Information" as the hint text of the artwork during a progressive rendering, DOD operation:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-1234>
atom$mat_art_hint_width <x>
atom$mat_art_hint_height <y>
atom$mat_art_hint_title <“Travel Information”>
```

## atom$mat_art_hint_title_font_id

**129 ($81)**

**atom$mat_art_hint_title_font_id** sets the ID of the DOD hint text.

### Syntax

```
atom$mat_art_hint_title_font_id <x>
```

| | |
|---|---|
| `<x>` | The ID of the DOD hint text. |

### Object(s) Handled

The objects handled by this atom are `trigger`, `ornament`, `org_group`, and `ind_group`.

### Return Value

None.

### Example

The following example sets the font ID of the DOD hint text:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-1234>
atom$mat_art_hint_width <64>
atom$mat_art_hint_height <64>
atom$mat_art_hint_title_x <2>
atom$mat_art_hint_title_y <2>
atom$mat_art_hint_title <“Travel Information”>
atom$mat_art_hint_title_font_id <1>
```

## atom$mat_art_hint_title_font_size

**128 ($80)**

**atom$mat_art_hint_title_font_size** sets the font size of the DOD hint text.

#### Syntax

```
atom$mat_art_hint_title_font_size <x>
```

| | |
|---|---|
| `<x>` | The font size of the title of the placeholder artwork. Values are **1** or greater. |

### Object(s) Handled

The objects handled by this atom are `trigger`, `ornament`, `org_group`, and `ind_group`.

### Return Value

None.

### Example

The following example sets the title font size as eight:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-1234>
atom$mat_art_hint_width <64>
atom$mat_art_hint_height <64>
atom$mat_art_hint_title_x <2>
atom$mat_art_hint_title_y <2>
atom$mat_art_hint_title <“Travel Information”>
atom$mat_art_hint_title_font_id <1>
atom$mat_art_hint_title_font_size <8>
```

## atom$mat_art_hint_title_font_style

**130($82)**

**atom$mat_art_hint_title_font_style** sets the font style of the DOD hint text. See atom$mat_font_id on page 3-117 for the list of font styles.

### Syntax

```
atom$mat_art_hint_font_style_x <x>
```

| | |
|---|---|
| `<x>` | The font style of the title of the placeholder artwork. Values are **1** or greater. |

### Object(s) Handled

The objects handled by this atom are `trigger`, `ornament`, `org_group`, and `ind_group`.

### Return Value

None.

### Example

The following example sets the title font style to courier:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-1234>
atom$mat_art_hint_width <64>
atom$mat_art_hint_height <64>
atom$mat_art_hint_title_x <2>
atom$mat_art_hint_title_y <2>
atom$mat_art_hint_title <“Travel Information”>
atom$mat_art_hint_title_font_id <1>
atom$mat_art_hint_title_font_size <8>
atom$mat_art_hint_title_font_style <1>
```

# atom$mat_art_hint_title_x

## 125 ($7D)

**atom$mat_art_hint_title_x** defines the horizontal (x-axis) location of the title of the placeholder (hint) artwork that appears during the progressive rendering of data on demand (DOD) artwork.

This atom is optional and is used with atom$mat_art_hint_title_y on page 3-20. If specified, it can be used to cause the text of the title to overlay the artwork. If this atom is not specified, the title appears in the location where it fits best in relation to the artwork, namely, first in the lower right corner, then the upper right corner, and then across the bottom. This atom is sent by the host.

### Syntax

```
atom$mat_art_hint_title_x <x>
```

| | |
|---|---|
| `<x>` | The horizontal pixel position of the title of the placeholder artwork. Values are **1** or greater. |

## Object(s) Handled

The objects handled by this atom are `trigger`, `ornament`, `org_group`, and `ind_group`.

## Return Value

None.

## Example

The following example defines the title of the placeholder artwork to be two pixels across from the upper left corner:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-1234>
atom$mat_art_hint_width <64>
atom$mat_art_hint_height <64>
atom$mat_art_hint_title_x <2>
atom$mat_art_hint_title_y <2>
atom$mat_art_hint_title <“Travel Information”>
```

## atom$mat_art_hint_title_y

### 127 ($7F)

**atom$mat_art_hint_title_y** defines the vertical (y-axis) location of the title of the placeholder (hint) artwork that appears during the progressive rendering of data on demand (DOD) artwork.

This atom is optional and is used with atom$mat_art_hint_title_font_id on page 3-16. If specified, it can be used to cause the text of the title to overlay the artwork. If this atom is not specified, the title appears in the location where it fits best in relation to the artwork, namely first in the lower right corner, then the upper right corner, and then across the bottom. This atom is sent by the host.

### Syntax

```
atom$mat_art_hint_title_y <y>
```

| | |
|---|---|
| `<y>` | The vertical pixel position of the title of the placeholder artwork. Values are **1** or greater. |

#### Object(s) Handled

The objects handled by this atom are `trigger`, `ornament`, `org_group`, and `ind_group`.

#### Return Value

None.

#### Example

The following example defines the title of the placeholder artwork to be two pixels down from the upper left corner:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-1234>
atom$mat_art_hint_width <64>
atom$mat_art_hint_height <64>
atom$mat_art_hint_title_x <2>
atom$mat_art_hint_title_y <2>
atom$mat_art_hint_title, “Travel Information”>
```

# atom$mat_art_hint_width

**121 ($79)**

**atom$mat_art_hint_width** defines the width of the placeholder (hint) artwork that appears during the progressive rendering of data on demand (DOD) artwork. This atom is sent by the host and is always used in conjunction with atom$mat_art_hint_height on page 3-12.

## Syntax

```
atom$mat_art_hint_width <x>
```

| | |
|---|---|
| `<x>` | The width of the artwork to be downloaded. Values are **1** or greater. A value of **1** is equal to the width of one pixel. |

## Object(s) Handled

The objects handled by this atom are `trigger`, `ornament`, `org_group`, and `ind_group`.

## Return Value

None.

## Example

The following example determines the width (32 pixels) of the artwork to be downloaded:

```
atom$man_start_object <trigger, “”>
atom$mat_relative_tag <1>
atom$mat_art_hint_width <32>
atom$mat_art_hint_height <20>
atom$mat_art_hint_title <“title”>
atom$mat_art_id <1-0-1234>
```

## atom$mat_art_id

### 14 ($0E)

**atom$mat_art_id** defines the global ID of the art piece associated with the object you are creating. The format for an art ID is <1-byte, 1-byte, 2-bytes>. Local art IDs are specified as <255-x-x>, which are stored and sent from the client database, not the host.

### Syntax

```
atom$mat_art_id <art_ID>
```

| | |
|---|---|
| `<art_ID>` | Specifies the global ID of the art. The default value is **0**, which specifies that no art is associated with the object. |

### Object(s) Handled

The objects handled by this atom are `org_group` and `ind_group`.

### Return Value

The art ID.

### Example

The following example assigns an art ID to a button:

```
atom$man_start_object <trigger>
atom$mat_art_id <1-0-1234>
```

# atom$mat_art_seq

## 183 ($87)

**atom$mat_art_seq** specifies frames of art to be used for various trigger states. This atom is used in the 5.0 clients primarily to provide rollover button functionality across multiple trigger styles.

### Syntax

```
atom$mat_art_seq <x>
```

| | |
|---|---|
| `<x>` | Specifies the art frame to be used for mouseup, rollover, disabled, and mousedown states, in that order, through a series of up to four bytes. If a particular frame is not specified, the mouseup frame is used in that position. |

### Object(s) Handled

The objects handled by this atom are `org_group` and `ind_group`.

### Return Value

None.

### Example

```
atom$man_start_object <trigger>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-2201>
atom$mat_art_animation_rate <250>
atom$mat_art_seq <1>
.
.
.
atom$man_end_object
```

## atom$mat_auto_complete

### 185 ($B9)

**atom$mat_art_id** enables Netscape-like autocomplete for edit boxes. The edit box performs autocomplete; the atom is left open ended to allow other controls to search history lists using search algorithms.

### Syntax

```
atom$mat_auto_complete <x, y>
```

| | |
|---|---|
| `<x>` | Specifies the `search list`, where x is:<br><br>`0` - Use web/search words/keywords history list<br><br>`1` - Use mail address book search list<br><br>`2` - Use other search list |
| `<y>` | Specifies the `<search_method>`, where y is:<br><br>`0` - Use web search to ignore http://www. during matches<br><br>`1` - Use standard sorted search |

### Example

The following example tells the display manager to use the web/search words/keywords history list and tells the web search to ignore http://www. during matches:

```
atom$man_set_context_relative <2>
atom$mat_auto_complete <0, 0>
atom$man_end_context
```

## atom$mat_bool_active_offline
### 157 ($9D)

**atom$mat_bool_active_offline** determines whether the current object is enabled when the member is offline.

### Syntax

```
atom$mat_bool_active_offline <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the current object is enabled when the member is offline. Values are: | |
| | Yes | This object is enabled when the member is offline (default). |
| | No | This object is not enabled when the member is offline. |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

None.

### Example

The following example makes the current `trigger` object (AOL icon) inactive when the member is offline:

```
atom$mat_object_type <trigger, “”>
atom$mat_trigger_style <plain_picture>
atom$mat_relative_tag <61>
atom$mat_bool_palette_art <yes>
atom$mat_horizontal_spacing <1>
atom$mat_vertical_spacing <0>
atom$mat_bool_active_offline <no>
atom$mat_bool_child_movable <no>
atom$mat_context_help <”Go to the AOL home page.”>
```

## atom$mat_bool_active_online

### 158 ($9E)

**atom$mat_bool_active_online** determines whether the current object is enabled when the member is online.

### Syntax

```
atom$mat_bool_active_online <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the current object is enabled when the member is offline. Values are: | |
| | Yes | This object is enabled when the member is offline (default). |
| | No | This object is not enabled when the member is offline. |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

None.

### Example

The following example makes the current trigger object (AOL icon) active when the member is online:

```
atom$mat_object_type <trigger, “”>
atom$mat_trigger_style <plain_picture>
atom$mat_relative_tag <61>
atom$mat_bool_palette_art <yes>
atom$mat_horizontal_spacing <1>
atom$mat_vertical_spacing <0>
atom$mat_bool_active_online <yes>
atom$mat_bool_child_movable <no>
atom$mat_context_help <”Go to the AOL home page.”>
```

# atom$mat_bool_auto_closeable

## 172 ($AC)

**atom$mat_bool_auto_closeable** identifies which forms can or cannot be closed when the total number of forms opened has reached a limit and the client needs to close some of the open forms. Only a dozen forms (added with **atom$mat_bool_auto_closeable <no>**) are expected to be not automatically closeable. By default, all forms are automatically closeable. See also: atom$mat_bool_non_closeable on page 3-80.

### Syntax

```
atom$mat_bool_auto_closeable <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the current independent window object can be automatically closed when the client window limit has been reached. Attribute is ignored on nonindependent windows. Values are: |

| | |
|---|---|
| Yes | This independent window can be closed automatically (default). |
| No | This independent window cannot be closed automatically. |

### Object(s) Handled

Independent window.

### Return Value

Unchanged.

### Example

```
atom$man_start_object <ind_group, "non-auto-close-form">
...
atom$mat_bool_auto_closeable <no>
...
atom$man_end_object
```

## atom$mat_bool_background_flood

### 94 ($5E)

**atom$mat_bool_background_flood** fills the background of the current object with a color. The color is specified using **atom$mat_color_face**. This atom is used with groups, windows, or certain views (namely, list boxes and static text). It cannot be used with editable views.

#### Syntax

```
atom$mat_bool_background_flood <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the background of an object is flooded with color. Values are: | |
| | Yes | The background of an object is flooded with color. |
| | No | The background of an object is not flooded with color (default). |

### Object(s) Handled

The objects handled by this atom are `org_group`, `ind_group`, and `view`.

### Return Value

None.

### Example

The following example fills the background of the window titled "Background Flooded Window" with the color blue:

```
atom$man_start_object <ind_group, "Background Flooded
Window">
atom$mat_bool_background_flood <yes>
atom$mat_color_face <0 0 80>
```

## atom$mat_bool_background_pic
**93 ($5D)**

**atom$mat_bool_background_pic** fills the background of the current object with a picture. A picture must be specified using atom$mat_art_id on page 3-22.

### Syntax

```
atom$mat_bool_background_pic <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the background of an object is flooded with color. Values are: | |
| | Yes | The background of an object is flooded with color. |
| | No | The background of an object is not flooded with color (default). |

### Return Value

None.

### Object(s) Handled

The objects handled by this atom are `org_group` and `ind_group`.

### Example

The following example fills the background of the window titled "Background Flooded Window" with the piece of art with an ID of 1-0-2196:

```
atom$man_start_object <ind_group, "Background Flooded Window">
atom$mat_bool_background_pic <yes>
atom$mat_art_id <1-0-2196>
```

## atom$mat_bool_background_tile

### 101 ($65)

**atom$mat_bool_background_tile** fills the background of the current object with a picture and arranges it in a tiled formation. A picture must be specified using atom$mat_art_id on page 3-22.

### Syntax

```
atom$mat_bool_background_tile <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the background picture is tiled. Values are: | |
| | Yes | The background picture is tiled. |
| | No | The background picture is not tiled (default). |

### Object(s) Handled

The objects handled by this atom are `org_group` and `ind_group`.

### Return Value

None.

### Example

The following example fills the background of a window titled "Layered Backgrounds" with the piece of art with an ID of 1-0-2207; the art is arranged in a tiled formation:

```
atom$man_start_object <ind_group, "Layered Backgrounds">
atom$mat_bool_background_tile <yes>
atom$mat_art_id <1-0-2207>
```

# atom$mat_bool_child_line_feed

## 151 ($97)

**atom$mat_bool_child_line_feed** forces a child object to a new line (row) in the toolbar or parent object. Typically, it is used to start a new row of buttons on a toolbar.

### Syntax

```
atom$mat_bool_child_line_feed <boolean>
```

`<boolean>` Specifies whether to start a new line with the current child object. Values are:

| | |
|---|---|
| Yes | Start a new line with this child. |
| No | Do not start a new line with this child (default). |

### Object(s) Handled

The objects handled by this atom are children objects of `tool_group`.

### Return Value

None.

