BNCS User Guides

Atos IT Solutions and Services

Skins, Styles and States

BNCS

Contents

1 Introduction

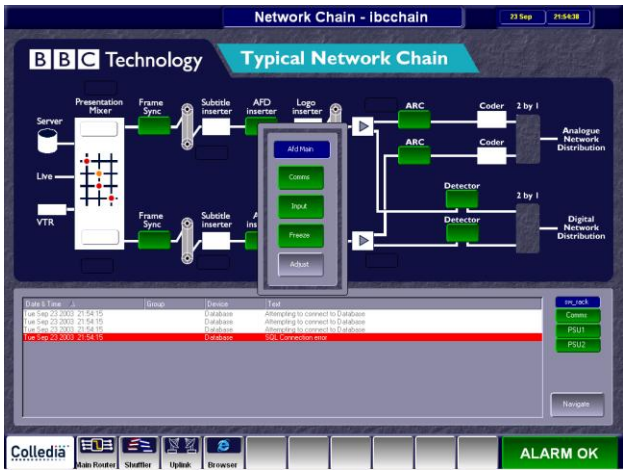
When designing the user interface it is useful to be able to:

- have consistent appearance of the UI components. This helps develop a "visual language" - that means that buttons that do similar jobs have a similar appearance.
- change the appearance of the UI components outside the application that uses them.

Developing a consistent user interface helps with reuse of components between systems.

1.1 An example

Let's take just one element of the following application as an example:



Let's consider the master alarm button which resides at the bottom right of this display.

Here's a closer look – this button shows the current state of the master alarm. To put this control in context, the text changes to "ALARM" and the button turns red when there is an active alarm.

What can we say about this button? It's a green button with "ALARM OK" written on it. Technically correct, but we need to consider it further – there are three things we can pick out.

State: The button is shown is coloured green for "good" with appropriate text on it. The display can change depending upon the state of a particular thing on our system – this button goes red when there is an active master alarm.

If any other alarm buttons appear in this system they will share the same set of colours.

State refers to the dynamic settings of a component.

Style: The button has a larger than normal font as this display needs to be clear. This *style* is unlikely to change between the various states that this button can display.

Style refers to the set-once settings of a component (usually stuff you set-up at the beginning).

Skin: The button is drawn with slightly rounded corners, shaded background colour and against a background of "rock" effect.

Skins are the static stuff that are set for a system and can't easily change.

1.2 Same but different?

Consider this example.

State: The same as the example above.

Style: The same as the example above.

Skin: The edges of the button are different to the above example shaded up from the edge towards a flat-filled text area. The background behind this button is "traditional" Windows grey.

1.3 How the panel is designed – a team effort

When the **panel designer** put this master alarm control on the screen they needed to specify:

- Size and position of the control
- The larger than normal font size (I choose these words carefully)
- The "alarm good" state (note – not actually specifiying green)

A **system designer** has to have specified:

- what the "alarm good" state (and all the others) actually means i.e. green

The **skin designer** specifies

- how the buttons and other components are drawn – the edges and fills
- backgrounds to dialogs – whether flat filled or tiled pixmaps
- the base font face and normal font size

The upshot of all this is that it makes no difference to the panel designer whether the final appearance of the control is the first or second example – both these are generated from the same UI description that the panel designer produced.

1.4 Stylesheets

One of the goals stated at the start of this document was to be able to change the appearance of a system externally to the applications that make it up.

It is no good therefore hard-coding fonts/sizes/colours deep into an application. Instead use a lookup to where that setting is stored.

Stylesheets are a means of translating a named value into individual control settings.

So using our example above (don't worry if the actual settings don't make sense, it's the concept that's important):

Stylesheet entry	Settings
alarm_ok	colour.background=dark green colour.text=white
alarm_active	colour.background=red colour.text=white

Use of stylesheets:

- encourages the consistent user interface by using similar settings for similar things
- is faster than passing individual settings
- allows us to change the appearance of externally to the application

Some controls only allow the use of stylesheets to define appearance.

Stylesheets can actually be used to store any setting at all and so can be used to store styles, states or even complete settings for a control.

BNCS

2 Version Control

Version	Date	Author	Comments
0.1	30/11/03	David Yates	First Version
0.2	19/12/11	Andrew Prince	New format
0.3	18/9/14	David Yates	Removes inaccurate last paragraph (about hard coding colours – this is neither true on real systems nor possible with smart buttons)