Working with Status Bar Controls

This chapter covers the following topics:

- Introduction
- Creating a Status Bar Control
- Using Status Bar Controls without Panes
- Outputting Text to a Status Bar Control
- Sharing a Status Bar in an MDI Application
- Pane-specific Context Menus

Introduction

Note:
Status bar controls are not to be confused with the traditional dialog status bar which is created by selecting the **status bar** check box in the Dialog Attributes window in the dialog editor, or by setting the dialog's `HAS-STATUS-BAR` attribute at run-time. If you are using status bar controls, you should leave the **status bar** check box unchecked and not set the `HAS-STATUS-BAR` attribute.

Creating a Status Bar Control

Status bar controls are created in the dialog editor in the same way as other standard controls (such as list boxes or push buttons) are. That is, they are either created statically in the dialog editor via the **Insert** menu or by drag and drop from the Insert tool bar, or dynamically at run-time by using a `PROCESS GUI ACTION ADD` statement with the `TYPE` attribute set to `STATUSBARCTRL`.

Unlike most other control types, status bar controls cannot be nested within another control and cannot be created within an MDI child dialog. In an MDI application, the status bar control(s) must belong to the MDI frame dialog.

A status bar control may have zero or more panes associated with it. Panes may be defined in the dialog editor from within the status bar control's attribute window, or at run-time by performing a `PROCESS GUI ACTION ADD` statement with the `TYPE` attribute set to `STATUSBARPANE`.

Using Status Bar Controls without Panes

A status bar control without panes offers restricted functionality, because most attributes providing access to the enhanced functionality of status bar controls are only supported for status bar panes. If you wish to do more with a status bar control than simply display a line of text, but don't need to split up the status bar control into multiple sections, you should create a single pane that occupies the full width of the status bar control.

Stretchy vs. non-stretchy panes

If panes are defined for a status bar control, it should be decided whether each pane should stretch (or contract) when the containing dialog is resized, or whether it should maintain a constant width. The former are referred to here as "stretchy" panes, and the latter as "non-stretchy" panes.

There is no explicit flag in the Status Bar Control Attributes window to mark a pane as stretchy or non-stretchy. Instead, any pane defined with a width (`RECTANGLE-W` attribute) of 0 is implicitly assumed to be a stretchy pane, whereas any panes with a non-zero width definition are implicitly assumed to be fixed-width panes of the specified width (in pixels). Because the `RECTANGLE-W` attribute defaults to 0, all panes are initially stetchy when defined in the dialog editor.

The width of a visible stretchy pane is determined by taking the total width available for all panes in the status bar control, subtracting the widths of all visible fixed-width panes, then dividing the result by the number of visible stretchy panes.

Note:
The total available width for all panes normally excludes the sizing gripper, implying that the last pane stops short of the gripper, if present. However, if the status bar control has exactly one pane, and that pane is a stretchy pane, the full width of the dialog (including any sizing gripper) is used.

Outputting Text to a Status Bar Control

Text can be output to the status bar control in one of three ways:

1. For status bar controls with panes, by setting the `STRING` attribute of the pane whose text is to be set.
2. By setting the `STRING` attribute of the status bar control itself, which is equivalent to setting the `STRING` attribute of the first stretchy pane (if any) for status bar controls with panes.
3. By setting the `STATUS-TEXT` attribute of the dialog. This is equivalent to setting the `STRING` attribute of the status bar control (if any) identified by the dialog’s `STATUS-HANDLE` attribute.

Note that the last method is often the most convenient for setting the message text, because it does not require a knowledge of the status bar control or pane handles.

Example:

```
DEFINE DATA LOCAL
01 #DLG$WINDOW  HANDLE OF WINDOW
01 #STAT-1      HANDLE OF STATUSBARCTRL
01 #PANE-1      HANDLE OF STATUSBARPANE
END-DEFINE
...
#DLG$WINDOW.STATUS-HANDLE := #STAT-1
...
#PANE-1.STRING := ’Method 1’
...
#STAT-1.STRING := ’Method 2’
...
#DLG$WINDOW.STATUS-TEXT := ’Method 3’
```

Note:
The dialog editor automatically generates code to set the `STATUS-HANDLE` attribute to the first status bar control (if any). Therefore, the `STATUS-HANDLE` attribute only needs to be set explicitly if you are dynamically creating status bar controls, or if you have defined more than one status bar control in a dialog, and wish to switch between them.

Sharing a Status Bar in an MDI Application

Because status bar controls cannot be created for MDI child dialogs, it is convenient to not have to define multiple status bar controls in the MDI frame dialog. An alternative method is to define just a single status bar, and share it between each child dialog. This can be achieved as follows:

1. Define all possible panes you wish to use in your application within a single status bar control in the MDI frame dialog.
2. Mark all panes as "shared".
3. Export the handles of all panes to corresponding shadow variables in a GDA, so that the MDI child dialogs can access them directly.
4. In the `COMMAND-STATUS` event handler, set the `VISIBLE` attribute of all panes you wish to display for that dialog to `TRUE`. All other panes will be automatically made invisible.

Note:
In the `COMMAND-STATUS` event, you must also set the `ENABLED` state of any commands (signals, or menu or tool bar items which do not reference another object via their `SAME-AS` attribute) associated with the dialog, otherwise they will be automatically disabled. The commands associated with the dialog are all non-shared commands for the MDI frame and all shared commands for the active MDI child (or MDI frame, if no MDI child dialog is active).

Pane-specific Context Menus

Context menus are defined for the status bar control and not per-pane. However, if you wish to ensure that the context menu for a status bar control only appears when the user right clicks a particular pane, you can associate a context menu with the status bar control, but suppress it if the user clicks outside that pane.

Example:

```
DEFINE DATA LOCAL
01 #CTXMENU-1   HANDLE OF CONTEXTMENU
01 #STAT-1      HANDLE OF STATUSBARCTRL
01 #PANE-1      HANDLE OF STATUSBARPANE
01 #PANE-2      HANDLE OF STATUSBARPANE
01 #PANE-3      HANDLE OF STATUSBARPANE
01 #PANE        HANDLE OF STATUSBARPANE
01 #X (I4)
01 #Y (I4)
END-DEFINE
...
#STAT-1.CONTEXT-MENU := #CTXMENU-1
...
DECIDE ON FIRST *CONTROL
   ...
   VALUE #CTXMENU-1
```

```
       DECIDE ON FIRST *EVENT
           ...
             VALUE 'BEFORE-OPEN'
             /* Get click position relative to status bar control
             PROCESS GUI ACTION INQ-CLICKPOSITION WITH
               #STAT-1 #X #Y GIVING *ERROR
             /* Get pane (if any) at specified position
             PROCESS GUI ACTION INQ-ITEM-BY-POSITION WITH
               #STAT-1 #X #Y #PANE
             /* Only show context menu if user clicked in second pane
             IF #PANE = #PANE-2
                #CTXMENU-1.ENABLED := TRUE
             ELSE
                #CTXMENU-1.ENABLED := FALSE
             END-IF
        ...
    END-DECIDE
 ...
END-DECIDE
...
END
```

Note:
If you wish to display a different context menu for different status bar panes, the menu items must be created dynamically in the context menu's `BEFORE-OPEN` event.