### Example

The following example forces the third button (child object) on a toolbar to the next line (row) of buttons:

```
atom$uni_start_stream
atom$man_start_object <ind_group, "">
atom$man_start_object <tool_group, "">
atom$mat_orientation <hff>
atom$mat_frame_style <4>
atom$man_start_object <trigger, “Do”>
atom$mat_relative_tag <1>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the first button”>
      atom$uni_end_stream
atom$man_end_object
atom$man_start_object <trigger, “Undo”>
atom$mat_relative_tag <2>
atom$act_replace_select_action
```

```
      atom$uni_start_stream
      atom$async_error_box <”Action for the second button”>
      atom$uni_end_stream
atom$man_end_object
atom$man_start_object <trigger, “Go To”>
atom$mat_bool_child_line_feed <yes>
atom$mat_relative_tag <3>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the third button”>
      atom$uni_end_stream
atom$man_end_object
atom$man_end_object
```

## atom$mat_bool_child_movable

**150 ($96)**

**atom$mat_bool_child_movable** determines whether the current child object is movable in a toolbar or parent window.

### Syntax

```
atom$mat_bool_child_movable <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the current child object can be moved in the toolbar or parent window. Values are: |

| | |
|---|---|
| Yes | This object can be moved in the parent window (default). |
| No | This object cannot be moved in the parent window. |

### Object(s) Handled

The objects handled by this atom are children objects of `tool_group`.

### Return Value

None.

### Example

The following example makes the current child object (AOL icon) fixed in the toolbar so that it cannot be moved:

```
atom$mat_object_type <trigger, “”>
atom$mat_trigger_style <plain_picture>
atom$mat_relative_tag <61>
atom$mat_bool_palette_art <yes>
atom$mat_horizontal_spacing <1>
atom$mat_vertical_spacing <0>
atom$mat_bool_active_online <yes>
atom$mat_bool_child_movable <no>
atom$mat_context_help <”Go to the AOL home page.”>
```

## atom$mat_bool_child_removable

**149 ($95)**

**atom$mat_bool_child_removable** determines whether the current child object is removable from a toolbar or parent window.

### Syntax

```
atom$mat_bool_child_removable <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the current child object can be removed from the toolbar or parent window. Values are: | |
| | Yes | This object can be removed from the parent window (default). |
| | No | This object cannot be removed from the parent window. |

### Object(s) Handled

The objects handled by this atom are children objects of `tool_group`.

### Return Value

None.

### Example

The following example makes the child object (button) removable from the toolbar:

```
atom$man_start_object <ind_group, "">
atom$man_start_object <tool_group, "">
atom$mat_bool_detached <yes>
atom$mat_orientation <hff>
atom$mat_frame_style <4>
atom$man_start_object <trigger, “Do”>
atom$mat_bool_child_removable <yes>
atom$mat_relative_tag <1>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the first button”>
      atom$uni_end_stream
atom$man_end_object
```

# atom$mat_bool_children_movable

**148 ($94)**

**atom$mat_bool_children_movable** determines whether children objects are movable in the toolbar (or current object).

## Syntax

```
atom$mat_bool_children_movable <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether children can be moved in the toolbar (or current object). Values are: | |
| | Yes | The children can be moved in the current object (default). |
| | No | The children cannot be moved in the current object. |

## Object(s) Handled

The object handled by this atom is `tool_group`.

## Return Value

None.

## Example

The following example makes the children objects (buttons) movable in the toolbar:

```
atom$man_start_object <ind_group, "">
atom$man_start_object <tool_group, "">
atom$mat_bool_children_movable <yes>
atom$mat_orientation <hff>
atom$mat_frame_style <4>
atom$man_start_object <trigger, “Do”>
atom$mat_relative_tag <1>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the first button”>
      atom$uni_end_stream
atom$man_end_object
atom$man_start_object <trigger, “Undo”>
atom$mat_relative_tag <2>
atom$act_replace_select_action
```

```
            atom$uni_start_stream
            atom$async_error_box <”Action for the second button”>
            atom$uni_end_stream
      atom$man_end_object
```

# atom$mat_bool_children_removable

## 147 ($93)

**atom$mat_bool_children_removable** determines whether children objects are removable from the toolbar (or current object).

## Syntax

```
atom$mat_bool_children_removable <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether children can be removed from the toolbar (or current object). Values are: | |
| | Yes | The children can be removed from the current object (default). |
| | No | The children cannot be removed from the current object. |

## Object(s) Handled

The object handled by this atom is `tool_group`.

## Return Value

None.

## Example

The following example makes children objects (buttons) removable from the toolbar:

```
atom$uni_start_stream
atom$man_start_object <ind_group, "">
atom$man_start_object <tool_group, "">
atom$mat_bool_children_removable <yes>
atom$mat_orientation <hff>
atom$mat_frame_style <4>
atom$man_start_object <trigger, “Do”>
atom$mat_relative_tag <1>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the first button”>
      atom$uni_end_stream
atom$man_end_object
```

## atom$mat_bool_contiguous

**41 ($29)**

**atom$mat_bool_contiguous** forces all items in a dynamic multiselect list (`dms_list`) to be contiguous. When a member selects (for example, highlights) the second and fourth items in a list, the third item becomes highlighted as well.

### Syntax

```
atom$mat_bool_contiguous <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the contiguous feature for list items that are multiselectable is in use. Values are: |

| | |
|---|---|
| Yes | The contiguous feature for list items is in use. |
| No | The contiguous feature for list items is not in use (default). |

### Object(s) Handled

The object handled by this atom is `dms_list`.

### Return Value

None.

### Example

The following example makes the items in a dynamic, multiselect list contiguous:

```
atom$man_start_object <dms_list>
atom$mat_bool_contiguous <yes>
```

# atom$mat_bool_customizable

## 143 ($8F)

**atom$mat_bool_customizable** determines whether the toolbar or parent window (current object) and its children objects can be moved and modified.

### Syntax

```
atom$mat_bool_customizable <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the current object (toolbar or parent window) and its children objects can be moved and modified. Values are: |

| | |
|---|---|
| Yes | The object and children can be moved and modified (default). |
| No | The object and children cannot be modified. |

### Object(s) Handled

The object handled by this atom is `tool_group`.

### Return Value

None.

### Example

The following example establishes the toolbar and its children as customizable so they can be moved and modified:

```
atom$uni_start_stream
atom$man_start_object <ind_group, "">
atom$man_start_object <tool_group, "">
atom$mat_bool_customizable <yes>
atom$mat_orientation <hff>
atom$mat_frame_style <4>
atom$man_start_object <trigger, “Do”>
atom$mat_relative_tag <1>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the first button”>
      atom$uni_end_stream
atom$man_end_object
```

## atom$mat_bool_default

**2 ($02)**

**atom$mat_bool_default** defines a button or icon (trigger objects only) on a form as the default action. The default button is the one whose action is executed when a member presses ENTER. Only one trigger can be specified for any independent group (ind_group) object.

### Syntax

```
atom$mat_bool_default <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the object is the default. Values are: | |
| | Yes | The object is the default. |
| | No | The object is not the default (default). |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

None.

### Example

The following example defines a default button:

```
atom$man_start_object <trigger>
atom$mat_bool_default <yes>
```

# atom$mat_bool_default_send

**43 ($2B)**

**atom$mat_bool_default_send** sends an item on a window to the host server that, otherwise, is not sent or blocks the transmission mode of an item to the host. Also, this atom can override the behavior of **atom$de_ez_send_form**, which sends data from any children of the current object to the host.

## Syntax

```
atom$mat_bool_default_send <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the object on the window is sent to the host when the `trigger` object sending the window's data is selected by a member. Values are: |

| | |
|---|---|
| Yes | The object on the window is sent to the host. This is the default value when the current object is `dms_list` or edit_view. |
| No | The object on the window is not sent to the host. This is the default value when the current object is `dss_list`. |

## Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, and `edit_view`.

## Return Value

None.

## Example

The following example overrides the default behavior of the **atom$de_ez_send_form** atom:

```
atom$man_start_object <dss_list>
atom$mat_bool_default_send <no>
```

**Note**: The atom stream that defines a window's object always precedes the atom stream containing the **atom$de_ez_send_form** command, which sends the form's data.

# atom$mat_bool_detachable

**144 ($90)**

**atom$mat_bool_detachable** determines whether the toolbar or current object is detachable from the parent window.

## Syntax

```
atom$mat_bool_detachable <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the toolbar or current object is detachable from the parent window. Values are: | |
| | Yes | The object is detachable from the parent (default). |
| | No | The object is not detachable from the parent. |

## Object(s) Handled

The object handled by this atom is `tool_group`.

## Return Value

None.

## Example

The following example makes a toolbar detachable from the parent window:

```
atom$uni_start_stream
atom$man_start_object <ind_group, "">
atom$man_start_object <tool_group, "">
atom$mat_bool_detachable <yes>
atom$mat_orientation <hff>
atom$mat_frame_style <4>
atom$man_start_object <trigger, “Do”>
atom$mat_relative_tag <1>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the first button”>
      atom$uni_end_stream
atom$man_end_object
atom$man_start_object <trigger, “Undo”>
atom$mat_relative_tag <2>
atom$act_replace_select_action
```

```
            atom$uni_start_stream
            atom$async_error_box <”Action for the second button”>
            atom$uni_end_stream
      atom$man_end_object
      atom$man_end_object
```

# atom$mat_bool_detached

**142 ($8E)**

**atom$mat_bool_detached** detaches the toolbar or current object from the parent window.

## Syntax

```
atom$mat_bool_detached <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the toolbar or current object is detached from the parent window. Values are: | |
| | Yes | The object is detached from the parent. |
| | No | The object is not detached from the parent (default). |

## Object(s) Handled

The object handled by this atom is `tool_group`.

## Return Value

None.

## Example

The following example detaches a toolbar from the parent window:

```
atom$uni_start_stream
atom$man_start_object <ind_group, "">
atom$man_start_object <tool_group, "">
atom$mat_bool_detached <yes>
atom$mat_orientation <hff>
atom$mat_frame_style <4>
atom$man_start_object <trigger, “Do”>
atom$mat_relative_tag <1>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the first button”>
      atom$uni_end_stream
atom$man_end_object
atom$man_start_object <trigger, “Undo”>
atom$mat_relative_tag <2>
atom$act_replace_select_action
      atom$uni_start_stream
```

```
            atom$async_error_box <”Action for the second button”>
            atom$uni_end_stream
      atom$man_end_object
      atom$man_end_object
```

# atom$mat_bool_disabled

## 1 ($01)

**atom$mat_bool_disabled** specifies an object as nonselectable on a form and makes the object appear dim or unhighlighted. This atom is commonly used on a `trigger` object.

An example of the use of this command is the **Send** button on the Compose Mail window. When the member is offline, that button is disabled because mail can be composed offline but not sent. Once the member logs on to the online service, that button is no longer disabled.

### Syntax

```
atom$mat_bool_disabled <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the object on the window is selectable. Values are: |

| | |
|---|---|
| Yes | The object on the window is not selectable. This is the default value when the current object is `dms_list`, `dss_list`, or edit_view. |
| No | The object on the window is selectable. This is the default value when the current object is `org_group`, `ind_group`, `sms_list`, `sss_list`, `trigger`, `ornament`, `view`, `range`, or `select_range`. |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example inhibits the action of object number 4 in the form:

```
atom$man_start_object <trigger>
atom$mat_relative_tag <4>
.
```

```
.
.
atom$man_set_context_globalid <32-123>
atom$man_set_context_relative <4>
atom$mat_bool_disabled <yes>
```

# atom$mat_bool_dock_horizontal

**145 ($91)**

**atom$mat_bool_dock_horizontal** determines whether the toolbar or current object can be attached to the top and bottom edges of the parent window.

## Syntax

```
atom$mat_bool_dock_horizontal <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the toolbar or current object can be attached to the top and bottom edges of the parent window. Values are: | |
| | Yes | The object can be attached to the top and bottom edges of the parent window (default). |
| | No | The object cannot be attached to the top and bottom edges of the parent window. |

## Object(s) Handled

The object handled by this atom is `tool_group`.

## Return Value

None.

## Example

The following example makes the toolbar attachable to the top and bottom edges of the parent window:

```
atom$uni_start_stream
atom$man_start_object <ind_group, "">
atom$man_start_object <tool_group, "">
atom$mat_bool_dock_horizontal <yes>
atom$mat_orientation <hff>
atom$mat_frame_style <4>
atom$man_start_object <trigger, “Do”>
atom$mat_relative_tag <1>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the first button”>
      atom$uni_end_stream
```

```
atom$man_end_object
atom$man_start_object <trigger, “Undo”>
atom$mat_relative_tag <2>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the second button”>
      atom$uni_end_stream
atom$man_end_object
atom$man_end_object
```

# atom$mat_bool_dock_vertical

**146 ($92)**

**atom$mat_bool_dock_vertical** determines whether the toolbar or current object can be attached to the left and right edges of the parent window.

## Syntax

```
atom$mat_bool_dock_vertical <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the toolbar or current object can be attached to the left and right edges of the parent window. Values are: | |
| | Yes | The object can be attached to the left and right edges of the parent window (default). |
| | No | The object cannot be attached to the left and right edges of the parent window. |

## Object(s) Handled

The object handled by this atom is `tool_group`.

## Return Value

None.

## Example

The following example makes the toolbar attachable to the left and right edges of the parent window:

```
atom$uni_start_stream
atom$man_start_object <ind_group, "">
atom$man_start_object <tool_group, "">
atom$mat_bool_dock_vertical <yes>
atom$mat_orientation <hff>
atom$mat_frame_style <4>
atom$man_start_object <trigger, “Do”>
atom$mat_relative_tag <1>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the first button”>
      atom$uni_end_stream
atom$man_end_object
```

```
atom$man_start_object <trigger, “Undo”>
atom$mat_relative_tag <2>
atom$act_replace_select_action
      atom$uni_start_stream
      atom$async_error_box <”Action for the second button”>
      atom$uni_end_stream
atom$man_end_object
atom$man_end_object
```

# atom$mat_bool_double_space

## 44 ($2C)

**atom$mat_bool_double_space** selects double-spacing for text view objects. The double-spacing feature is used in the chat area of the online service. It is available as a member preference in the chat rooms.When "Double-space incoming messages" is selected on the Chat Preferences menu, the data in the text view object displaying member chat conversation appears in double-spaced format.

### Syntax

```
atom$mat_bool_double_space <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether double-spacing is selected for the text view. Values are: |

| | |
|---|---|
| Yes | Double-spacing is selected. |
| No | Double-spacing is not selected (default). |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example selects double-spacing for a view object:

```
atom$man_start_object <view>
atom$mat_bool_double_space <yes>
```

## atom$mat_bool_draw_focus

### 165 ($A5)

**atom$mat_bool_draw_focus** draws a focus rectangle (when the button is in focus) on the current button (`trigger`) object. Normally, focus rectangles for trigger styles 3, 6, 8, and 9 are not drawn on the button for aesthetic reasons.

### Syntax

```
atom$mat_bool_draw_focus <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether a focus rectangle is drawn on the `trigger` object. Values are: | |
| | Yes | The focus rectangle is drawn on the button. |
| | No | The focus rectangle is not drawn on the button (default). |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

None.

### Example

The following example draws a focus rectangle around a pictured button:

```
atom$man_start_sibling <trigger, "">
atom$mat_precise_x <10>
atom$mat_precise_y <75>
atom$mat_precise_width <174>
atom$mat_precise_height <50>
atom$mat_trigger_style <plain_picture>
atom$mat_art_id <1-0-30871>
atom$mat_bool_draw_focus <yes>
```

## atom$mat_bool_drop_at_top

### 114 ($72)

**atom$mat_bool_drop_at_top** determines whether mail dragged from the current object is dropped at the top of the Favorite Places list of the Personal Filing Cabinet (PFC) of the online service. The PFC is used to organize and maintain objects such as mail documents, download files, and Favorite Places (bookmarks).

### Syntax

```
atom$mat_bool_drop_at_top <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether mail is dropped at the top of the Favorite Places list. Values are: |

| | |
|---|---|
| Yes | Mail is dropped at the top of the list. |
| No | Mail is not dropped at the top of the list (default). |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example specifies that mail be dropped at the top of the list:

```
atom$man_start_object <ind_group>
atom$mat_bool_drop_at_top <yes>
atom$man_end_object
```

## atom$mat_bool_dropdown_button

**160 ($A0)**

**atom$mat_bool_dropdown_button** gives a button (trigger) the attribute of triggering a drop-down listbox. Use **atom$mat_bool_dropdown_button** when defining the function of the toolbar buttons.

