symbian

Style Guide

- Dialogs
- Glossary of terms
- Infoprints and messages
- Menus
- Toolbars
- Basic UI principles

Dialogs

Chapter Contents

- General
  - Cosmetic aspects of dialogs
  - Preferences — have separate dialogs for View and General preferences
  - "About" dialogs for 3rd party and add-on programs
  - Show relationships between settings on dialog lines
  - Use "&" instead of "and" on dialog line settings
  - Add explanatory text to dialogs that are not easy to understand
  - Use of quote marks round text in dialogs
  - Don't use "Are you sure?" in confirmation dialogs
  - Put measurement units after edit boxes, not in the title
  - Text and controls should be dimmed when not available
  - Titles should reflect menu command names & be friendly
  - Spacing around punctuation in dialogs
  - Don't show the effects of changes until the dialog is complete
  - Try to avoid buttons if you can
  - Dialog titles: say as much as you need in order to explain the dialog
  - Validation in dialogs — take the user to the line they need to set
- Buttons
  - Eikon doesn't support display of shifted shortcuts under buttons, so...
  - Buttons in text editors in dialogs
  - Use Stop rather than Abort or Cancel in error dialogs
  - Buttons with two words on them
  - Dealing with a mixture of labelled and unlabelled buttons
  - Keep buttons in secondary dialogs in the same position
  - When to specify keynames for buttons and when not
  - Persistent dialogs e.g "New entry" in Data should have a Close button
  - Embedded apps should have a Done button
  - Using an ellipsis or arrow on buttons
  - N buttons must also be activated by N and Esc
  - OK & Cancel buttons in dialogs don't need Enter & Esc under them
  - Use Y/N in dialogs with ? in the title
  - Enter is the "do it" key - not always an exit key
  - Button names
  - Choosing where to position buttons
- Multi-page dialogs
  - Grouping settings & choosing the title of the page
  - With multi-page dialogs (esp. from More buttons), display the most important page first
  - Multi-page dialogs can make your app much easier to use
- Tickboxes
  - Use of active and passive voice on dialog lines
  - About using multi-page dialogs

General

Section Contents

- **Cosmetic aspects of dialogs**
- **Preferences — have separate dialogs for View and General preferences**
- **"About" dialogs for 3rd party and add-on programs**
- **Show relationships between settings on dialog lines**
- **Use "&" instead of "and" on dialog line settings**
- **Add explanatory text to dialogs that are not easy to understand**
- **Use of quote marks round text in dialogs**
- **Don't use "Are you sure?" in confirmation dialogs**
- **Put measurement units after edit boxes, not in the title**
- **Text and controls should be dimmed when not available**
- **Titles should reflect menu command names & be friendly**
- **Spacing around punctuation in dialogs**
- **Don't show the effects of changes until the dialog is complete**
- **Try to avoid buttons if you can**
- **Dialog titles: say as much as you need in order to explain the dialog**
- **Validation in dialogs — take the user to the line they need to set**

Cosmetic aspects of dialogs

If you need to, see the dialogs in the in-built programs for details of the cosmetic appearance of dialogs, e.g. borders, shadows, etc.

Preferences — have separate dialogs for View and General preferences

If you have a sufficient number of preferences that control how the information looks on screen and how the program works (e.g. 3 view preferences and 2 control preferences), you should separate the view preferences from the control preferences. Call the control preferences "General preferences" on the menu tile.

If you only have e.g. one view preference, and one control preference, then just have one "Preferences" command on the Tools menu.

Keeping the view and control preferences separate like this gives you more space to increase the number of preferences in future versions of your program, should you so wish, and also advertises the fact that there are things you can do to change both the appearance and behavior of the program.

"About" dialogs for 3rd party and add-on programs

Some 3rd party programs, and additional programs, will need About dialogs. Where they are to be included, they should be accessible from an "About xx" command which should be placed at the bottom of the Tools menu in the program.

The shortcut for the command should be Ctrl-Sh-A.

The text and/or graphics that appear in the dialog should be determined by the owner of the particular program.

The only style standards we have are that the text should contain details of: —

- Program name and version
- Copyright symbol, copyright holder name and date

All the rest is up to the program owner to determine.

Show relationships between settings on dialog lines

If it's not obvious that a dialog line is tied to / affected by the line above, indent it.

When the setting on a dialog line (let's call it Line B) is related to the setting made on a previous dialog line (let's call this line, Line A), you should show this relationship to the user. There are a number of ways to do it, depending on the relationship involved:

- if the setting on the Line A determines whether or not the user can change Line B, use dimming. Line B should be active only when the appropriate setting has been selected on Line A.

Note: if you have used dimming, but the text that you wish to use to label Line B does not indicate that Line B is related to Line A, you should also indent Line B. E.g. in the Import options dialog in Data, the text "Use" on the line following "Text qualifier" does not make it suffiently clear on first glance that this line is related to the "Text qualifier" line. So the "Use" line is indented to make it clear that this line is related to the line above.

- if there is a dynamic relationship between the Lines A and B such that a change on either line updates the other, then use indents to show the relationship between the two. For example, in Agenda Find dialog between the lines Find range and the From and To lines that follow.

Use "&" instead of "and" on dialog line settings

It looks neater and makes the line of text shorter.

Add explanatory text to dialogs that are not easy to understand

No matter how carefully you choose the words for dialog lines, some dialogs are not understandable at first glance, and others contain jargon or abbreviations that aren't necessarily known by most folks.

In these dialogs, add a short line of explanatory text at the top of the dialog, just to make things a bit clearer.

E.g. in Time in the Add city dialog, we have the line "'GMT offset' is time difference from Greenwich Mean Time"

and in the Add names and Edit names dialogs in Sheet we have:

"Note: Value can be a number, text, cell or range reference".

