V4. Views

As described in previous topics, "views" are a special type of control that contain other controls.   This window contains one view and several controls within the view.   If you were to enter edit mode and drag the view (by clicking in white area at edge of window), then the controls within it would also be moved. Similarly, if a view is hidden, then all of the controls within it are also hidden.   Thus views make it possible for programmers to quickly move, hide, scroll, add, etc., groups of controls within ViewIt windows.

About Control Drivers

Views and controls in ViewIt windows are either supported by old-style CDEFs (control definition resources) or the new "control drivers" introduced with ViewIt.   Control drivers can be much more powerful than CDEFs since they are sent many more events and messages and can store much more information with each control.   Most of the controls found in the Import ("+") menu when in edit mode, for example, are supported by control drivers.

The following control drivers are shipped with ViewIt:

Name	BaseID	Use
BaseVw	1300	views
BaseCt	1310	basic controls
PeekCt	1390	debugging (memory display)
ScrapCt	1800	scrap (clipboard) display
HelpCt	1900	on-line help text
ShellCt	7400	development (see "Override")

Each of these drivers includes example controls that appear in ViewIt's Import menu: "1302 Framed" adds a new view driven by BaseVw, "1312 Static Text" adds a new control driven by BaseCt, "1901 Help Control" adds a new control driven by HelpCt, etc.

Each control driver includes its own on-line help which is available by hitting the "Driver Help" button in the Control dialog (i.e., to learn about a driver, add one of its controls to a window, select it, open the Control dialog, and press the "Driver Help" button).   The type and amount of information presented when this button is hit is defined completely by the control driver.   For simple drivers like PeekCt that supports the "Peek Memory" control, the help presented is quite simple, but for more powerful drivers like BaseCt that support a large number of different control types or more sophisticated controls, the help presented will often consist of multiple topics.

The View Driver

ViewIt views are supported by a special control driver named "BaseVw" which is included with the ViewIt product.   The following topics describe BaseVw and the view-related features that it supports. This info is being presented here (instead of via the "Driver Help" button) since BaseVw is the only driver that is used to support views.

• Operation

BaseVw views are typically added to ViewIt windows from FVEW resources listed in the Import menu. These views should always be type "Static" (set in Control dialog).

Each BaseVw view can contain any number of "daughter" (non-view) controls.   These daughter controls will appear to be drawn above the view's background, but below and within the view's frame.

BaseVw also supports the automatic checking/unchecking of daughter check boxes and radio buttons. Check boxes are checked/unchecked in response to clicks within them or equivalent command key presses.   Similarly, hits in radio buttons cause them to be checked, but also cause all adjacent radio buttons to be unchecked ("adjacent", that is, with respect to their position in the list of controls, not their screen position).   Thus the standard behavior of radio buttons is obtained by simply placing the buttons adjacent to one another in the control list.

• Scrolling

BaseVw takes full advantage of ViewIt's support for "hand scrolling" of control contents (described in the "Controls" topic).   If the Max V or Max H dimensions of the view's Content (set in the Bounds dialog) are not zero, then the view may have a content area that is larger than the visible content (i.e., part of the view may be scrolled out of view, hiding some controls).   If the hand icon in the Bounds dialog is also

checked, then such a view can be hand scrolled by dragging its contents.   Such views can also be hand scrolled when in editing mode (whether the hand icon is checked or not) by OPTION-dragging the control's contents.

   BaseVw also supports the use of scroll bars with any view that has a content area larger than the visible content.   To add a vertical scroll bar when Max V > 0, add 1 to the view's VarCode (set in Control dialog).   To add a horizontal bar when Max H > 0, add 2 to the view's VarCode.   Each bar is drawn as part of the view's frame, so you must also set the view's corresponding Bottom or Right Indent to at least 15 pixels to make room for the scroll bar.   Scroll bars can be used with or w/o hand scrolling, and vice versa.

   When the active state of a window changes, BaseVw will automatically change the active state of the scroll bars in a view if the "Active" flag for the view is set in the "Messages Sent" menu (in Control dialog).

   TIP: If a scrollable view is also attached to a window edge, then make the Max H or Max V corresponding to the side that is attached a multiple of 8 pixels.   This ensures that the window will display the view's full content area when expanded to its maximum size, otherwise a few pixels will be left out of view to support scrolling in increments of 8 pixels (see "Controls" topic for further information about scrolling and attaching controls).

• Instructions
   Special instructions can be placed in the "Instructions" string that is associated with each BaseVw view (set on page 2 of Control dialog).   These instructions have the format,
     a©[control numbers]=[activation condition]
     s©[control numbers]=[show/hide condition]
where "a" denotes an "activation" instruction that is used to activate/inactivate controls, and "s" is a "show" instruction used to show or hide controls.

   The "control numbers" are 2-digit numbers (the number of the control in its parent view) separated by commas or periods, where a period is used to denote a range of control numbers.   The controls designated by these numbers will be activated/inactivated or shown/hidden according to whether the condition is true or false.

   The "condition" is of a set of 2-digit control numbers separated by "+" or "-", where "+" indicates that the corresponding control must be checked for the condition to be true, and "-" that the control must be unchecked.   Also note that controls specified in a "condition" must be check box or radio button-type controls.

   Any number of activate and show instructions can be added to a view's instruction string if each of these is followed by a carriage return.   For example,
     a©07.10,12=+15-16+23
     a©12,44=+34
     s©44=+34
would inactivate controls 7-10 and 12 unless control 15 is checked and 16 is unchecked and 23 is checked (AND logic).   Control 12 can also be made active if control 34 is checked (OR logic), and control 44 will be both inactive and hidden unless control 34 is checked.

   Instructions are evaluated whenever a control in a view is hit (by clicking or command key), and when SetVal is called for all controls in all views (c = d = 0).   Calls to SetVal or SetCtlValue for specific controls do not result in executing view instructions.

• Common Uses
   The "Plain" view with its solid background and no frame is commonly used within simple dialog windows.   TIP: If the plain view is to act as a fixed background in the window,
you can attach it to the right and bottom sides of the window (via Bounds dialog) and lock its position (via "Lock" menu) so that it can't be accidentally moved when in edit mode.

   The "Framed" view is commonly used to support paging of groups of controls (multiple overlapping framed views are hidden and shown in succession using ShoCtl) or to break up the contents of a window into groups of controls.

   The "Scrollable" view offers another way to reduce clutter by supporting the scrolling of groups of controls.   Controls in such views can also be dynamically manipulated using AddCtl (to add controls), DspCtl (to remove controls), and ScrCtl (to scroll or resize view's content area).

   The "Transparent" view can be useful in cases where there is a reason to place one visible view above

another, and the underlying view's background or controls must be seen.

All view types can be optionally "attached" to the right or bottom sides of the window, and in turn can have daughter controls attached to themselves.   This window, for example, has one plain view attached to the bottom and right sides of the window, and a help control attached to the view.   If the window is resized, then ViewIt automatically resizes the view to fit the window and the help control to fit the view.   A complete discussion of "attaching" controls to support growing and zooming is presented in the "Controls" topic.

• Limitations

All controls specified in instructions must be in the same view, and views themselves cannot be shown/ hidden or act./inact. with instructions.   To do the latter, or for creating more complex interactions between controls, use commands such as ShoCtl and ActCtl, or toolbox calls.

BaseVw provides no support for displaying title text, no support for colors beyond frame, body, & content, and no support for data linking, resource linking, control values, and other features common to non-view controls.

• Source Code

The source code and development projects for this driver can be obtained as part of the Inside FaceWare package.