### Syntax

```
atom$mat_bool_dropdown_button <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the current object is a standard menu or a popup menu triggered from a drop-down button. Values are: |

| | |
|---|---|
| Yes | The object is a popup menu triggered from a drop-down button. |
| No | The object is not a popup menu triggered from a drop-down button (default). |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

None.

### Example

The following example defines the menu object as a drop-down menu that is triggered from the button:

```
atom$man_start_object <trigger, "DropDown Button">
atom$mat_bool_dropdown_button> <yes>
atom$act_replace_select_action
atom$uni_start_stream
atom$man_set_context_relative <10>
atom$man_display_popup_menu
atom$man_end_context
atom$uni_end_stream
```

## atom$mat_bool_encode_unicode

### 134 ($86)

**atom$mat_bool_encode_unicode** is described in Appendix B, "Internationalization (i18n) Atoms."

## atom$mat_bool_expand_to_fit
**110 ($6E)**

**atom$mat_bool_expand_to_fit** expands an object's minor axis to make it fit into a specific area of a form.

### Syntax

```
atom$mat_bool_expand_to_fit <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether to expand an object's minor axis. Values are: | |
| | Yes | Expand the object. |
| | No | Do not expand the object (default). |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example expands a view object to fit into a specific area of a form:

```
atom$man_start_object <org_group>
atom$man_start_object <view>
atom$mat_bool_expand_to_fit <yes>
.
.
.
atom$man_end_object
```

## atom$mat_bool_exportable

### 37 ($25)

**atom$mat_bool_exportable** makes the data extraction feature available that lets a member copy or cut data (using an Edit menu) from one text view object to another.

### Syntax

```
atom$mat_bool_exportable <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the cut or copy text feature between objects is available. Values are: |

| | |
|---|---|
| Yes | The cut or copy text feature is available (default). |
| No | The cut or copy text feature is not available. |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example inhibits the cutting or copying of text in a view:

```
atom$man_start_object <view>
atom$mat_bool_exportable <no>
```

## atom$mat_bool_first_script

### 106 ($6A)

**atom$mat_bool_first_script** is described in Appendix B, "Internationalization (i18n) Atoms."

## atom$mat_bool_force_scroll
### 33 ($21)

**atom$mat_bool_force_scroll** selects automatic scrolling as data is added to the text view object. The automatic scrolling feature is used in the chat rooms of the online service. It is used in the text view object where member chat conversation appears. As members chat back and forth, the data automatically scrolls upward. If force scroll is not on, the member must manually scroll down the text view by clicking the down arrow on the scroll bar.

### Syntax

```
atom$mat_bool_force_scroll <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether automatic scrolling is selected. Values are: |

| | |
|---|---|
| Yes | Automatic scrolling is selected. |
| No | Automatic scrolling is not selected (default). |

### Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `view`, and `edit_view`.

### Return Value

None.

### Example

The following example selects automatic scrolling for a view:

```
atom$man_start_object <view>
atom$mat_bool_force_scroll <yes>
```

## atom$mat_bool_gradual_shadow

**86 ($56)**

**atom$mat_bool_gradual_shadow** creates a shadowing effect for the current `trigger` object that blends in with its background. This atom works only with video modes that are running at least 65,000 colors. This atom has no effect if the trigger style type is set to `trigger_style_framed` or `trigger_style_plain_pict`.

### Syntax

```
atom$mat_bool_gradual_shadow <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether gradual shadowing is selected. Values are: | |
| | Yes | Gradual shadowing is selected. |
| | No | Gradual shadowing is not selected (default). |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

None.

### Example

The following example selects gradual shadowing for an object:

```
atom$man_start_object <trigger>
atom$mat_relative_tag <1>
atom$mat_trigger_style <2>
atom$mat_art_id <1-0-2199>
atom$mat_precise_x <370>
atom$mat_precise_y <10>
atom$mat_bool_gradual_shadow <yes>
atom$man_end_object
```

# atom$mat_bool_graphic_view
## 27 ($1B)

**atom$mat_bool_graphic_view** defines the current object as a graphic view.

### Syntax

```
atom$mat_bool_graphic_view <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the object is a graphic view. Values are: | |
| | Yes | The object is a graphic view. |
| | No | The object is not a graphic view (default). |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example defines a view as a graphic display:

```
atom$man_start_object <view>
atom$mat_bool_graphic_view <yes>
```

## atom$mat_bool_hidden

### 7 ($07)

**atom$mat_bool_hidden** hides an object on a window and reserves the space where that object would have appeared. When an object is hidden, a blank space appears instead of the object.

For example, five buttons exist on a form as part of a horizontal group and are centered. If the middle button is hidden, the remaining four buttons remain in their positions as if the middle button were still there. A gap of blank space appears in the location of the hidden button.

See atom$mat_bool_invisible on page 3-72 for similar functionality.

### Syntax

```
atom$mat_bool_hidden <boolean>
```

`<boolean>` Specifies whether the object is hidden. Values are:

| | |
|---|---|
| Yes | The object is hidden. |
| No | The object is not hidden (default). |

### Object(s) Handled

The objects handled by this atom are `org_group`, `dms_list`, `dss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

### Return Value

None.

### Example

The following example hides an object on a window:

```
atom$man_start_object <trigger>
atom$mat_relative_tag <2>
atom$mat_title <"List Responses">
atom$mat_bool_hidden <yes>
```

# atom$mat_bool_hide_url
## 171 ($AB)

**atom$mat_bool_hide_url** hides an URL from the history list, and it cannot be seen or edited in favorite places.

## Syntax

```
atom$mat_bool_hide_url <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the object is hidden. Values are: | |
| | Yes | The URL does not show in the history list and cannot be seen or edited in favorite places. |
| | No | The URL shows in the history list and can be seen or edited in favorite places (default). |

## Object(s) Handled

The object handled by this atom is `ind_group`.

## Return Value

Unchanged.

## Example

The following example hides the URL from the history list, and it cannot be seen or edited in favorite places:

```
atom$man_start_object
atom$mat_bool_hide_url <yes>
atom$man_end_object
```

## atom$mat_bool_horizontal_scroll

**32 ($20)**

**atom$mat_bool_horizontal_scroll** selects a horizontal scroll bar when a view object is displayed on a form.

This feature is used when a **.gif** (GIF) file, displayed using the online service GIF viewer, is wider than the space available to display it.

### Syntax

```
atom$mat_bool_horizontal_scroll <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the horizontal scroll bar is selected. Values are: | |
| | Yes | The horizontal scroll bar is selected. |
| | No | The horizontal scroll bar is not selected (default). |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example selects a horizontal scroll bar for the view object:

```
atom$man_start_object <edit_view>
atom$mat_bool_graphic_view <yes>
atom$mat_bool_horizontal_scroll <yes>
```

# atom$mat_bool_ignore_url
## 175 ($AF)

**atom$mat_bool_ignore_url** ignores an URL; no favorite places heart appears on the form, which allows a form to appear in the history but not be a favorite place. The atom was created to handle the "Welcome" screen for Casablanca.

### Syntax

```
atom$mat_bool_ignore_url <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the URL is ignored. Values are: | |
| | Yes | The URL is ignored. |
| | No | The URL is not ignored (default). |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example ignores the URL:

```
atom$man_start_object
atom$mat_bool_ignore_url <yes>
atom$man_end_object
```

## atom$mat_bool_ignore_url_list

### 162 ($A2)

**atom$mat_bool_ignore_url_list** determines that this window object is not an item for any uniform resource locater (URL) history list or most recently used URL list. If the window is not eligible for Favorite Places (by default), its URL is not an URL list item.

### Syntax

```
atom$mat_bool_ignore_url_list <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the window object is an item for the URL list. Values are: | |
| | Yes | The window is not an URL list item. |
| | No | The window is an URL list item (default). |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example defines the window object (Shadow Game) as not an URL history list item:

```
atom$man_start_object <ind_group, "Shadow Game">
atom$mat_bool_ignore_url_list <yes>
atom$man_preset_url <”aol://1722:amer”>
```

## atom$mat_bool_importable

**38 ($26)**

**atom$mat_bool_importable** makes the data extraction feature available that lets a member paste data (using an Edit menu) into the current text view object.

### Syntax

```
atom$mat_bool_importable <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the paste feature is available for the current view. Values are: |

| | |
|---|---|
| Yes | The paste feature is available (default). |
| No | The paste feature is not available. |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example makes the paste feature available for editing in a view object:

```
atom$man_start_object <view>
atom$mat_bool_importable <yes>
```

## atom$mat_bool_inactive_for_guest
### 159 ($9F)

**atom$mat_bool_inactive_for_guest** determines whether the current object is disabled when the member is a guest.

### Syntax

```
atom$mat_bool_inactive_for_guest <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the current object is disabled when the member is a guest. Values are: |

| | |
|---|---|
| Yes | This object is disabled when the member is a guest. |
| No | This object is not disabled when the member is a guest (default). |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

None.

### Example

The following example makes the current `trigger` object (AOL icon) inactive when the member is a guest:

```
atom$mat_object_type <trigger, “”>
atom$mat_trigger_style <plain_picture>
atom$mat_relative_tag <61>
atom$mat_bool_palette_art <yes>
atom$mat_horizontal_spacing <1>
atom$mat_vertical_spacing <0>
atom$mat_bool_inactive_for_guest <yes>
atom$mat_bool_child_movable <no>
atom$mat_context_help <”Go to the AOL home page.”>
```

# atom$mat_bool_invert

**63 ($3F)**

**atom$mat_bool_invert** reverses the colors of artwork when it is displayed.

## Syntax

```
atom$mat_bool_invert <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the art colors are reversed. Values are: | |
| | Yes | The art colors are reversed. |
| | No | The art colors are not reversed (default). |

## Object(s) Handled

The objects handled by this atom are `trigger` and `ornament`.

## Return Value

None.

## Example

The following example reverses the colors of a piece of art for display:

```
atom$man_start_object <ornament>
atom$mat_art_id <1-0-1234>
atom$mat_bool_invert <yes>
.
.
.
atom$man_update_display
```

## atom$mat_bool_invisible
### 6 ($06)

**atom$mat_bool_invisible** hides an object on a window and removes the space where that object would have appeared in the window. When an object is defined invisible, other objects on the window are rearranged for proper alignment.

For example, five buttons exist on a window as part of a horizontal group and are centered. If the middle button is made invisible, the remaining four buttons are realigned as if the middle button never existed. This feature prevents a gap of blank space from appearing where the invisible button used to be.

See atom$mat_bool_hidden on page 3-64 for similar functionality.

### Syntax

```
atom$mat_bool_invisible <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether an object is invisible. Values are: | |
| | Yes | The object is invisible. |
| | No | The object is not invisible (default). |

### Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

### Return Value

None.

### Example

The following example defines object number 3 as invisible in a window:

```
atom$man_start_object <trigger>
atom$mat_relative_tag <3>
atom$mat_title <"List Responses">
atom$mat_bool_invisible <yes>
```

## atom$mat_bool_language_popup

### 117 ($75)

**atom$mat_bool_language_popup** is described in Appendix B, "Internationalization (i18n) Atoms."

## atom$mat_bool_list_allow_entry

**108 ($6C)**

**atom$mat_bool_list_allow_entry** defines whether a `dss_list` object is an editable popup list or a standard popup list. It is used in conjunction with atom$mat_height on page 3-126.

### Syntax

```
atom$mat_bool_list_allow_entry <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the `dss_list` object is an editable popup list. Values are: |
| | Yes — The object is an editable popup list. |
| | No — The object is a standard (noneditable) popup list (default). |

### Object(s) Handled

The object handled by this atom is `dss_list`.

### Return Value

None.

### Example

The following example defines the current `dss_list` object as a standard popup list:

```
atom$man_start_object <dss_list>
atom$mat_height <1>
atom$mat_bool_list_allow_entry <no>
```

### Example 2

The following example defines the current `dss_list` object as an editable popup list:

```
atom$man_start_object <dss_list>
atom$mat_height <1>
atom$mat_bool_list_allow_entry <yes>
```

# atom$mat_bool_list_icons

**26 ($1A)**

**atom$mat_bool_list_icons** leaves space for mini-icons, such as the search icon that looks like an open book, next to list box items.

This atom is used when the list box is created. Then, as each list box item is created, atom$mat_art_id on page 3-22 is used to specify the art ID for the mini-icon you select.

## Syntax

```
atom$mat_bool_list_icons <boolean>
```

`<boolean>` Specifies whether a space is provided for mini-icons in a list box. Values are:

| | |
|---|---|
| Yes | Mini-icon space is provided (default). |
| No | Mini-icon space is not provided. |

## Object(s) Handled

The objects handled by this atom are `dms_list` and `dss_list`.

## Return Value

None.

## Example

The following example puts mini-icons in the list box items:

```
atom$man_start_object <dss_list>
atom$mat_bool_list_icons <yes>
atom$man_start_object <trigger>
atom$mat_title <"About the Library">
atom$mat_art_id <0-1-1234>
atom$man_end_object
atom$man_start_object <trigger>
atom$mat_title <"Files to Download">
atom$mat_art_id <0-1-1235>
atom$man_end_object
atom$man_start_object <trigger>
atom$mat_title <"Search the Library">
atom$mat_art_id <0-1-1236>
atom$man_end_object
```

## atom$mat_bool_menu

### 42 ($2A)

**atom$mat_bool_menu** places the object being created on the main menu bar of the online service. Once an object is placed on the main menu bar, it cannot be removed.

### Syntax

```
atom$mat_bool_menu [<yes>]
```

| | |
|---|---|
| `[<yes>]` | Specifies that the object being created is added to the main menu bar. The only optional value is yes. |

### Return Value

None.

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Example

The following example places an object on the main menu bar of the online service:

```
atom$man_start_object <dss_list>
atom$mat_title <"Extras">
atom$mat_bool_menu <yes>
atom$man_start_object <trigger>
atom$mat_title <"Show Tools">
atom$man_end_object
atom$man_start_object <trigger>
atom$mat_title <"Fit Image to View">
atom$man_end_object
```

## atom$mat_bool_modal

### 5 ($05)

**atom$mat_bool_modal** forces a window to be a modal window. A modal window requires interaction from a member (for example, selecting a button) before the member can interact with any other window in the application.

### Syntax

```
atom$mat_bool_modal <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the window is modal. Values are: | |
| | Yes | The window is modal. |
| | No | The window is not modal (default). |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example defines the window as modal:

```
atom$mat_start_object <ind_group>
atom$mat_bool_modal <yes>
```

## atom$mat_bool_modified

**75 ($4B)**

**atom$mat_bool_modified** notifies the Display Manager whether the content of an `edit_view` has changed.

### Syntax

```
atom$mat_bool_modified <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the content of a view has changed. Values are: | |
| | Yes | The view content has changed. |
| | No | The view content has not changed (default). |

### Object(s) Handled

The object handled by this atom is `edit_view`.

### Return Value

None.

### Example

The following example notifies the Display Manager that the content of object 3 has not changed:

```
atom$mat_set_context_relative <3>
atom$mat_bool_modified <no>
```

# atom$mat_bool_no_border

## 74 ($4A)

**atom$mat_bool_no_border** determines whether or not to remove the border from the current object.

### Syntax

```
atom$mat_bool_no_border <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the object has a border. Values are: | |
| | Yes | Remove the border from the current object. This value is the default if the current object is `ornament`. |
| | No | Do not remove the border from the current object. This value is the default if the current object is `dms_list`, `dss_list`, `view`, or `edit_view`. |

### Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `ornament`, `view`, and `edit_view`.