Use of quote marks round text in dialogs

Where the text is user-generated, e.g. the name of a user's file, or the text string that they have chosen to replace, we should use double quotes. For example, Cannot find "xyz".

Note: filenames that are generated by the machine, e.g. Word(01) should be treated as though the user has entered the name of the file, otherwise there will be inconsistent use of quote marks around the names of files the user can edit.

Where the text is generated by the machine, e.g the name of a program, or a system file that the user cannot edit, we should use single quotes, e.g Show 'System' folder.

Where you are uncertain of the source, e.g for countries in Time, you should treat them as though they are generated by the user.

Don't use "Are you sure?" in confirmation dialogs

Confirmation dialogs should appear in the form: —

Revert to saved?

All changes will be lost

No Yes

Otherwise you have two questions in the dialog, and it's as though you don't trust the user to make the decision without a second confimation in the form "Are you sure?"

Put measurement units after edit boxes, not in the title

In dialogs that have edit boxes for numbers in a particular unit of measure ment, e.g the Indents dialog in Word, put the measurement unit that is currently in use as small text after the edit box.

Reasons being:

- Having the unit, e.g ("in"), in a dialog title doesn't look very tidy.
- Placing the measurement unit on the line to which it applies allows us to have more than one measurement unit to be current in a single dialog, should we ever wish to do this.

Text and controls should be dimmed when not available

When you dim dialog lines when they aren't available, you should also make the controls invisisble as well.

E.g. in Time, the sound setting "Silent for" is dimmed until you have the "Alarm sound" line above set to "Silent for". When the text "Silent for" is dimmed, so should the controls 00:30.

Dialog lines should be dimmed rather than made invisible when they aren't available.

Titles should reflect menu command names & be friendly

If we need to add explanatory text to the dialog title, we probably haven't got the name of the menu command right.

For example, with a dialog to give information about an object that has been inserted in the current file, the title:

"Embedded object info";

Would be much better as "Information on inserted object"

We are then linking the dialog to the name of the commands etc. that are used to insert objects e.g. in Word.

Also, "Embedded" is a bit of a techy term and "Inserted" is more friendly.

Spacing around punctuation in dialogs

We should conform to the usual punctuation and spacing rules in English, ie:

- no spaces between the last word in a sentence and a question mark or full stop
- no spaces between a word and the comma, colon or semi-colon that follows it
- one space between a comma, semi-colon or full stop and the word that follows it

---

Don't show the effects of changes until the dialog is complete

Don't show the effects of any changes the user makes in a dialog, until the dialog has been confirmed, i.e. don't have the screen redrawn behind the dialog.

For example, Data has dialogs which allow you to change the layout of columns and the fonts used. We decided not to show the effects of any change the user makes to the layout and font until the dialog has been confirmed and removed.

The reasons for this are:

- It would be unclear whether the changes would remain if you hit Cancel instead of confirming the dialog.
- It does not pass the "cup-of-tea" test. i.e. you make some changes, leave the dialog up, come back (after making some tea) and cancel the dialog — and you may be surprised that the screen reverts to a different setup (the original).

  In this case, you can only see those changes which affect that part of the screen that is showing so it's not really useful.

- It's inconsistent with the behavior of any other dialogs in the system.

Try to avoid buttons if you can

(other than Cancel and OK). They're OK, but they're not great. Use e.g. multi-page dialogs instead, where possible.

Buttons are not intuitive to keyboard users — the keypresses they require often seem random or unrelated to the action.

Dialog titles: say as much as you need in order to explain the dialog

Many dialogs won't need anything other than the command name. Use longer dialog titles wherever it helps explain what the dialog's for.

Other info:

Windows 95 just uses the name of the menu command. But it has one line of explanation on the status bar at the bottom of the window, which we don't. Even with this hard-to-see "status info", though, Microsoft might have been better with more explanatory dialog titles.

Validation in dialogs — take the user to the line they need to set

If the user presses Enter in a dialog and fails to set a line that is validated before the dialog can be closed, you need to:

1. Display a message saying e.g. "No xx name entered".

2. Move the cursor to the line that the user must set.

If you have a multi-page dialog and the user has failed to set several lines, display the message and take them to the first line to set, then when they press Enter again, display a message and take them to the second line to set, and so on.

Buttons

Section Contents

- **Eikon doesn't support display of shifted shortcuts under buttons, so...**
- **Buttons in text editors in dialogs**
- **Use Stop rather than Abort or Cancel in error dialogs**
- **Buttons with two words on them**
- **Dealing with a mixture of labelled and unlabelled buttons**
- **Keep buttons in secondary dialogs in the same position**
- **When to specify keynames for buttons and when not**
- **Persistent dialogs e.g "New entry" in Data should have a Close button**
- **Embedded apps should have a Done button**
- **Using an ellipsis or arrow on buttons**
- **N buttons must also be activated by N and Esc**
- **OK & Cancel buttons in dialogs don't need Enter & Esc under them**
- **Use Y/N in dialogs with ? in the title**
- **Enter is the "do it" key - not always an exit key**
- **Button names**
- **Choosing where to position buttons**

Eikon doesn't support display of shifted shortcuts under buttons, so...

If you have a button in a dialog for a standard program command that has a shifted shortcut, e.g a Font button, you should allow the button in the dialog to be activated by the standard shifted shortcut for the command, and its unshifted equivalent. You should display the unshifted shortcut as the legend under the button.

E.g. a Font button in a dialog should be activated by both Ctrl+F and Sh+Ctrl+F, though Ctrl+F should be displayed as the legend under the

button.

Buttons in text editors in dialogs

Examples of this are in the text page of the Create new day entry dialog in Agenda, and in the header & footer dialogs in Page setup.

All the buttons beneath the text box should be right next to each other so that they form a toolbar.