### Return Value

None.

### Example

The following example removes a border from a view object:

```
atom$man_start_object <view>
atom$mat_bool_no_border <yes>
```

## atom$mat_bool_non_closeable

**36 ($24)**

**atom$mat_bool_non_closeable** prevents a window from being closed by a member. The window can be closed implicitly using **atom$man_close**.

### Syntax

```
atom$mat_bool_non_closeable <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the window can be closed by a member. Values are: | |
| | Yes | The window cannot be closed. |
| | No | The window can be closed (default). |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example defines the window as noncloseable:

```
atom$mat_start_object <ind_group>
atom$mat_title <"Welcome!">
atom$mat_bool_non_closeable <yes>
```

# atom$mat_bool_page_control

## 140 ($8C)

**atom$mat_bool_page_control** defines a `tab_group` object as a page control for a set of tabbed pages.

## Syntax

```
atom$mat_bool_page_control <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the `tab_group` object is a page control. Values are: |

| | |
|---|---|
| Yes | The `tab_group` object is a page control. |
| No | The `tab_group` object is not a page control (default). |

## Object(s) Handled

The objects handled by this atom are `tab_group` and `tab_page`.

## Return Value

None.

## Example

The following example defines the `tab_group` object as a page control of two tabbed pages:

```
atom$man_start_object <tab_group, “Tab Pages”>
atom$mat_bool_page_control> <yes>
atom$mat_bool_writeable <no>
atom$man_start_object <tab_page, “Tab 1”>
atom$mat_relative_tag <1>
atom$mat_bool_writeable <no>
atom$mat_bool_background_flood <yes>
atom$mat_color_face <234, 231, 210>
atom$mat_color_text <0, 0, 0>
atom$man_start_object <trigger, “Send”>
atom$mat_trigger_style <512>
atom$mat_art_id <1-0-4970>.
atom$man_end_object
atom$man_start_object <ornament, “”>
atom$mat_color_frame_shadow <194, 199, 24>
```

```
.
.
.
atom$man_end_object
atom$man_end_object
atom$man_end_object
atom$man_start_object <tab_page, “Tab 2”>
atom$mat_orientation <vee>
atom$mat_relative_tag <3>
.
.
.
atom$uni_end_stream
```

# atom$mat_bool_palette

## 28 ($1C)

**atom$mat_bool_palette** creates a palette window. A palette window is a floating independent window that appears on top of all other online service windows and is movable in and out of the client window region. The background defaults to a system gray color. There are no control buttons in the title bar because it is a noncloseable window with no window size control.

### Syntax

```
atom$mat_bool_palette <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the window is a palette window. Values are: |

| | |
|---|---|
| Yes | The window is a palette window. |
| No | The window is not a palette window (default). |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example defines a window as a palette window:

```
atom$man_start_object <ind_group>
atom$mat_bool_palette <yes>
```

## atom$mat_bool_palette_art

**119 ($77)**

**atom$mat_bool_palette_art** defines the current object (button) as an object of a palette window, which forces the current object background color to the system gray default.

### Syntax

```
atom$mat_bool_palette_art <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the current object is an object of a palette window. Values are: | |
| | Yes | The object is a palette window object. |
| | No | This object is not a palette window object (default). |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

None.

### Example

The following example defines the `trigger` object as a palette window object, which makes the `trigger` background gray:

```
atom$mat_object_type <trigger, “”>
atom$mat_trigger_style <plain_picture>
atom$mat_relative_tag <61>
atom$mat_bool_palette_art <yes>
atom$mat_horizontal_spacing <1>
atom$mat_vertical_spacing <0>
atom$mat_bool_active_online <yes>
atom$mat_bool_child_movable <no>
atom$mat_context_help <”Go to the AOL home page.”>
```

# atom$mat_bool_permanent

**1 ($47)**

**atom$mat_bool_permanent** defines an object as permanent so that it can never be deleted by an atom stream. This atom is used on the root object because the root of the object tree must always be present.

## Syntax

```
atom$mat_bool_permanent <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the object is permanent. Values are: | |
| | Yes | The object is permanent. |
| | No | The object is not permanent (default). |

## Object(s) Handled

All objects are handled by this atom.

## Return Value

None.

## Example

The following example defines the root object as permanent:

```
atom$man_start_object <root>
atom$mat_bool_permanent <yes>
```

## atom$mat_bool_popup_menu
### 161 ($A1)

**atom$mat_bool_popup_menu** defines a menu object as a popup menu.

### Syntax

```
atom$mat_bool_popup_menu <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the current object is a standard menu or a popup menu. Values are: |

| | |
|---|---|
| Yes | The object is a popup menu. |
| No | The object is not a popup menu (default). |

### Object(s) Handled

The object handled by this atom is `dss_list`.

### Return Value

None.

### Example

The following example defines the menu object as a popup menu:

```
atom$man_start_object <dss_list, “”>
atom$mat_relative_tag <10>
atom$mat_bool_popup_menu> <yes>
atom$man_start_object <trigger, "Menu Item 1">
.
.
.
atom$man_end_object
atom$man_start_object <trigger, "Menu Item 2">
.
.
.
atom$man_end_object
atom$man_end_context
```

# atom$mat_bool_precise

**77 ($4D)**

**atom$mat_bool_precise** positions children of an independent or organizational group with precise coordinates. Use atom$mat_precise_x on page 3-151 and atom$mat_precise_y on page 3-153 to specify the precise pixel coordinates for each child.

## Syntax

```
atom$mat_bool_precise <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether to position the current object's children with precise coordinates. Values are: | |
| | Yes | Position the children with precise coordinates. |
| | No | Do not position the children with precise coordinates (default). |

## Object(s) Handled

The objects handled by this atom are `org_group`, `ind_group`, `sms_list`, and `sss_list`.

## Return Value

None.

## Example

The following example specifies precise coordinates for positioning objects:

```
atom$man_start_object <ind_group, "Roll Over Beethoven">
atom$mat_bool_background_pic <yes>
atom$mat_art_id <1-0-2196>
atom$mat_bool_precise <yes>
atom$mat_precise_width <500>
atom$mat_precise_height <333>
atom$man_start_object <ornament, "MUSIC FORUMS">
atom$mat_orientation <hcc>
atom$mat_precise_x <9>
atom$mat_precise_y <10>
atom$mat_precise_width <204>
atom$mat_precise_height <18>
```

```
.
.
.
atom$man_end_object
.
.   (The definition of more objects goes here...)
.
atom$man_end_object
```

## atom$mat_bool_protected_input
### 34 ($22)

**atom$mat_bool_protected_input** protects and hides member input entries in a view for editable view fields.

### Syntax

```
atom$mat_bool_protected_input <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the input to the editable view field is protected. Values are: | |
| | Yes | The editable view field input is protected. |
| | No | The editable view field input is not protected (default). |

### Object(s) Handled

The object handled by this atom is `edit_view`.

### Return Value

None.

### Example

The following example protects information from being seen when it is entered in the view field by a member:

```
atom$man_start_object <edit_view>
atom$mat_bool_protected_input <yes>
```

## atom$mat_bool_repeat_animation

### 100 ($64)

**atom$mat_bool_repeat_animation** repeats the animation sequence of the current object.

### Syntax

```
atom$mat_bool_repeat_animation <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the animation sequence is repeated. Values are: | |
| | Yes | The animation sequence is repeated. |
| | No | The animation sequence is not repeated (default). |

### Object(s) Handled

The objects handled by this atom are `org_group` and `ind_group`.

### Return Value

None.

### Example

The following example repeats the animation sequence of the `trigger` object:

```
atom$man_start_object <trigger>
.
.
.
atom$mat_art_animation_rate <250>
atom$mat_art_animation_seq <1 5 0 1>
atom$mat_bool_repeat_animation <yes>
.
.
.
atom$man_end_object
```

# atom$mat_bool_resize_horizontal
**($04)**

**atom$mat_bool_resize_horizontal** determines whether an object can grow or shrink horizontally in a window when a member resizes the window.

## Syntax

```
atom$mat_bool_resize_horizontal <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether an object can grow or shrink horizontally when the window it is on is resized. Values are: | |
| | Yes | The object can grow or shrink horizontally. This value is the default if the current object is ind_group, `dms_list`, `dss_list`, `view`, or `edit_view`. |
| | No | The object cannot grow or shrink horizontally. This value is the default if the current object is `org_group`, `sms_list`, `sss_list`, `trigger`, `ornament`, `range`, or `select_range`. |

## Object(s) Handled

All objects are handled by this atom.

## Return Value

None.

## Example

The following example inhibits horizontal resizing of an object whenever the window is resized by a member:

```
atom$man_start_object <edit_view>
atom$mat_bool_resize_horizontal <no>
```

## atom$mat_bool_resize_vertical
**3 ($03)**

**atom$mat_bool_resize_vertical** determines whether an object on a window can grow or shrink vertically when a member resizes the window.

### Syntax

```
atom$mat_bool_resize_vertical <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether an object can grow or shrink vertically when the window it is on is resized. Values are: | |
| | Yes | The object can grow or shrink vertically. This value is the default if the current object is `ind_group`, `dms_list`, `dss_list`, `view`, or `edit_view`. |
| | No | The object cannot grow or shrink vertically. This value is the default if the current object is `org_group`, `sms_list`, `sss_list`, `trigger`, `ornament`, `range`, or `select_range`. |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example inhibits vertical resizing of an object when the window is resized by a member:

```
atom$man_start_object <edit_view>
atom$mat_bool_resize_vertical <no>
```

# atom$mat_bool_savabletopfc

## 176 ($B0)

**atom$mat_bool_savabletopfc** determines whether the Save To PFC menu option is enabled when the form containing this atom is the top form. It is used as a parameter of **atom$man_get_attribute**.

### Syntax

```
atom$man_get_attribute atom$mat_bool_savabletopfc
```

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

Unchanged.

### Example

The following example tells the display manager that the Save To PFC menu option is enabled when the form containing this atom is the top form:

```
atom$man_set_context_globalid <41-6139>
atom$man_get_attribute atom$mat_bool_savabletopfc
atom$if_last_return_true_then <1,2>
```

## atom$mat_bool_signoff_menu

### 173 ($AD)

**atom$mat_bool_signoff_menu** sets the boolean flag to identify the menu entry that contains the signoff menu text. Only a menu with this option flagged has its menu text changed to the string specified after the switch /O "new_text". This atom is needed to satisfy a requirement from Product Marketing, and it changes the Sign Off menu from just "Sign Off" to "Sign Off of AOL" or anything else Product Marketing wants to say.

### Syntax

```
atom$mat_bool_signoff_menu <boolean>
```

| | |
|---|---|
| `<boolean>` | Identifies the menu entry that contains the signoff menu text. Values are: |

| | |
|---|---|
| Yes | It is a signoff menu. |
| No | It is not a signoff menu (default). |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

Unchanged.

### Example

The following example changes the Sign Off menu from just "Sign Off" to Product Marketing wants it to say:

```
atom$man_start_object
atom$mat_bool_signoff_menu <yes>
atom$man_end_object
```

# atom$mat_bool_spinner

## 141 ($8D)

**atom$mat_bool_spinner** identifies the current animated art object as a spinner, for example, the spinning AOL icon in the AOL 5.0 client.

### Syntax

```
atom$mat_bool_spinner <boolean>
```

| | |
|---|---|
| `<boolean>` | Identifies the current animated art object as a spinner. Values are: |

| | |
|---|---|
| Yes | It is a spinner. |
| No | It is not a spinner (default). |

### Object(s) Handled

The object handled by this atom is `tab_group`.

### Return Value

None.

### Example

The following example identifies the animated art object as a spinner:

```
atom$man_start_object <trigger>
atom$mat_relative_tag <1>
atom$mat_art_id <1-0-2201>
atom$mat_art_animation_rate <250>
atom$mat_bool_spinner <1>
.
.
.
atom$man_end_object
```

## atom$mat_bool_static_url

### 177 ($B1)

**atom$mat_bool_static_url** lets you change the URL on hybrid forms if there is no current URL.

### Syntax

```
atom$mat_bool_static_url <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the URL remains as set. Values are: | |
| | Yes | It is a static URL. |
| | No | It is not a static URL (default). |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

Unchanged.

### Example

The following example identifies that the **http://www.aol.com** URL remains as set:

```
atom$man_start_object
atom$mat_url "http://www.aol.com"
atom$mat_bool_static_url <yes>
atom$man_end_object
```

# atom$mat_bool_tool_group

**126 ($7E)**

**atom$mat_bool_tool_group** specifies that the current organizational group (`org_group`) object appears as a toolbar.

## Syntax

```
atom$mat_bool_tool_group <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the current object appears as a toolbar. Values are: |
| | Yes — The object appears as a toolbar. |
| | No — The object does not appear as a toolbar (default). |

## Object(s) Handled

The object handled by this atom is `org_group`.

## Return Value

None.

## Example

The following example specifies that the current (`org_group`) object appears as a toolbar:

```
atom$man_start_object <org_group>
.
.
.
atom$mat_bool_tool_group <yes>
```

## atom$mat_bool_url_sink

**113 ($71)**

**atom$mat_bool_url_sink** determines whether uniform resource locators (URLs) can be dragged from a list, such as a Favorite Places list, and dropped on the current object.

### Syntax

```
atom$mat_bool_url_sink <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether URLs can be dropped on the object. Values are: |

| | |
|---|---|
| Yes | URLs can be dropped on the object. |
| No | URLs cannot be dropped on the object (default). |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example enables URLs to be dragged from a list object and dropped on the current object:

```
atom$man_start_object <ind_group>
atom$mat_bool_url_sink <yes>
atom$man_end_object
```

## atom$mat_bool_vertical_scroll
**0 ($00)**

**atom$mat_bool_vertical_scroll** selects vertical scroll bars for use with any view object displayed on a form.

### Syntax

```
atom$mat_bool_vertical_scroll <boolean>
```

| | |
|---|---|
| `<boolean>` | Specifies whether the vertical scroll bar is displayed. Values are: |

| | |
|---|---|
| Yes | The vertical scroll bar is displayed (default). |
| No | The vertical scroll bar is not displayed. |

### Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `view`, and `edit_view`.

### Return Value

None.

### Example

The following example inhibits a vertical scroll bar from appearing with an editable view:

```
atom$man_start_object <edit_view>
atom$mat_bool_vertical_scroll <no>
```

## atom$mat_bool_writeable

**39 ($27)**

**atom$mat_bool_writeable** defines the view as a main editable field in a window. This field can be saved or printed if a member selects Save or Print.

The body of the mail message on the Compose Mail form is an example of a main editable field.

### Syntax

```
atom$mat_bool_writeable <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether an object is the main editable field on the window. Values are: | |
| | Yes | The object is the main editable field. |
| | No | The object is not the main editable field (default). |

### Object(s) Handled

The object handled by this atom is `edit_view`.

### Return Value

None.

### Example

The following example defines the view as a main editable field:

```
atom$man_start_object <ind_group>
atom$mat_title <"Compose Mail">
atom$man_start_object <edit_view>
atom$mat_bool_writeable <yes>
```

The result of this atom stream is that when a member selects Print from the main menu bar of the online service application, the content of the Compose Mail form prints.

# atom$mat_bottom_spacing

## 189 ($BD)

**atom$mat_bottom_spacing** sets the bottom spacing in the Tokyo (Windows) client.

### Syntax

```
atom$mat_bottom_spacing <x>
```

|  |  |
|---|---|
| `<x>` | Specifies the bottom spacing in pixels. |

### Return Value

None.

### Object(s) Handled

None.

### Example

The following example creates a button with 2 pixels of space (padding) on the bottom:

```
atom$man_start_object <trigger, "Hello">
atom$mat_left_spacing <4>
atom$mat_top_spacing <2>
atom$mat_right_spacing <4>
atom$mat_bottom_spacing <2>
atom$man_end_object
```

## atom$mat_capacity

### 9 ($09)

**atom$mat_capacity** specifies a limit on the number of characters an editable view field can contain.

### Syntax

```
atom$mat_capacity <x>
```

| | |
|---|---|
| `<x>` | Specifies how many characters are allowed for this field. Values are **1** or greater. |

### Object(s) Handled

The object handled by this atom is `edit_view`.

### Return Value

None.

### Example

The following example specifies a character limit of 32000 on an editable view:

```
atom$man_start_object <edit_view>
atom$mat_capacity <32000>
```

# atom$mat_color_bottom_edge

**85 ($55)**

**atom$mat_color_bottom_edge** defines the color of the bottom and right edges of the current button object.

## Syntax

```
atom$mat_color_bottom_edge <r g b>
```

| | | |
|---|---|---|
| `<r g b>` | Specifies a color using a 3-byte RGB value as follows: | |
| | `r` | First byte defines the red value. |
| | `g` | Second byte defines the green value. |
| | `b` | Third byte defines the blue value. |

## Object(s) Handled

The objects handled by this atom are `ind_group`, `dms_list`, `sms_list`, `dss_list`, `sss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

## Return Value

None.

## Example

The following example defines the button's bottom and right edges as green:

```
atom$man_start_object <trigger>
atom$mat_art_id <1-0-2199>
atom$mat_color_bottom_edge <0 255 0>
```

## atom$mat_color_face

**82 ($52)**

**atom$mat_color_face** defines the face color of the current object. The face of a button or group is the main area, minus the edges. The face of a window is the background of the window.

### Syntax

```
atom$mat_color_face <r g b>
```

| | |
|---|---|
| `<r g b>` | Specifies a color using a 3-byte RGB value as follows: |

| | |
|---|---|
| `r` | First byte defines the red value. |
| `g` | Second byte defines the green value. |
| `b` | Third byte defines the blue value. |

### Object(s) Handled

The objects handled by this atom are `org_group`, `ind_group`, `dms_list`, `dss_list`, and `trigger`.

### Return Value

None.

### Example

The following example defines the face color of a trigger as green:

```
atom$man_start_object <trigger, "Continue">
atom$mat_color_face <0 80 0>
```

# atom$mat_color_frame_hilight

## 95 ($5F)

**atom$mat_color_frame_hilight** defines the highlighted color of the frame for the current object. Frame styles can have a highlighted edge and/or a shadowed edge. For more information about frame styles, see atom$mat_frame_style on page 3-124.

### Syntax

```
atom$mat_color_frame_hilight <r g b>
```

| | |
|---|---|
| `<r g b>` | Specifies a color using a 3-byte RGB value as follows: |

| | |
|---|---|
| `r` | First byte defines the red value. |
| `g` | Second byte defines the green value. |
| `b` | Third byte defines the blue value. |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example defines the color of a frame’s highlighted edge:

```
atom$man_start_object <ornament, "MUSIC FORUMS">
atom$mat_orientation <hcc>
atom$mat_precise_x <9>
atom$mat_precise_y <10>
atom$mat_precise_width <204>
atom$mat_precise_height <18>
atom$mat_font_sis <0 10 1>
atom$mat_color_text <248 120 120>
atom$mat_color_face <1 1 1>
atom$mat_color_frame_hilight <124 126 0>
atom$mat_color_frame_shadow <255 255 255>
atom$mat_frame_style <7>
```

## atom$mat_color_frame_shadow

### 96 ($60)

**atom$mat_color_frame_shadow** defines the shadowed color of the frame for the current object. Frame styles can have a highlighted edge and/or a shadowed edge. For more information about frame styles, see atom$mat_frame_style on page 3-124.

### Syntax

```
atom$mat_color_frame_shadow <r g b>
```

| | | |
|---|---|---|
| `<r g b>` | Specifies a color using a 3-byte RGB value as follows: | |
| | `r` | First byte defines the red value. |
| | `g` | Second byte defines the green value. |
| | `b` | Third byte defines the blue value. |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example defines the color of a frame’s shadowed edge:

```
atom$man_start_object <ornament, “MUSIC FORUMS”>
atom$mat_orientation <hcc>
atom$mat_precise_x <9>
atom$mat_precise_y <10>
atom$mat_precise_width <204>
atom$mat_precise_height <18>
atom$mat_font_sis <0 10 1>
atom$mat_color_text <248 120 120>
atom$mat_color_face <1 1 1>
atom$mat_color_frame_hilight <124 126 0>
atom$mat_color_frame_shadow <255 255 255>
atom$mat_frame_style <7>
```

## atom$mat_color_selected

### 89 ($59)

**atom$mat_color_selected** defines the color of the current button object that appears when the button is selected by a member.

### Syntax

```
atom$mat_color_selected <r g b>
```

| | |
|---|---|
| `<r g b>` | Specifies a color using a 3-byte RGB value as follows: |

| | |
|---|---|
| `r` | First byte defines the red value. |
| `g` | Second byte defines the green value. |
| `b` | Third byte defines the blue value. |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example defines a button color (yellow/black) that appears only when it is selected:

```
atom$man_start_object <trigger>
atom$mat_color_selected <124 126 0>
```

## atom$mat_color_text
### 83 ($53)

**atom$mat_color_text** defines the text color for the current object.

### Syntax

```
atom$mat_color_text <r g b>
```

| | | |
|---|---|---|
| `<r g b>` | Specifies a color using a 3-byte RGB value as follows: | |
| | `r` | First byte defines the red value. |
| | `g` | Second byte defines the green value. |
| | `b` | Third byte defines the blue value. |

### Object(s) Handled

The objects handled by this atom are `dms_list`, `sms_list`, `dss_list`, `sss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

### Return Value

None.

### Example

The following example defines the text to appear black:

```
atom$man_start_object <dss_list>
atom$mat_color_face <248 232 208>
atom$mat_color_text <64 64 64>
atom$man_start_object <trigger, "About This Area">
```

## atom$mat_color_text_shadow
### 90 ($5A)

**atom$mat_color_text_shadow** applies a colorized shadow to the text of the current object. By default, text is not shadowed.

### Syntax

```
atom$mat_color_text_shadow <r g b>
```

| | |
|---|---|
| `<r g b>` | Specifies a shadow color using a 3-byte RGB value as follows: |

| | |
|---|---|
| `r` | First byte defines the red value. |
| `g` | Second byte defines the green value. |
| `b` | Third byte defines the blue value. |

### Object(s) Handled

The object handled by this atom is `ornament`.

### Return Value

None.

### Example

The following example specifies the text to be shadowed in a particular color:

```
atom$man_start_object <ornament, "MUSIC FORUMS">
atom$mat_orientation <hcc>
atom$mat_precise_x <9>
atom$mat_precise_y <10>
atom$mat_precise_width <204>
atom$mat_precise_height <18>
atom$mat_font_sis <0 10 1>
atom$mat_color_text <248 120 120>
atom$mat_color_text_shadow <192 192 192>
atom$mat_color_face <1 1 1>
atom$mat_color_frame_hilight <124 126 0>
atom$mat_color_frame_shadow <255 255 255>
atom$mat_frame_style <7>
```

## atom$mat_color_top_edge

**84 ($54)**

**atom$mat_color_top_edge** defines the color for the top and left edges of the current button object.

### Syntax

```
atom$mat_color_top_edge <r g b>
```

| | | |
|---|---|---|
| `<r g b>` | Specifies a color using a 3-byte RGB value as follows: | |
| | `r` | First byte defines the red value. |
| | `g` | Second byte defines the green value. |
| | `b` | Third byte defines the blue value. |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

None.

### Example

The following example defines the color of a button's top and left edges:

```
atom$man_start_object <trigger>
atom$mat_art_id <1-0-2199>
atom$mat_color_top_edge <124 126 0>
```

## atom$mat_command_key

### 53 ($35)

**atom$mat_command_key** assigns a key or key combination to activate a particular object. This feature lets you set up shortcut keys for members to use.

Command keys are typically defined for menu bar items. When a command key is pressed, the Display Manager first looks for the object on the top window, and if the object is not found, it looks on the menu bar. If an object on the menu bar has an assigned command key, the command key is automatically displayed on the drop-down menu to the right of the object's title.

### Syntax

```
atom$mat_command_key <first_key[, modifier_key]>
```

| | |
|---|---|
| `<first_key>` | Specifies an ASCII key that the member must press to activate an object. |
| `[<modifier_key>]` | Specifies if a second key must be used in conjunction with the first key and, if so, designates which key. Values are: |

| | |
|---|---|
| `0` | No second key is used. The member types only the ASCII key specified in the first argument. |
| `1` | The second key is the CTRL key and must be used in conjunction with the ASCII key identified in the first argument (default). |
| `2` | The second key is the SHIFT key and must be used in conjunction with the ASCII key identified in the first argument. |

### Object(s) Handled

The object handled by this atom is `trigger`.

### Return Value

None.

## Example

The following example assigns command keys for an object on a window:

```
atom$man_start_object <trigger>
atom$mat_title <"Keyword">
atom$mat_command_key <k, 2>
```

The result of this example is that the action associated with the object titled "Keyword" is initiated when a member simultaneously presses the SHIFT and K keys.

# atom$mat_context_help

## 102 ($66)

**atom$mat_context_help** invokes a help bubble (or balloon help) when a member positions the cursor over a specific object or location on a form. The argument to this atom is the help text that appears in the help bubble.

### Syntax

```
atom$mat_context_help <help_text>
```

| | |
|---|---|
| `<help_text>` | The text string that appears in the help bubble. |

### Object(s) Handled

The objects handled by this atom are `dms_list`, `sms_list`, `dss_list`, `sss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

### Return Value

None.

### Example

The following example associates a help bubble with the mail write button on the toolbar of the online service:

```
atom$man_start_object <trigger>
atom$mat_context_help <"Write mail and send files.">
.
.
.
atom$man_end_object
```

## atom$mat_dirty_query

**54 ($36)**

**atom$mat_dirty_query** determines whether a member is prompted when attempting to close a window when any editable field on the window has been changed. This allows the member to save information in the window before it is closed.

### Syntax

```
atom$mat_dirty_query <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the member is prompted before the window is closed. Values are: | |
| | Yes | The member is prompted. |
| | No | The member is not prompted (default). |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example specifies that a prompt to a member is required whenever the member attempts to close the window while the editable fields on the window have changes:

```
atom$man_start_object <ind_group>
atom$mat_title <"NEW.DOC">
atom$mat_dirty_query <yes>
atom$man_start_object <edit_view>
atom$mat_bool_writeable <yes>
atom$man_end_object
atom$man_start_object <trigger>
```

# atom$mat_factory_id
## 191 ($BF)

**atom$mat_factory_id** notifies the Display Manager that a particular object is managed by another GUID. The factory ID is a 16 byte GUID (as created by Microsoft's UUIDGEN tool, for example). This ID is used to look up the tool or object that used to create the window (independent group in this example) when the **man_update_display** is processed. A GUID is used because it can be uniquely generated by anyone and a global list does not have to be maintained by AOL. Unlike **mat_tool_id**, which is generally applied to "views", **mat_factory_id** can be used for any type of FDO object.

### Syntax

```
atom$mat_factory_id <x>
```

| | |
|---|---|
| `<x>` | Specifies the GUID that manages the object. |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example notifies the Display Manager that a particular object is managed by another factory ID:

```
man_start_object <ind_group, "Hello">
mat_factory_id <23x, EAx, 9Bx, 04x, 67x, CCx, 4Ax, 50x, F7x,
A8x, 97x, 00x, 87x, 35x, 11x, D1x>
...
...
...
man_update_display
man_end_object
```

## atom$mat_field_script

### 104 ($68)

**atom$mat_field_script** is described in Appendix B, "Internationalization (i18n) Atoms."

# atom$mat_font_id

## 48 ($30)

**atom$mat_font_id** defines the font ID for the current object. This atom has been superseded by atom$mat_font_sis on page 3-119, which lets you set the font ID, the font size, and font style.

## Syntax

```
atom$mat_font_id <font_id>
```

`<font_id>` Specifies the font ID in which to display the text. Values are:

| | |
|---|---|
| `0` | arial (If not found, ms_sans_serif) |
| `1` | courier |
| `2` | times_roman (If not found, ms_serif) |
| `3` | system |
| `4` | fixed_system |
| `5` | ms_serif |
| `6` | ms_sans_serif |
| `7` | small_fonts |
| `8` | courier_new |
| `9` | script |
| `10` | ms_mincho |
| `11` | ms_gothic |

## Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

## Return Value

None.

## Example

The following example defines an MS serif font for the object title News:

```
atom$man_start_object <trigger>
atom$mat_title <"News">
atom$mat_font_id <5>
```

# atom$mat_font_sis

## 10 ($0A)

**atom$mat_font_sis** defines the font ID, size, and style for the current object. This atom can be used with any field that displays text. This atom supersedes **atom$mat_font_id**, **atom$mat_font_size**, and **atom$mat_font_style**, and should be used to set the font ID, font size, and font style.