All the buttons should be activated by the standard shortcuts for those functions. You can only indicate the shortcut under the button if the shortcut is an unshifted one. Eikon does not currently support the display of shifted shortcuts under buttons in dialogs.

Standard buttons should be activated as follows:

In Agenda where there are B I U Font Object Format (format object) buttons in the text edit box in the edit entry dialog:

- Bold Ctrl-B
- Italic Ctrl-I
- Underline Ctrl-U
- Font Ctrl-F

  Ctrl-F should be indicated under the button

- Object Ctrl-O

  Ctrl-O should be indicated under the button

- Format object Ctrl-J

  Ctrl-J should be indicated under the button

In Header/footer text editors where there are B I U Font Format > Insert >

- Bold Ctrl-B
- Italic Ctrl-I
- Underline Ctrl-U
- Font Sh-Ctrl-F (Not Ctrl-F because this is the shortcut for finding text)

  Keypress isn't advertised under the button

- Format displays its list of commands with Ctrl-M

  Ctrl-M is indicated under the button.

- Insert displays its list of commands with Ctrl-S

Commands that cascade from these buttons do not currently have shortcuts in Eikon — this is because the shortcuts are reserved for buttons that may be required in the dialogs.

Any developer who wishes to give these controls shortcuts in their own programs will need to provide their own customised controls. If they

do this, they should follow the conventions and use the standard shortcuts that are used for the menu commands that are associated with these functions. So they might add:

Format commands —

- Borders/Colours by Sh-Ctrl-D
- Alignment by Sh-Ctrl-A
- Tab positions by Sh-Ctrl-Y

Insert commands —

- Filename Ctrl-N (N chosen because F is used and N for new file has association with file)
- Page number Ctrl-P (P free and for "page")
- Number of pages Ctrl-G (N not free, G in "pages")
- Current time Sh-Ctrl-T and Ctrl-T (note: both of these because we use Sh-Ctrl-T in Time to set the time)
- Current date Ctrl-E (not D because already used for Borders/Colours)

Use Stop rather than Abort or Cancel in error dialogs

In dialogs where not retrying can cause data to be lost, e.g. in disk error dialogs, or an action to remain unfinished, use "Stop" and "Retry" as the buttons, not "Abort" & "Retry". Reasons:

- "Abort" is a word we have decided not to use.
- "Cancel" implies that you will go back to the situation you were in before, and you won't because data will have been lost.

Buttons with two words on them

If you have two words on a button, don't make the button wider, just put the second word on a new line under the first to make a "deeper" button. We should avoid using buttons of different widths, although in some places we may not be able to avoid it — e.g. buttons like Slot definitions in Agenda.

Dealing with a mixture of labelled and unlabelled buttons

Dialogs with more than 2 buttons will typically have a mixture of labelled and unlabelled buttons. If the buttons are placed along the bottom of the dialog (which we've tried to avoid) it looks odd to have e.g. More Cancel OK

Ctrl-M

Ideal solution is to have vertical buttons down the RHS and create a bit of space between the labelled one(s) at the top and the Cancel OK ones at the bottom.

If you have to keep the buttons along the bottom of the dialog, do the same thing — move the Cancel OK buttons to the right hand side and keep the labelled one(s) to the left/middle.

Keep buttons in secondary dialogs in the same position

If you have buttons across the bottom in the first dialog, make sure that they're across the bottom in any secondary dialog that is displayed

from a button in the first dialog. If you have them down the RHS in the first dialog, make sure they're in this position in the second one too.

Important note: don't have them in the same position pixel for pixel though, otherwise you might not realise you'd pressed the button in the second dialog and quit the first one when you didn't mean to.

E.g. "Create new day entry" dialog in Agenda has More... Cancel and OK, and Cancel and OK in the dialog you access from the More... button. Placing the buttons in the same location in both dialogs improved appearance and made them easier to use.

When to specify keynames for buttons and when not

If you have just two buttons in a dialog e.g. Insert text and Cancel, don't put a legend under the Insert text button unless the key you need to press to activate it isn't Enter. Do not put Esc under the Cancel button either.

Always specify keynames for buttons that use other than Esc and Enter.

Persistent dialogs e.g "New entry" in Data should have a Close button

Cancel would make the user think that all the entries they'd added will disappear. Close chosen rather than Finished because it fits better on the button. Also the same word that is used instead of exit in the programs.

Embedded apps should have a Done button

This is to confirm to the user that any changes they have made will be saved. If we use Cancel they might think they were thrown away.

Using an ellipsis or arrow on buttons

For consistency with menu commands, we use ellipses on buttons that display a further dialog. This rule applies to buttons in dialogs, but not buttons on toolbars. There should not be a space between the last character of the button name and the ellipsis. Note: be sure to use a proper ellipsis character and not three dots (the ellipsis character is narrower).

That way, users will know that whenever they see an ellipsis, the command/button (whatever) doesn't perform an action on selection, but presents a dialog requiring further information first.

Buttons that display a choice list should have an arrow. The arrow should point to the right if the choice list appears to the right or left of the button, or point down if the choice list appears below the button.

N buttons must also be activated by N and Esc

Y/N buttons in dialogs (used instead of OK/Cancel in dialogs where the title ends in a ?) should be activated by Y/N, and N should also be activated by Esc. Note that Y should not be activated by Enter because it is too easy for users to make a mistake in confirmation dialogs that use Y and N.

The use of these keys is fundamental to computer UIs, so we can assume users will know which keys to press.

So...don't add the grey explanatory text underneath them.

OK & Cancel buttons in dialogs don't need Enter & Esc under them

Enter is the "do it" key for dialogs and Esc is the "remove dialog and don't do it" key.

We assume that users know this — it's fundamental to computer UIs.

Consequently we do not need explanatory text underneath OK and Cancel buttons in dialogs, unless the keys that you need to press are other than Enter and Esc.

Use Y/N in dialogs with ? in the title

Note that as a matter of style, any dialog whose title ends in a question mark, must have Yes/No buttons, and not "OK/Cancel" buttons.

Enter is the "do it" key - not always an exit key

Enter is the DO IT key in dialogs, but this does not always have to be a dialog completion key. Imagine a dialog which was exclusively for entering as many numbers as the user wanted to enter. You might decide to use Enter for "Enter number", and have a "Done" exit button (probably the Esc key). In the Series 5 software, Enter is reserved on occasion for "carriage return".

(You might decide no overall "Cancel" action is required in this situation, or Esc might produce a Save Changes confirmation, for example.)

Button names

Always have standard exit button names. 90% of dialogs should just have OK/Cancel (at the bottom right). Alternatives which should cover most other dialogs are:

"Yes"/"No"

"Done" (which is better than "Close", which means "close this window/dialog thing" : many users don't think of it as a "thing"),

"Continue"

"Stop". ("Cancel" is so commonly used to mean "don't do it", that if it's actually a "stop doing it" action that you're providing — e.g. half-way through a multi-file copy, where you can't actually "cancel" it as some files will have been copied — "Stop" is better.)

"OK" is not always OK! If you're saying "Program is dead", or "your foot is on a landmine and will now be blown off", you want [Continue] or [Click here] or [Sorry] (or maybe [<expletive deleted>]), but not [OK]!

Watch out for double negatives between text and buttons - as in "Cancel changes? [OK] [Cancel]". What does "[Cancel]" do there?

Except in rare circumstances, don't change the meaning or wording of a button or keypress on a dialog. It would have to be intuitive to the user for you to make such a change — intuitive enough to override the general consistency principle.

(By the way, even "OK" isn't safe, really. OK just says "OK", after all, not e.g. "Do it" or "Accept changes". I told someone once to look at the "Restore" dialog in PsiWin — I just wanted to show him how it worked — and when he'd finished looking at it, he hit "OK", because, OK, he'd seen it and understood it. It's another reminder of how far "advanced" we in the industry are in computer use compared to normal people — ie how far removed from their everyday operation and understanding and terms of reference. It was saying "OK?" and he said "Yes, OK". There is a case for using e.g. "Do it" and "Accept changes" for the two main things "OK" is commonly used for. But I doubt we could come up with a system-wide solution which didn't confuse novices sometimes, and patronise PC users most times.)

Choosing where to position buttons

If you have Cancel and OK buttons (which the vast majority of dialogs do), they should usually go as a pair at the bottom right, i.e.:

Cancel

OK

Other buttons can either go above OK/Cancel (starting at the top right of the dialog and working downwards).

You can instead position buttons in a row across the bottom of a dialog, if you've a good reason — e.g. you need the full screenwidth for the rest of the dialog, or the dialog looks too unbalanced with buttons on the right. If you do this, they should be in this order:

Other buttons Cancel OK

On a multi-page dialog all exit/confirm buttons (ie those which exit the dialog) must be outside the pages (Cancel must cancel *all* settings made on a multi-page dialog). Those which apply to a particular page must be on that page.

Multi-page dialogs

Section Contents

- **Grouping settings & choosing the title of the page**
- **With multi-page dialogs (esp. from More buttons), display the most important page first**
- **Multi-page dialogs can make your app much easier to use**

Grouping settings & choosing the title of the page

We should group settings so that related settings appear on the same page, and give that page a title that sums up the settings on the page. For settings on a page that aren't related and that can't be included on a specialized page, use the term "Other" as the title of the page.

E.g. in Agenda we have Text, Details, Alarm and Other as the titles of the pages in the "Edit day entry details" dialog.

With multi-page dialogs (esp. from More buttons), display the most important page first

For an example, see the More button from the Edit day entry dialog in Agenda, displaying the Alarms page first. This is the page that most users would be likely to need next, having filled in the basic details in the previous dialog.

Multi-page dialogs can make your app much easier to use

Putting lots of things on one screen/dialog is usually counter-productive — the more there are, the more novices and intermediates get confused. Even just one setting which they don't understand may put them off using the dialog.

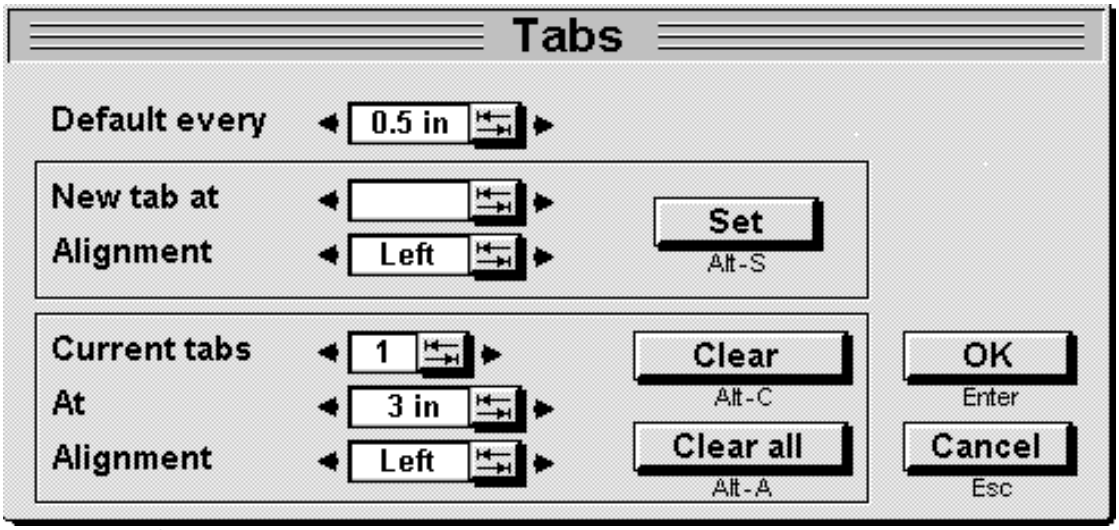
So if a dialog contains a lot of info, try to use a multi-page dialog, and move less important, less commonly used, or jargon-like/unintelligible settings off the first page.