### Syntax

```
atom$mat_font_sis <font_id[, size][, style]>
```

| | |
|---|---|
| `<font_id>` | Specifies the font ID in which to display the text. Values are: |

| | |
|---|---|
| `0` | arial (If not found, ms_sans_serif) |
| `1` | courier |
| `2` | times_roman (If not found, ms_serif) |
| `3` | system |
| `4` | fixed_system |
| `5` | ms_serif |
| `6` | ms_sans_serif |
| `7` | small_fonts |
| `8` | courier_new |
| `9` | script |
| `10` | ms_mincho |
| `11` | ms_gothic |

| | |
|---|---|
| `[<size>]` | Specifies the point size in which to display the text. |
| `[<style>]` | Specifies the style in which to display the text. Values are: |

| | |
|---|---|
| `0` | normal |

| | |
|---|---|
| `1` | bold |
| `2` | italic |
| `4` | underline |

If the style values are added together, the text displays a combination of the styles. For example, a value of **3** displays text in bold and italic.

## Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

## Return Value

None.

## Example

The following example defines an Arial font, 8 point font size, and bold font style to an object's title:

```
atom$man_start_object <trigger>
atom$mat_title <"What's New">
atom$mat_font_sis <0, 8, 1>
```

# atom$mat_font_size

**49 ($31)**

**atom$mat_font_size** defines the font size for the current object. This atom has been superseded by atom$mat_font_sis on page 3-119, which lets you set the font size, the font ID, and font style.

## Syntax

```
atom$mat_font_size <size>
```

| | |
|---|---|
| `<size>` | Specifies the point size in which to display the text. |

## Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

## Return Value

None.

## Example

The following example defines a font size 8 for an object's title:

```
atom$man_start_object <trigger>
atom$mat_title <"News">
atom$mat_font_size <8>
```

## atom$mat_font_style

### 50 ($32)

**atom$mat_font_style** defines the font style for the current object. This atom has been superseded by atom$mat_font_sis on page 3-119, which lets you set the font style, the font ID, and font size.

### Syntax

```
atom$mat_font_style <style>
```

`<style>` Specifies the style in which to display the text. This is an optional argument. Values are:

| | |
|---|---|
| `0` | normal |
| `1` | bold |
| `2` | italic |
| `4` | underline |

If the style values are added together, the text displays a combination of the styles. For example, a value of **3** displays text in bold and italic style.

### Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

### Return Value

None.

### Example

The following example sets the font's style to bold:

```
atom$man_start_object <trigger>
atom$mat_title <"News">
atom$mat_font_style <4>
```

## atom$mat_form_icon

### 115 ($73)

**atom$mat_form_icon** specifies the art ID for the mini-icon located in the upper-left corner of a window common to all Windows 95 applications. This atom is intended for use with the 32-bit (Windows 95) client version of the service.

### Syntax

```
atom$mat_form_icon <art_id>
```

| | |
|---|---|
| `<art_id>` | Specifies the art ID for the icon. |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example assigns the art ID for the icon for the `trigger` object:

```
atom$man_start_object <trigger>
atom$mat_relative_tag <1>
atom$mat_trigger_style <2>
atom$mat_art_id <1-2-3456>
atom$mat_form_icon <1-0-2199>
atom$mat_precise_x <370>
atom$mat_precise_y <10>
.
.
.
atom$man_end_object
```

## atom$mat_frame_style

### 87 ($57)

**atom$mat_frame_style** determines the type of frame style for the current object.

### Syntax

```
atom$mat_frame_style <type>
```

| | |
|---|---|
| `<type>` | Specifies the type of frame style to use. Values are: |

| | |
|---|---|
| `0` | `none`—Single line border (Windows default) |
| `1` | `single_line_pop_out`—Single line border with pop-out shadowing |
| `2` | `single_line_pop_in`—Single line border with pop-in shadowing |
| `3` | `pop_in`—No border with pop-in shadowing |
| `4` | `pop_out`—No border with pop-out shadowing |
| `5` | `double_line`—Double-line border |
| `6` | `shadow`—Standard shadowing |
| `7` | `highlight`—Highlighting for static text |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example defines the frame style (no border with pop-in shadowing) for a dynamic, single-select list:

```
atom$man_start_object <dss_list>
atom$mat_frame_style <3>
```

## atom$mat_height
### 16 ($10)

**atom$mat_height** specifies the height of the current object. You can use this atom to establish how many text lines you want to appear in a view field. **atom$mat_height** and **atom$mat_width** work independently on a window.

### Syntax

```
atom$mat_height <x>
```

| | |
|---|---|
| `<x>` | Specifies the height for the current object. Values are **1** or greater. A value of **1** is equal to the height of one line of text in the standard system font. |

### Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

### Return Value

None.

### Example

The following example specifies the height of a view object:

```
atom$man_start_object <view>
atom$mat_height <10>
atom$mat_width <60>
```

This example sets the height of the view to 10 characters. The height is based on the size of the standard system font.

## atom$mat_horizontal_spacing

**57 ($39)**

**atom$mat_horizontal_spacing** changes the horizontal spacing between the current object and any object to its right. You can use this atom to change the spacing between objects on a window.

### Syntax

```
atom$mat_horizontal_spacing <x>
```

| | |
|---|---|
| `<x>` | Specifies the amount of horizontal spacing in multiples of normal spacing. Values are **1** or greater. For example, a value of **1** is equal to the normal (default) space between two objects, **2** is equal to twice the normal space, and so on. A value of **0** equals no spacing, which is used for objects on the toolbar. |

### Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

### Return Value

None.

### Example

The following example changes the horizontal spacing of two buttons:

```
atom$man_start_object <ind_group>
atom$mat_orientation <hff>
atom$mat_position <hfc>
atom$man_start_object <trigger>
atom$mat_title <"Button 1">
atom$mat_horizontal_spacing <3>
atom$man_end_object
atom$man_start_object <trigger>
atom$mat_title <"Button 2">
```

## atom$mat_increment

### 73 ($49)

**atom$mat_increment** defines the incremental value for editable range fields (spin gadgets). atom$mat_minimum on page 3-137 and atom$mat_maximum on page 3-136 are used in conjunction with this atom.

The application of this atom can be seen on the Download Preferences window of the Windows client of the online service, where a member can set the number of downloads on which to retain information.

### Syntax

```
atom$mat_increment <x>
```

| | |
|---|---|
| `<x>` | Specifies the incremental numeric value for the editable range field. Values are **0** or greater with a default value of **10**. |

### Object(s) Handled

The object handled by this atom is `select_range`.

### Return Value

None.

### Example

The following example defines an incremental value of **10** for an editable range:

```
atom$man_start_object <select_range>
atom$mat_minimum <0>
atom$mat_maximum <100>
atom$mat_increment <10>
```

# atom$mat_intl_font_sis

## 178 ($B2)

**atom$mat_intl_font_sis** defines the secondary font ID, size, and style for the current object. By default, the current object should use the font defined by **atom$mat_font_sis** (or **atom$mat_font_id**, **atom$mat_font_size**, **atom$mat_font_style**). When international text is sent to the object, based on the text type, the secondary font may be used to display the international text.

### Syntax

```
atom$mat_intl_font_sis <font_id[, size][, style]>
```

`<font_id>` Specifies the font ID in which to display the text. Values are:

| | |
|---|---|
| `0` | `arial` (If not found, `ms_sans_serif`) |
| `1` | courier |
| `2` | `times_roman` (If not found, `ms_serif`) |
| `3` | system |
| `4` | fixed_system |
| `5` | ms_serif |
| `6` | ms_sans_serif |
| `7` | small_fonts |
| `8` | courier_new |
| `9` | script |
| `10` | ms_mincho |
| `11` | ms_gothic |

`[<size>]` Specifies the point size in which to display the text.

`[<style>]` Specifies the style in which to display the text. Values are as follows:

| | |
|---|---|
| `0` | normal |

| | |
|---|---|
| `1` | bold |
| `2` | italic |
| `4` | underline |

## Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

## Return Value

None.

## Example

The following example defines an Arial font, 8 point font size, and bold font style to an object's title:

```
atom$man_start_object <trigger>
atom$mat_title <"What's New">
atom$mat_intl_font_sis <0, 8, 1>
```

## atom$mat_left_spacing
### 186 ($BA)

**atom$mat_left_spacing** sets the left spacing in the Tokyo (Windows) client.

### Syntax

```
atom$mat_left_spacing <x>
```

| | |
|---|---|
| `<x>` | Specifies the left spacing in pixels. |

### Return Value

None.

### Object(s) Handled

### Example

The following example creates a button with 4 pixels of space (padding) on the left:

```
atom$man_start_object <trigger, "Hello">
atom$mat_left_spacing <4>
atom$mat_top_spacing <2>
atom$mat_right_spacing <4>
atom$mat_bottom_spacing <2>
atom$man_end_object
```

## atom$mat_link_content_to_rid

### 76 ($4C)

**atom$mat_link_content_to_rid** links multiple text fields and saves them as one piece of text. When you link text fields, the first text field must always be defined as the main editable view, using **atom$mat_bool_writeable**.

This atom is currently used in many Rainman forms. For more information about Rainman, see the America Online manual, *Bringing Information Online Using Rainman*.

### Syntax

```
atom$mat_link_content_to_rid <x>
```

| | |
|---|---|
| `<x>` | Specifies the relative ID of the next text field to link the current text field to. Values are **1** or greater. |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example links multiple text fields together:

```
atom$uni_start_stream
.
.
.
atom$man_start_object <edit_view>
atom$mat_relative_tag <1>
atom$mat_bool_writeable <yes>
atom$mat_link_content_to_rid <2>
atom$mat_end_object
atom$man_start_object <edit_view>
atom$mat_relative_tag <2>
atom$mat_link_content_to_rid <3>
atom$mat_end_object
.
.  (Definition of object with relative ID 3 goes here...)
```

```
.
atom$uni_end_stream
```

## atom$mat_log_object

### 65 ($41)

**atom$mat_log_object** specifies the logging for an object. The argument defines which log holds the object data, or the argument can turn logging off. The object must be the main view on the window.

### Syntax

```
atom$mat_log_object [<log_flag>]
```

`[<log_flag>]` Specifies which log to use or turns off object logging. Values are:

| | |
|---|---|
| `0` | session_log—Session log (Default) |
| `1` | chat_log—Chat log |
| `2` | im_log—Instant Message™ log |
| `3` | no_log—Do not log |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example specifies that data be logged to the session log for a view object:

```
atom$man_start_object <view>
atom$mat_bool_writeable <yes>
atom$mat_log_object <0>
```

## atom$mat_managed_by

### 55 ($37)

**atom$mat_managed_by** notifies the Display Manager that a particular object is managed by another protocol (tool).

### Syntax

```
atom$mat_managed_by <x>
```

| | |
|---|---|
| `<x>` | Specifies the protocol ID that manages the object. Protocol IDs are maintained in the **atoms.h** file and are listed in Table 3-1 on page 3-1 of this manual. |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example notifies the Display Manager that an object is managed by another protocol with an ID of 20 (Multimedia Interface):

```
atom$man_start_object <view>
atom$mat_managed_by <20>
```

## atom$mat_maximum

### 60 ($3C)

**atom$mat_maximum** specifies the maximum value of a range field. This atom is used in conjunction with atom$mat_minimum on page 3-137.

The application of this atom can be seen on the Download Preferences window of the Windows client of the online service, where a member can set the number of downloads on which to retain information. This atom can also be used with noneditable ranges (for example, gauges, or meters).

### Syntax

```
atom$mat_maximum <x>
```

| | |
|---|---|
| `<x>` | Specifies the highest allowable number for the range field. Values are **0** or greater. The default value is **100** except when the current object type is `range` or `select_range`. |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example specifies a minimum and maximum value for a range:

```
atom$man_start_object <select_range>
atom$mat_minimum <0>
atom$mat_maximum <100>
```

# atom$mat_minimum

## 59 ($3B)

**atom$mat_minimum** specifies the minimum value of a range field. This atom is used in conjunction with atom$mat_maximum on page 3-136.

The application of this atom can be seen on the Download Preferences window of the Windows client of the online service, where a member can set the number of downloads on which to retain information. This atom can also be used with noneditable ranges (for example, gauges or meters).

## Syntax

```
atom$mat_minimum <dword>
```

| | |
|---|---|
| `<dword>` | Specifies the lowest allowable value for the range field. Values are **0** or greater with a default value of **0**. |

## Object(s) Handled

The objects handled by this atom are `range` and `select_range`.

## Return Value

None.

## Example

The following example specifies a minimum and maximum value for a range:

```
atom$man_start_object <select_range>
atom$mat_minimum <0>
atom$mat_maximum <100>
```

## atom$mat_object_id
### 12 ($0C)

**atom$mat_object_id** defines the global ID for the current object.

### Syntax

```
atom$mat_object_id <global_ID>
```

| | |
|---|---|
| `<global_ID>` | Specifies the global ID for the current object. The default value is **0**. |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example assigns a global ID of 32-137 to the window:

```
atom$man_start_object <ind_group>
atom$mat_object_id <32-137>
```

# atom$mat_object_type

## 45 ($2D)

**atom$mat_object_type** defines the object type of the current object.

### Syntax

```
atom$mat_object_type <object_type>
```

`<object_type>` Specifies the object type for the current object. The object type can be specified using the object ID or object name, as follows:

| | |
|---|---|
| `0` | `org_group`—An organizational group. |
| `1` | `ind_group`—An independent group (for example, a window). |
| `2` | `dms_list`—A dynamic, multiselect list (for example, a list box where more than one item is selectable). |
| `3` | `sms_list`—A static, multiselect list (for example, a check box). |
| `4` | `dss_list`—A dynamic, single-select list (for example, a list box where only one item is selectable). |
| `5` | `sss_list`—A static, single-select list (for example, radio buttons). |
| `6` | `trigger`—An actionable object (for example, a button, icon, menu item). |
| `7` | `ornament`—A static or decorative object (for example, a picture, static text). |
| `8` | `view`—A view field (used for display of larger amounts of text or a graphic where, if necessary, scroll bars are available). |
| `9` | `edit_view`—An editable view field (used for input text fields where members can enter text). |

|  |  |
|---|---|
| 12 | `range` — A range (used for display of graphic representation of a value, such as a bar gauge). |
| 13 | `select_range` — A selectable range (used for an easy user interface for entering numeric data, such as a spin gadget). |
| 17 | `tool_group` — A toolbar (used on any form or frame window including tab controls). |
| 18 | `tab_group` — A tab control (defines an object group of tab pages, frames or forms). |
| 19 | `tab_page` — A tabbed page, frame or form (used on a list box with a labeled tab). |

## Object(s) Handled

All objects are handled by this atom.

## Return Value

None.