Like almost everything else, a dialog should focus on the everyday things users want to do, and express things in terms the user understands. Try to make sure your first page always concentrates on this.

(This consideration almost always outweighs saving power users one more click / keypress. The only time it doesn't is when dialogs are already entirely to do with technical features which normal people won't be using.)

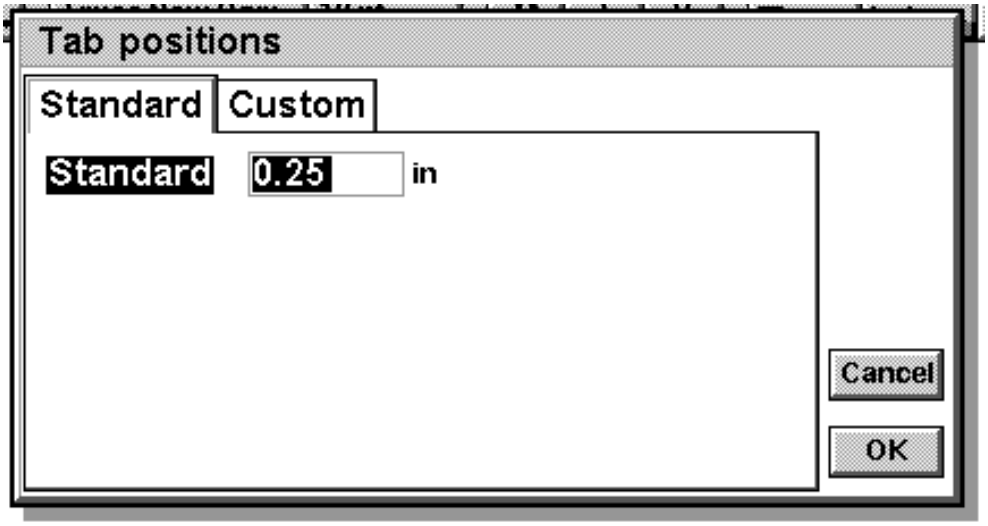
Example: Set Tabs

In looking at the "Set Tabs" dialog we realised that Windows versions of this are hard to understand; all that most people should ever use is the "Tabs every" setting, instead of thinking they're supposed to hand-set each position (or failing to understand the whole thing at all). So instead of e.g:



An example of a bad "tabs settings" dialog

we instead used:



An example of a good "tabs settings" dialog

...ie with all the custom tab setting hidden away on a second page.

Hopefully you can see the difference between the above two diagrams in ease of use for normal people.

Tickboxes

Section Contents

- **Use of active and passive voice on dialog lines**
- **About using multi-page dialogs**

Use of active and passive voice on dialog lines

Where you are changing the setting of an individual item, e.g. the "tentative" status of an entry in Agenda, you should use the passive voice, e.g. here we have:

Entry is tentative

Where you are changing a "global" setting or preference, you should use the active voice, e.g. in Agenda preferences we have:

Show duration or end time

The active voice in preference dialogs etc. matches use of the active voice in menu commands e.g. Show toolbar.

About using multi-page dialogs

Don't have too many page titles. The combined width of titles must fit within the width of the dialog.

Glossary of terms

Chapter Contents

- If you intend to translate, compile a glossary of major terms as you develop the program
- Basic terms and usage
- Keep a glossary to help you make your UI consistent and usable!
- Terms we've decided NOT to use and what to use instead

If you intend to translate, compile a glossary of major terms as you develop the program

It makes localisation much quicker if you send translators a glossary of the main terms in a program, prior to the resource file being finalised.

That way you can also check terms against the standard glossary for Series 5 programs.

Basic terms and usage

For computer-related terminology, which has no obvious everyday metaphors, standardising with other systems is the best way to improve recognition. So we use these terms:

- "PsiWin" not "Psion Explorer" for the program that runs on the PC
- "Docking cable" not "3Link", "serial link cable" or "link cable"
- "arrow keys" not "cursor keys"
- "handles" for the square blobs that appear on a selected embedded object that allow you to change the size of the object.
- "List of open files and programs" not "task list" for the list of open files and running programs
- "toolbar" not "status window" for the row of buttons and the clock on the right hand side of the screen
- "Program icons" for the buttons that take you to the different programs, not "buttonbar"
- "Command icons" for the buttons along the left edge of the screen, not "sidebar"
- "IrDA" not "IRDA" or "irda"
- "Infrared" not "infra red" or "infra-red".
- "menu bar" not "menubar"
- "toolbar" for the buttons on the right hand side of the screen
- "top toolbar" not "toolbar" or "toolband" for the buttons along the top edge of the screen
- "command" or "menu command", not "option", "menu option" or "menu item"
- "menu cascade" not "cascading menu"
- "dialog" not "dialog box" or "box"
- "tabs" on the top of multi-page dialogs that take you to the different "pages" in the dialog
- "dialog lines" not "dialog items"
- "shortcut key", not "hot-key" (e.g. for Ctrl+X, Ctrl+C. Key combinations are usually written with a "+" nowadays, e.g. CTRL+F rather than CTRL-F.)
- "mnemonic", not "accelerator" (e.g. for Alt+F Alt+O to get File|Open: it's a poor word but, don't worry, it's very rarely used/seen)
- "file", not "document"
- "filename" not "file name"
- "folder", not "directory"
- "disk", not "drive"
- "Internal disk" for the disk built into the machine
- "Memory disk" for the plug in cards you can get

- "program", not "application"
- "object", not "document" or "file" for embedded files

Windows users will pick up the above much faster than the alternatives.

Use the word "tap" not "click" for the pen tapping the screen. (This also helps guard against thinking in terms of mice.)

Key names are usually provided for you at the system level. But where they're hand-entered (as in Help files, say), be sure to use the text exactly as it's written on the keyboard - ie "Ctrl", "Fn", "Shift" and "Tab". Note also that it's "Menu", not "Alt".

For multi-word dialog lines and menu commands, use one initial capital letter only - eg "Create new agenda", not "Create New Agenda". (This is like the 3a, not Windows. It's widely preferred.)

Files are usually referred to as app name (in quotes if possible) plus "file" — '"Data" file', '"Word" file' etc. Occasionally you may find exceptions — if it's much clearer to use a different name everywhere, that's OK, but bear in mind people will be used to seeing app name plus "file" everywhere. (One example is the "Insert" menu, where you want to insert a new "Recording", say. Here there's no "file" involved, just a new object.)

Use "entry" for what is often called a "record" in the database.

Keep a glossary to help you make your UI consistent and usable!

A simple way to improve usability hugely, for little effort, is to generate a glossary of intelligible terms, check it with other people, then stick to it (This sentence was written back to front to reflect the current status quo in the software industry. A more "sensible" way to write it would be "a simple way to make your app unusable, quickly, is NOT to bother generating a sensible glossary of intelligible terms, or not to stick to it.").

Please keep a text file listing the words you've chosen to use (and publish it for comment):

You'll often catch yourself using different words for the same thing; just use one. (When a normal user is used to one word, then sees another, they think not surprisingly that it must refer to something else).

Try to use intelligible English words, instead of industry jargon, as far as you can. For instance, the Series5 Data app uses the term "entry" (not "record"). People can understand this. Imagine a normal person being given the task of designing how a computerised phone-book should work. Would they think "I'd better think up some unintelligible term to describe each thing I put in this phonebook"?

Terms we've decided NOT to use and what to use instead

Term	Reason for avoiding it	Alternative
abort	Too techy. Also unpleasant connotations in English.	"stop" or "cancel", depending on context.
application		"program"
Are you sure?		Remove completely and make sure title of confirmation dialog is a question and ends with "?"
default	Too techy.	Better as "standard", or "preferred" (in Time e.g. "preferred alarm time")
directory		"folder"
drive		"disk"
embedded file		"object"

exit/exited		"close/closed"
hide/hidden	Hide seen as a negative action.	Turn the whole thing round so instead of having options to hide things, use options to "show" things. Except for e.g. hidden files
jump to		"go to". Matches similar options in Windows apps.
just	Does not work in some places, so we chose "only" for reasons of consistency.	"only".
loading		"opening"
location (on screen)	Could be confused in e.g. World with location on map.	"position"
path		"folder"
picture		"sketch" if you are inserting a file that was created in the Sketch app. Matches name of application in which pictures are created and so points the user to the correct program to create them.
record		"entry"
search		"find" in menu commands. Matches similar options in Windows. But "search for" in the Find dialogs.
status window		"toolbar"
task		"file" or "program", depending on context
task bar		"list of open files"
task list		"list of open files"
toolband		"top toolbar"

Infoprints and messages

Chapter Contents

- Punctuation in infoprints and messages
- Where you try to give advice, don't use "Check if the..."
- With short messages, remove the verb, unless...
- Don't use "Busy" when you can be more specific
- Avoid use of pleasantries - don't say please this, please that
- Don't try to explain the unexplainable
- Use of quote marks round text in messages
- Keep messages short and to the point
- Messages to use when commands and features aren't available
- Use "No xx name entered" when user must enter a name before pressing Enter
- Make error messages meaningful
- Messages should appear in the top right corner of the screen
- Don't use error/status messages you don't need (Answerphone)
- Have simple indications of goodness and badness (Virus Scanner)
- Use different messages for single and multiple files, etc.

Punctuation in infoprints and messages

Don't use a full stop at the end.

Do use commas and hyphens in the line of text.

Try not to use a full stop if the line of text comes in two parts; use a hyphen to separate the two parts instead. For example:

Instead of:

Cannot paste. Copy and paste areas are different

Use:

Cannot paste — copy and paste areas are different

Where you try to give advice, don't use "Check if the..."

Just state the possible causes of the problem.

For example, for the message

"Problem with Infrared

Check that the serial port is not in use, and that there is enough free memory"

would be better as:

"Problem with Infrared

Serial port may be use, or there is not enough memory"

With short messages, remove the verb, unless...

...it helps to remove ambiguity.

For example, the message "Disk is not present" is equally clear as "Disk not present" without the verb.

However, the message "Filename is too long" could be ambiguous if it were reduced to "Filename too long", so this message should retain use of the verb.

Don't use "Busy" when you can be more specific

For example, if the user is having to wait because the program is searching for something, display a "Searching..." message; if the program is recalculating, use "Recalculating...".

Avoid use of pleasantries - don't say please this, please that

Rather than saying "Please enter a number", just say "Enter a number".

Users will get irritated by having to read a large number of "please"s, and this also makes for shorter text in other languages.

Don't try to explain the unexplainable

For example, when you cannot paste what you have copied because the place you wish to paste does not accept the type of information you have copied you should not use a message like:

"Cannot paste — different type of data required here"

You should just use the message:

"Nothing to paste"

Explaining why certain things occur, or don't occur, is not something that you should use infoprints and messages for. You should just use

them as an on-screen confirmation that something has happened, or not happened.

Use of quote marks round text in messages

Where the text is user-generated, e.g. the name of a user's file, or the text string that they have chosen to replace, use double quotes. For example; "Diary" found.