## Example

The following example assigns view as an object type to an object:

```
atom$man_insert_object_before <32-123>
atom$mat_object_type <view>
```

# atom$mat_orientation

## 8 ($08)

**atom$mat_orientation** defines justification and orientation for groups, static lists, and `ornament` text fields.

## Syntax

```
atom$mat_orientation <o_hj_vj>
```

| | |
|---|---|
| `<o_hj_vj>` | Specifies the group's orientation and justification. The argument requires three characters. The first character defines the orientation axis with the following values: |

| | |
|---|---|
| `h` | Horizontal |
| `v` | Vertical |

The second character defines the type of justification on the horizontal axis with the following values:

| | |
|---|---|
| `c` | Center |
| `l` | Left |
| `r` | Right |
| `f` | Full (similar to even justification but no space is left between outside objects and the edge of the screen) |
| `e` | Even (evenly spaced) |
| `n` | None |

The third character defines the type of justification on the vertical axis with the following values:

| | |
|---|---|
| `c` | Center |
| `t` | Top |
| `b` | Bottom |
| `f` | Full (similar to even justification but no space is left between outside objects and the edge of the screen) |

| | |
|---|---|
| `e` | Even (evenly spaced) |
| `n` | None |

This 8-bit argument can also have binary values assigned as follows:

Bits 7 and 6 Define the orientation axis with values as follows:

| | |
|---|---|
| `0` | Horizontal |
| `1` | Vertical |

Bits 5 - 0 Define the type of justification. Bits 5, 4, and 3 define the horizontal and bits 2, 1, and 0 define the vertical justification type, with 3-bit values for each set as follows:

| | |
|---|---|
| `2` | Center |
| `3` | Left |
| `4` | Right |
| `5` | Full |
| `6` | Even |

## Object(s) Handled

The objects handled by this atom are `org_group`, `ind_group`, `ornament`, `sms_list`, `sss_list`, `view`, and `edit_view`.

## Return Value

None.

## Example

The following example defines the orientation and justification for objects on a form:

```
atom$man_start_object <ind_group>
```

```
atom$man_start_object <org_group>
atom$mat_orientation <hee>
atom$man_start_object <trigger>
atom$mat_title <"Button 1">
atom$man_end_object
atom$man_start_object <trigger>
atom$mat_title <"Button 2">
atom$man_end_object
atom$man_start_object <trigger>
atom$mat_title <"Button 3">
atom$man_end_object
```

The result of this example is shown in Figure 3-1. It is a form with an organizational group made up of three buttons. The buttons are horizontally oriented and have even justification horizontally and vertically, as shown in Figure 3-1:

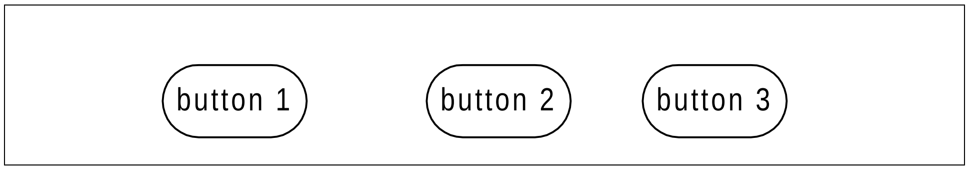


*Figure 3-1: Horizontal Group Orientation*

## atom$mat_paragraph

**66 ($42)**

**atom$mat_paragraph** determines whether text in a view field wraps to an indent after the first line of text.

Note that this atom is used on chat views to allow room for the screen name that appears to the left of a member's comments.

### Syntax

```
atom$mat_paragraph <boolean>
```

| | | |
|---|---|---|
| `<boolean>` | Specifies whether the text wraps to an indent. Values are: | |
| | Yes | Specifies that text indent is on. |
| | No | Specifies that text indent is off (default). |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example wraps the text to an indent in a view field for chat:

```
atom$man_start_object <view, “”>
atom$mat_size <50, 17, 8192>
atom$mat_relative_tag <256>
atom$mat_scroll_threshold <4096>
atom$mat_bool_force_scroll <yes>
atom$mat_paragraph <yes>
atom$mat_log_object <chat_log>
atom$mat_ruler <15, 45>
atom$mat_bool_force_scroll <yes>
atom$mat_bool_writeable <yes>
atom$man_end_object
```

# atom$mat_popup_relative_id

**163 ($A3)**

**atom$mat_popup_relative_id** assigns a relative ID to a popup menu object.

## Syntax

```
atom$mat_popup_relative_id <x>
```

| | |
|---|---|
| `<x>` | Specifies the relative ID of the popup menu. Values are **1** or greater. |

## Object(s) Handled

The objects handled by this atom are `dss_list` and `trigger`.

## Return Value

None.

## Example

The following example defines the relative ID of the popup menu:

```
atom$uni_start_stream
.
.
.
atom$man_start_object <trigger, “”>
atom$mat_popup_relative_id <2>
atom$mat_popup_pfc_path <”C:\AOL40\organize\”>
atom$man_end_context
.
.
.
atom$uni_end_stream
```

## atom$mat_popup_pfc_path

### 164 ($A4)

**atom$mat_popup_pfc_path** defines the path to the top-level object to build a popup menu.

### Syntax

```
atom$mat_popup_pfc_path <string>
```

| | |
|---|---|
| `<string>` | Specifies the path to the top-level object to build a popup menu. |

### Object(s) Handled

The object handled by this atom is `dss_list` and `trigger`.

### Return Value

None.

### Example

The following example defines the path to build a popup menu:

```
atom$uni_start_stream
.
.
.
atom$man_start_object <trigger, “”>
atom$mat_popup_relative_id <2>
atom$mat_popup_pfc_path <”C:\AOL40\organize”>
atom$man_end_context
.
.
.
atom$uni_end_stream
```

# atom$mat_position

**64 ($40)**

**atom$mat_position** determines the window's position on the screen. This atom is only used for independent group (`ind_group`) objects.

## Syntax

```
atom$mat_position <position>
```

| | | |
|---|---|---|
| `<position>` | Specifies the window's position on the screen. Values are as follows: | |
| | `0` | Defines a cascaded display that arranges active windows in a cascading fashion with the current window on top. |
| | `1-9` | Figure 3-2 shows a screen divided into 9 areas that relate to defined position values **1** through **9**. |

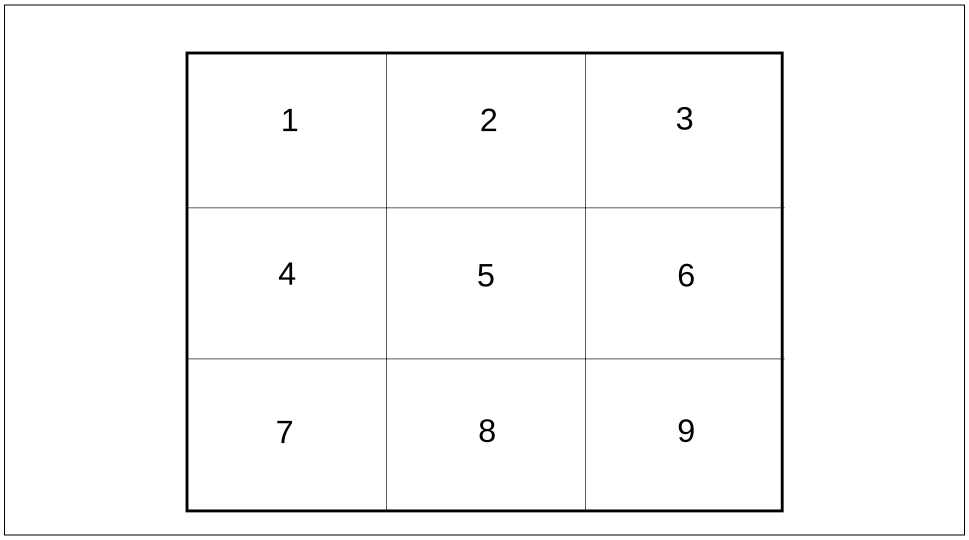


*Figure 3-2: Window Positioning*

## Object(s) handled

The object handled by this atom is `ind_group.`

## Return Value

None.

## Example

The following example specifies the object’s position as in the top right window of the screen:

```
atom$man_start_object <ind_group> ""
atom$mat_orientation <vff>
atom$mat_position <3>
...
atom$man_end_object
```

# atom$mat_precise_height

## 79 ($4F)

**atom$mat_precise_height** defines the height of the current object in pixels. To set both height and width, use atom$mat_size on page 3-165.

### Syntax

```
atom$mat_precise_height <x>
```

| | |
|---|---|
| `<x>` | Specifies the height in pixels. Values are **1** or greater. |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example defines a window to have a precise height of 333 pixels:

```
atom$man_start_object <ind_group>
atom$mat_bool_precise <yes>
atom$mat_precise_height <333>
```

## atom$mat_precise_width

### 78 ($4E)

**atom$mat_precise_width** defines the width of the current object in pixels. To set both height and width, use atom$mat_size on page 3-165.

### Syntax

```
atom$mat_precise_width <x>
```

| | |
|---|---|
| `<x>` | Specifies the width in pixels. Values are **1** or greater. |

### Object(s) Handled

All objects are handled by this atom.

### Return Value

None.

### Example

The following example defines a window to have a precise width of 500 pixels:

```
atom$man_start_object <ind_group>
atom$mat_bool_precise <yes>
atom$mat_precise_width <500>
```

# atom$mat_precise_x

## 80 ($50)

**atom$mat_precise_x** determines the horizontal (x-axis) coordinate in pixels that is used to position children of a precise group. This atom is used in conjunction with atom$mat_precise_y on page 3-153.

## Syntax

```
atom$mat_precise_x <x>
```

| | |
|---|---|
| `<x>` | Specifies the horizontal pixel coordinate used to position children of a precise group. Values are **0** or greater. |

## Object(s) Handled

The objects handled by this atom are `org_group`, `dms_list`, `sms_list`, `dss_list`, `sss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

## Return Value

None.

## Example

The following example defines the placement of the text `ornament` (Music Forums) to have a precise horizontal coordinate of 9 pixels in the Roll Over Beethoven window:

```
atom$man_start_object <ind_group, "Roll Over Beethoven">
atom$mat_bool_background_pic <yes>
atom$mat_art_id <1-0-2196>
atom$mat_bool_precise <yes>
atom$mat_precise_width <500>
atom$mat_precise_height <333>
atom$man_start_object <ornament, "Music Forums">
atom$mat_orientation <hcc>
atom$mat_precise_x <9>
atom$mat_precise_y <10>
atom$mat_precise_width <204>
atom$mat_precise_height <18>
.
.
.
atom$man_end_object
```

```
.
.  (The definition of more objects goes here...)
.
atom$man_end_object
```

## atom$mat_precise_y

**81 ($51)**

**atom$mat_precise_y** determines the vertical (y-axis) coordinate in pixels that is used to position children of a precise group. This atom is used in conjunction with atom$mat_precise_x on page 3-151.

### Syntax

```
atom$mat_precise_y <y>
```

| | |
|---|---|
| `<y>` | Specifies the vertical pixel coordinate used to position children of a precise group. Values are **0** or greater. |

### Object(s) Handled

The objects handled by this atom are `org_group`, `dms_list`, `sms_list`, `dss_list`, `sss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

### Return Value

None.

### Example

The following example defines the placement of a dynamic list box to have a precise vertical coordinate of 33 pixels in the Mozart Forum window:

```
atom$man_start_object <ind_group, "Mozart Forum">
atom$mat_bool_background_pic <yes>
atom$mat_art_id <1-0-2196>
atom$mat_bool_precise <yes>
atom$mat_precise_width <500>
atom$mat_precise_height <333>
atom$man_start_object <dss_list>
atom$mat_frame_style <3>
atom$mat_bool_no_border <yes>
atom$mat_color_face <248 232 208>
atom$mat_color_text <64 64 64>
atom$mat_precise_x <10>
atom$mat_precise_y <33>
atom$mat_precise_width <200>
atom$mat_precise_height <248>
atom$man_start_object <trigger, "RockLink">
atom$man_end_object
```

```
atom$man_start_object <trigger, "Critic’s Choice">
atom$man_end_object
.
.
.
atom$man_end_object
.
.  (The definition of more objects goes here...)
.
atom$man_end_object
```

## atom$mat_relative_tag

### 11 ($0B)

**atom$mat_relative_tag** defines the relative ID of the current object. Objects on the same window must have unique relative IDs.

### Syntax

```
atom$mat_relative_tag <dword>
```

| | |
|---|---|
| `<dword>` | Specifies the relative ID of the current object. Values are **0** or greater with a default value of **0**. |

### Object(s) Handled

The objects handled by this atom are `org_group`, `dms_list`, `sms_list`, `dss_list`, `sss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range`.

### Return Value

None.

### Example

The following example defines relative IDs (1 and 2) for objects on a window:

```
atom$man_start_object <trigger>
atom$mat_relative_tag <1>
atom$man_end_object
atom$man_start_object <trigger>
atom$mat_relative_tag <2>
atom$man_end_object
```

## atom$mat_right_spacing

### 188 ($BC)

**atom$mat_right_spacing** sets the right spacing in the Tokyo (Windows) client.

### Syntax

```
atom$mat_right_spacing <x>
```

| | |
|---|---|
| `<x>` | Specifies the right spacing in pixels. |

### Return Value

None.

### Object(s) Handled

None.

### Example

The following example creates a button with 4 pixels of space (padding) on the right:

```
atom$man_start_object <trigger, "Hello">
atom$mat_left_spacing <4>
atom$mat_top_spacing <2>
atom$mat_right_spacing <4>
atom$mat_bottom_spacing <2>
atom$man_end_object
```

# atom$mat_ruler
**18 ($12)**

**atom$mat_ruler** sets one or more tab stops to the ruler of the current object. Each argument indicates one tab stop. If no argument is specified, any tabs that exist on the ruler for the current object are cleared.

This atom is used with list boxes, views, and labels. The text of the object must be embedded with tab characters for this feature to work properly.

## Syntax

```
atom$mat_ruler <[tab_1][, tab_2]…[, tab_n]>
```

| | |
|---|---|
| `[<tab_1>]` | Specifies the first tab stop. A value of **1** is equal to one character of text in the standard system font. |
| `[,tab_2]…[,tab_n]` | Specifies other optional tab stops that can be added in a series. A value of **1** is equal to one character of text in the standard system font. |

## Return Value

None.

## Object(s) Handled

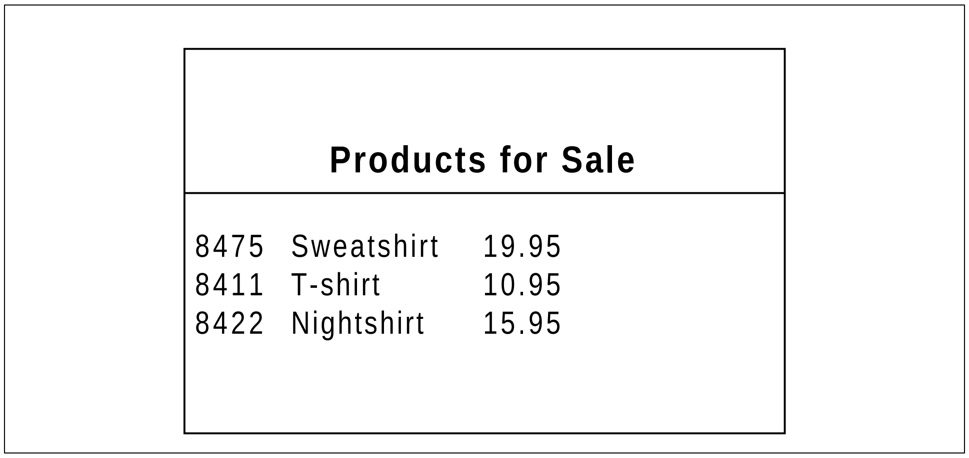
The objects handled by this atom are `dms_list`, `dss_list`, `view`, and `edit_view`.