Note: filenames that are generated by the machine, e.g. Word(01) etc. should be treated as though the user has entered the name of the file, otherwise we'll have inconsistent use of quote marks around the names of files the user can edit.

Where the text is generated by the machine, e.g. the name of a program, or a system file that the user cannot edit, use single quotes, e.g. `Word' not present.

Where you are uncertain of the source, e.g. for countries in Time, you should treat them as though they are generated by the user.

Keep messages short and to the point

Remember that they are only on the screen for a short amount of time, so users will not be able to read and understand a message that's very long.

For example, the message: "The disk has been removed — please replace it", can be shortened to "Disk is not present" without any loss in meaning for the user.

Messages to use when commands and features aren't available

If the feature is *never* available in the program, e.g. Infrared in Time, use the message:

"This item is unavailable"

If the feature is not available *at this time* in the program, e.g. pasting when you have not yet copied or cut anything, use the message:

"Nothing to xx"

e.g. Nothing to paste

If the feature is not available because the program is busy doing something else, e.g. in Record, commands are disabled while playing sound, use a message like:

"Not available while playing sound"

Use "No xx name entered" when user must enter a name before pressing Enter

xx refers to the thing that is to be named e.g. "No folder name entered".

This message should appear at the top right hand corner of the screen.

Make error messages meaningful

Try to indicate exactly what has gone wrong.

E.g don't give a message like "invalid data" when the cause of the problem is that the user has set a combination of margins that is too large for the page size that she has set to print to. Display a message like:

"Combined margins too large for paper size"

That way, she will see that she either has to change the margins or the paper size in order to solve the problem.

Messages should appear in the top right corner of the screen

E.g. static Infoprints like "Entry copied". This is because this is the place that folks are likely to notice them.

Busy messages on the other hand, should appear in the bottom left corner of the screen. The reason for this is that we can't have both types of messages appearing in the same place on screen, and you are likely to move your hand away from the screen while busy messages are displayed (you can't perform any further actions while these messages are on screen).

Don't use error/status messages you don't need (Answerphone)

My answerphone has a classic "stupid" status message. By default it speaks the date/time after each message, but there's an option to turn this "time announcement" off — presumably to get through the messages quicker. However, if you turn it off, then after it plays *each* message, it instead says "the Time Announcement is Off"! Needless to say, this takes about the same length of time as saying e.g. "Monday, 2:35pm".

Have simple indications of goodness and badness (Virus Scanner)

Here's what our Virus Checker said to me when I scanned a file:

```
C:\> SCAN C:\X.TXT
Virus data file  V9511 created 11/15/95  12:47:11
No viruses found in memory.
Scanning C:\X.TXT [MS-DOS_62]
Summary report on C:\X.TXT
File(s)
        Analyzed: ..............       1
        Scanned: ...............       0
        Possibly Infected: .....       0
Time: 00:00.00
```

If a program is trying to say e.g. "Everything's fine", or "You have a virus!", it had better make sure it's clear which it's saying. Unfortunately, here the phrase "Possibly infected" leaps out at the user. Unless you're a wizened old programmer, it makes you think "What?!", every time you see it! It is failing the principle of having simple indications of goodness and badness.

(There's a zero which is supposed to be related to it, but it so far over to the right of the screen you can't really tell. Even if said "Possibly infected: 0", though, that's still not very good - the phrase still leaps out and makes you think "What?!".)

Other points:

- It analyzed 1 file and scanned 0, apparently. Why didn't it scan the file then? Look at the command line - why won't it scan the %£&@$ file I asked it to scan?! (This is undoubtedly some program-specific use of the words "analyzed" and "scanned" - but that's no reason to throw them as-is at normal users, to whom it just looks like a mistake.)
- "Time: 00:00.00" - what's that mean? It's reset the PC to midnight? It scans at infinity files per second? What use is it anyway, even if it was saying a non-zero time? Since other apps don't tell you how long they took to run, it just makes people think this app must have some special time-critical nature which they don't know about.

Use different messages for single and multiple files, etc.

For instance if you want to display a message or status bar that says:

xx file(s) selected

You should in fact have two messages:

1 file selected

and

xx files selected

so that you don't end up with the ugly and inaccurate message 1 file(s) selected.

Menus

Chapter Contents

- Cascaded menus
  - Don't use More as the name of a cascade unless you really have to
  - Cascaded menus often look best at the bottom of menus
  - Keep the text as short as possible to avoid cascade weirdness
  - Commands that will be frequently used *must* have shortcuts
  - Commands must be displayed even when the cascade is dimmed
  - Print cannot be a cascaded menu when it's the first option on the menu
  - When to use cascaded menus
- Command standards
  - "More" menus in programs should contain...
  - Use "&" instead of "and" in menu commands
  - Gray out commands that you can't use
  - When to use lines to separate commands, and when not to...
  - Infrared should be a cascaded command on Tools
  - Place frequently used commands at the top or bottom of a menu
  - Actions (Zoom etc.) with a few settings should cycle not stop
  - For commands that cycle between settings...
  - Apps should have separate Import and Export
  - All apps should have a Switch view command using Ctrl-Q
  - Using ellipses for commands that display a dialog
  - No more than 8 items on a menu, unless you really have to
  - The difference between Settings, Options and Preferences
  - Capitalisation in command names, dialogs etc.
  - Giving clues about dialogs and cascaded menus
  - Graying out menu commands
  - In general, don't change the names of menu commands when view changes
- Shortcut keys
  - Standard menu items and their shortcuts
  - Don't display the shortcut in a menu tile when..
  - Don't include shortcuts for commands that cascade from buttons
  - Choosing shortcuts — use the standard ones for standard commands, then...
  - All commands should have shortcuts, if possible
  - No numbers to be used in menu shortcut keys
- Help
  - Help for 3rd party programs will be available from the Tools menu

Cascaded menus

Section Contents

- Don't use More as the name of a cascade unless you really have to
- Cascaded menus often look best at the bottom of menus

- **Keep the text as short as possible to avoid cascade weirdness**
- **Commands that will be frequently used *must* have shortcuts**
- **Commands must be displayed even when the cascade is dimmed**
- **Print cannot be a cascaded menu when it's the first option on the menu**
- **When to use cascaded menus**

Don't use More as the name of a cascade unless you really have to

If you can find a better name for further commands, do. More is a "last resort" term.