## Example

The following example sets the tab stops for text on a form:

```
atom$man_start_object <dss_list>
atom$mat_title <"Products for Sale">
atom$man_start_object <trigger>
atom$mat_ruler <7, 15>
atom$mat_title <"8475→Sweatshirt→19.95">
atom$man_start_object <trigger>
atom$mat_title <"8411→T-shirt→10.95">
atom$man_start_object <trigger>
atom$mat_title <"8422→Nightshirt→15.95">
```

The result of this example is shown in Figure 3-3:



**Products for Sale**

| | | |
|---|---|---|
| 8475 | Sweatshirt | 19.95 |
| 8411 | T-shirt | 10.95 |
| 8422 | Nightshirt | 15.95 |

*Figure 3-3: Tab Stops in a Form*

# atom$mat_sage_context_help

## 184 ($B8)

**atom$mat_sage_context_help** enables the "What's This" feature in the help system for the K2 client.

### Syntax

```
atom$mat_sage_context_help <string>
```

| | |
|---|---|
| `<string>` | Specifies the text string. |

### Return Value

None.

### Object(s) Handled

None.

### Example

The following example enables the "What's This" feature in the help system for the K2 client:

```
atom$man_start_object <view>
atom$mat_edit_view
atom$mat_sage_context_help <text>
```

## atom$mat_scroll_threshold
**67 ($43)**

**atom$mat_scroll_threshold** determines how many bytes of text are retained in a view. Once data in the view reaches its maximum capacity (see atom$mat_capacity on page 3-102), some old data is trimmed from the view so new data can appear. This feature is used in the chat area of the online service.

### Syntax

```
atom$mat_scroll_threshold <x>
```

| | |
|---|---|
| `<x>` | Specifies the number of bytes to retain in a view after its capacity is reached. The values are **1** or greater with a default value of **0**. |

### Object(s) Handled

The objects handled by this atom are `view` and `edit_view`.

### Return Value

None.

### Example

The following example specifies a view to trim old data and accept new data when there is a limited buffer:

```
atom$man_start_object <view>
atom$mat_capacity <32000>
atom$mat_scroll_threshold <8000>
```

# atom$mat_secure_field

**170 ($AA)**

**atom$mat_secure_field** is used to mark a field for secure extraction. The atom is also used to set the level of encryption for extracting the field.

## Syntax

```
atom$mat_secure_field <text>
```

| | |
|---|---|
| `<text>` | Specifies a two-byte argument that is a word that is used to indicate the cypher to be used when encrypting the extracted data. |

## Object(s) Handled

The object handled by this atom is `view`.

## Return Value

None.

## Example

The following example tells the display manager to secure a field:

```
atom$man_start_object <view>
atom$mat_edit_view
atom$mat_secure_field <text>
```

## atom$mat_secure_form

### 136 ($88)

**atom$mat_secure_form** forces all input field data on a form that is transferred to the host to be secure.

### Syntax

```
atom$mat_secure_form <x y>
```

| | |
|---|---|
| `<x>` | Specifies the token (two bytes) to send to the host server. The value is a code of two ASCII characters. Wu (57x 75x) defines a security tag for data transmission and is the default token. |
| `<y>` | Specifies the numeric token argument (four bytes), which specifies the security level of the field data transmissions to send to the host server. The default security level is 0 or token argument of 00x 00x 00x 00x. |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example sends a Wu token with an argument value of 00 hex to establish a secure field transmission level of 0. In this example, 57x = W and 75x = u.

```
atom$uni_start_stream
.
.
.
atom$man_start_object <ind_group>
atom$mat_size <A0x 78x>
atom$mat_relative_tag <101>
atom$mat_secure_form <57x 75x 00x 00x 00x 00x>
00x>atom$uni_end_stream
```

# atom$mat_shortcut_key

**68 ($44)**

**atom$mat_shortcut_key** defines the character that can be used by the member as a shortcut key for the current object. A member uses shortcut keys to select a menu item or a button on a window using keys on the keyboard, instead of pointing and clicking with the mouse.

## Syntax

```
atom$mat_shortcut_key <key>
```

| | |
|---|---|
| `<key>` | Specifies the ASCII value of the shortcut key. |

## Object(s) Handled

The object handled by this atom is `trigger`.

## Return Value

None.

## Example

The following example defines the shortcut key:

```
atom$man_start_object <dss_list>
atom$man_start_object <trigger>
atom$mat_title <"Go To">
atom$mat_shortcut_key <g>
atom$man_end_object
```

The result of this example is that members can activate the Go To menu item by pressing ALT-G.

## atom$mat_sink

### 111 ($6F)

**atom$mat_sink** specifies the destination folder where a Favorite Places item is saved. The argument to this atom is the path of the Favorite Places section of the Personal Filing Cabinet (PFC) of the online service. The PFC is used to organize and maintain objects such as mail documents, download files, and Favorite Places (bookmarks).

### Syntax

```
atom$mat_sink <path>
```

| | |
|---|---|
| `<path>` | The path for the Favorite Places section. |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example specifies that the item is semt to the Incoming/ Saved mail section of Favorite Places:

```
atom$man_start_object <ind_group>
atom$mat_sink <"Favorite Places/Incoming/Saved Mail">
atom$man_end_object
```

## atom$mat_size

**23 ($17)**

**atom$mat_size** defines the width and height of the current object.

### Syntax

```
atom$mat_size <width, height>
```

| | |
|---|---|
| `<width>` | Specifies the width for the current object. A value of **1** is equal to the width of one character of text in the standard system font. Values range from **0** to **235**. |
| `<height>` | Specifies the height for the current object. A value of **1** is equal to the height of one line of text in the standard system font. Values range from **0** to **235**. |

### Object(s) Handled

The objects handled by this atom are `dms_list`, `dss_list`, `trigger`, `ornament`, `view`, `edit_view`, `range`, and `select_range.`

### Return Value

None.

### Example

The following example defines a width of 25 and a height of 1 for an `ornament` object:

```
atom$man_start_sibling <ornament, “”>
atom$mat_size <25,1>
atom$mat_font_id <arial>
atom$mat_font_size <18>
atom$mat_font_style <bold>
atom$man_append_data <”keyword”>
atom$man_end_object
```

## atom$mat_sort_order

**69 ($45)**

**atom$mat_sort_order** establishes the sort order of items in a list.

### Syntax

```
atom$mat_sort_order <sort_value>
```

| | |
|---|---|
| `<sort_value>` | Specifies the order in which the list items are sorted. Values are: |

| | |
|---|---|
| `0` | normal—The new item is added at the end of the list (default). |
| `1` | reverse—The new item is added at the beginning of the list. |
| `2` | alphabetical—The new item is added alphabetically. |

### Object(s) Handled

The objects handled by this atom are `dms_list` and `dss_list`.

### Return Value

None.

### Example

The following example sorts items in a list in alphabetical order:

```
atom$man_start_object <dss_list>
atom$mat_sort_order <2>
.
.
.
atom$man_set_context_globalid <32-123>
atom$man_start_object <trigger>
atom$man_end_object
atom$man_start_object <trigger>
atom$man_end_object
atom$man_start_object <trigger>
atom$man_end_object
```

# atom$mat_spacing

## 190 ($BE)

**atom$mat_spacing** adds an absolute pixel spacing to the size of any object in the Tokyo (Windows) client.

### Syntax

```
atom$mat_spacing <w,x,y,z>
```

| | |
|---|---|
| `<w,x,y,z>` | Specifies the pixel spacing to be added to the left, top, right, and bottom of an object. |

### Return Value

None.

### Object(s) Handled

### Example

The following example creates a button with 4 pixels of space (padding) on the left and right, and 2 pixels on the top and bottom:

```
man_start_object <trigger, "Hello">
mat_spacing <4,2,4,2>
man_end_object
```

## atom$mat_style_id

**58 ($3A)**

**atom$mat_style_id** defines the size and position of a window through the use of an ID from the Windows style database. The style ID is normally the global ID of the window. The style ID is preset on Rainman forms.

### Syntax

```
atom$mat_style_id <style_ID>
```

| | |
|---|---|
| `<style_ID>` | Specifies the style ID to use. See the Windows style database for values. The default value is 0, which specifies that no style is used. |

### Object(s) Handled

The object handled by this atom is `ind_group`.

### Return Value

None.

### Example

The following example defines style 5 on a window:

```
atom$man_start_object <ind_group>
atom$mat_style_id <5>
```

# atom$mat_tab_get_cur_sel

**155 ($9B)**

**atom$mat_tab_get_cur_sel** gets the index value of the currently selected tabbed object or the tabbed page in focus.

## Syntax

```
atom$mat_tab_get_cur_sel
```

## Object(s) Handled

The object handled by this atom is `tab_group`.

## Return Value

Index value of the tabbed object in focus. Values are **0** or greater.

## Example

The following example gets the index value of the currently selected tabbed object and stores it in register A:

```
atom$uni_start_stream
atom$man_set_context_relative <11>
atom$mat_tab_get_cur_sel
atom$var_number_set_from_atom <A>
atom$uni_end_stream
```

## atom$mat_tab_set_cur_sel
### 154 ($9A)

**atom$mat_tab_set_cur_sel** puts a tabbed object in focus as the currently selected page or form.

### Syntax

```
atom$mat_tab_set_cur_sel <x>
```

| | |
|---|---|
| `<x>` | Specifies an index value of the tabbed object. Values are **0** or greater. |

### Object(s) Handled

The object handled by this atom is `tab_group`.

### Return Value

None.

### Example

The following example sets the tab control index to 0 and focuses (activates) object 11 as the currently selected tabbed object:

```
atom$man_set_context_globalid <41-6134>
atom$man_set_context_relative <11>
atom$mat_tab_set_cur_sel <0>
atom$man_end_context
```

# atom$mat_text_on_picture_pos

## 135 ($87)

**atom$mat_text_on_picture_pos** determines the position of a button's title and art. This atom is used only for a button object (`trigger`) with a trigger style of text_on_picture.

## Syntax

```
atom$mat_text_on_picture_pos <byte_bit_set>
```

| | |
|---|---|
| `<byte_bit_set>` | Specifies the position of the title and art on the button. Values of this 8-bit argument whose high 4 bits specify the position of the art and whose low 4 bits specify the position of the title are defined as follows: |

`00x`—art_middle_center
`10x`—art_upper_left
`20x`—art_upper_center
`30x`—art_upper_right
`40x`—art_middle_left
`50x`—art_middle_right
`60x`—art_lower_left
`70x`—art_lower_center
`80x`—art_lower_right
`00x`—title_middle_center
`01x`—title_upper_left
`02x`—title_upper_center
`03x`—title_upper_right
`04x`—title_middle_left
`05x`—title_middle_right
`06x`—title_lower_left
`07x`—title_lower_center
`08x`—title_lower_right

By default, the title and art are placed in the center of the button object (position value 00).

Figure 3-4 shows the face of a button object divided into nine areas that relate to the defined position values 0 through 8:

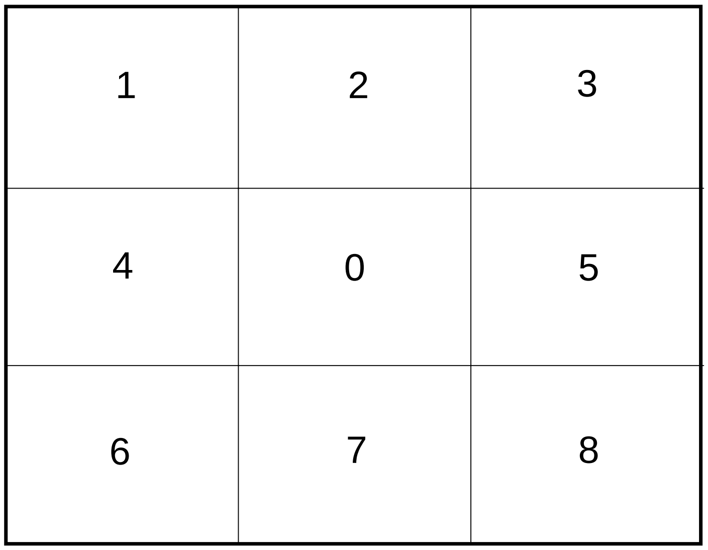


*Figure 3-4: Text or Art Positions on Button*

## Object(s) Handled

The object handled by this atom are `trigger` and `ornament`.

## Return Value

None.

## Example

The following example specifies the position of the text on the lower right portion of the art in the lower center section:

```
atom$man_start_object <trigger> "Test"
atom$mat_trigger_style <8>
atom$mat_text_on_picture_pos 72x
atom$mat_art_id <1-1-31435>
atom$mat_bool_default <yes>
...
atom$man_end_object
```

# atom$mat_title_hint

## 193 ($C1)

**atom$mat_title_hint** returns **wm_gettext** for screen readers even for objects that do not display a title when **wm_gettext** returns nothing.

### Syntax

```
atom$mat_title_hint <string>

atom$mat_text_on_picture_pos <byte_bit_set>
```

| | |
|---|---|
| `<string>` | Specifies the `hidden title to be stored.` |

### Object(s) Handled

The objects handled by this atom are `ind_group`, `dms_list`, `sms_list`, `dss_list, sss_list, trigger, ornament, view, edit_view, range` and `select_range.`

### Return Value

None.

### Example

The following example returns the title for the specified object:

```
atom$man_start_object <ind_group>
atom$mat_title_hint <"welcome">
```

## atom$mat_top_spacing

**187 ($BB)**

**atom$mat_top_spacing** sets the top spacing above any object in the Tokyo (Windows) client.

### Syntax

```
atom$mat_top_spacing <x>
```

| | |
|---|---|
| `<x>` | Specifies the top spacing in pixels. |

### Return Value

None.

### Object(s) Handled

### Example

The following example creates a button with 2 pixels of space (padding) on the top:

```
atom$man_start_object <trigger, "Hello">
atom$mat_left_spacing <4>
atom$mat_top_spacing <2>
atom$mat_right_spacing <4>
atom$mat_bottom_spacing <2>
atom$man_end_object
```

# atom$mat_treectrl_set_ class

## 192 ($C0)

**atom$mat_treectrl_set_ class** is used for tree control to set the `<class_type>`.

### Syntax

```
atom$mat_treectrl_set_ class <class_type>
```

| | |
|---|---|
| `<class_type>` | Specifies a `dword` for the class_type so the object handler registers based on the class type.Values are: |

| | |
|---|---|
| 1 | Class type is `folder`. |
| 2 | Class type is `buddy_online`. |
| 3 | Class type is `buddy_offline`. |

### Return value

None.

### Object Handled

The object handled by this atom is `Tree Control`.

### Example

The following example sets the class type for specified object to `buddy_online`:

```
atom$man_start_object <21> "TreeCtrl"
atom$mat_treectrl_set_class <2>
....
atom$man_end_object
```