E.g. instead of having a cascade like:

More >

>Edit sketch

>Edit voice note

>Edit memo

You could have:

Edit object >

>Sketch

>Voice note

>Memo

Cascaded menus often look best at the bottom of menus

If you have difficulty with the position of a cascade in the middle of a menu visually, try putting it at the end, especially if it contains commands that toggle and that are likely to be frequently used.

Keep the text as short as possible to avoid cascade weirdness

Cascaded commands should be kept should short, and not contain "Create new this, Create new that, Create new other", but instead contain just the closing half of the item, as in:

Create new > This

That

Other

This is to avoid the situation where the cascade is forced to pop up on the left of the menu rather than the right, because of the length of the text in the cascaded commands.

Menu cascades should be thought of like Option buttons in a dialog. You do not re-state the prompt for each item.

Menu tiles should not be too wide either.

App authors should consider re-arranging menu tiles, combining menu options, using multi-page dialogs, etc to avoid the problem.

However this problem is fixed, it is never going to be as good as avoiding the problem in the first place.

Commands that will be frequently used *must* have shortcuts

Otherwise they are too difficult to use in practice.

Commands must be displayed even when the cascade is dimmed

For example, in a read-only Agenda file, the Create new entry cascade on the Entry menu should still display its commands even when the file is read-only, and the commands are dimmed. The user must be able to see what all the commands are, even if she cannot use them at this moment, otherwise it appears that commands disappear and then reappear later.

Print cannot be a cascaded menu when it's the first option on the menu

For example, on the File menu when viewing an embedded document you might have just 2 options: Print and Close. Since you can't have a cascaded command as the first one on the menu, and it would look odd anyway, all Print options should appear on the top level here.

When to use cascaded menus

Consider cascaded menus wherever you have non-trivial numbers of commands on a menu, and some of them are rarely-used or hard-to-understand commands.

Because otherwise: —

- everyday things get hard to find in amongst the other commands;
- people feel scared by all these commands they don't understand.

Cascaded menus are best kept towards the bottom of a menu tile, to stop menu users being shocked by the appearance of the cascade while arrowing their way down to a non-cascaded command they want.

We must ensure that cascaded commands are still as accessible by mnemonics, of course, and shortcut-keys where appropriate.

Don't get *too* keen with cascaded menus as there are drawbacks, such as people unable to find commands & having to go into every submenu to find them, and menu users finding it all too fiddly.

Also, always consider — can you not use ONE command with a multi-page dialog instead of several commands?

Notes —

- The terms used on cascaded menus should be brief — ideally a single word.
- Cascaded menus are often used for selecting a value for an item (usually a set of items with Option buttons): e.g.

  Switch View: <list of views> or Alignment: <Left, Right, etc>.

- Use cascades for options that belong in a group and that share terms, e.g. Memory in, Memory clear etc. etc. This will help to make menu tiles narrower.

Command standards

Section Contents

- **"More" menus in programs should contain...**
- **Use "&" instead of "and" in menu commands**
- **Gray out commands that you can't use**
- **When to use lines to separate commands, and when not to...**
- **Infrared should be a cascaded command on Tools**
- **Place frequently used commands at the top or bottom of a menu**
- **Actions (Zoom etc.) with a few settings should cycle not stop**
- **For commands that cycle between settings...**
- **Apps should have separate Import and Export**
- **All apps should have a Switch view command using Ctrl-Q**
- **Using ellipses for commands that display a dialog**
- **No more than 8 items on a menu, unless you really have to**
- **The difference between Settings, Options and Preferences**
- **Capitalisation in command names, dialogs etc.**
- **Giving clues about dialogs and cascaded menus**
- **Graying out menu commands**
- **In general, don't change the names of menu commands when view changes**

"More" menus in programs should contain...

The More command on the File menu in programs can contain any of the following commands (in this order):

Save as...

Save

Revert to saved

Merge in...

Inport text file...

Export/Export as text file...

Tidy/archive file...

Use "&" instead of "and" in menu commands

E.g. "Record & replace" in Record, instead of "Record and replace".

Looks neater and makes the text string shorter.

Gray out commands that you can't use

In the same way that dialog lines that are currently unavailable are grayed out, menu commands should also be grayed out.

Makes it obvious to the user that they are unavailable and is much more satisfactory than providing a host of messages that say "Cannot do this..." etc. etc.

When to use lines to separate commands, and when not to...

Use them to group commands that have some relationship to each other, and separate them from commands that are unrelated, but that appear on the same menu, e.g.:

Create new file

Open file

Close

Do not use them around single commands, unless the command is at the top or bottom of the menu. A menu tile will quickly seem line-bound, if you have lines around single commands in the middle of the menu. E.g. don't have:

Create new file

Open file

Printing

More

Close

If you find this happens you should try rearranging the menu tiles. E.g. the above could be:

Create new file

Open file

Printing

More

Close

(After all, you are printing files and you have More file commands, so these all really belong together.)

Do not use lines between commands on cascades. This makes the cascaded command appear even more complex.

Move commands to other menus, create more cascades — you might even try changing the wording of the commands so that they fit more happily within a group.

Infrared should be a cascaded command on Tools

As:

Infrared >

> Send

> Receive

As stated in **Standard menu items and their shortcuts**.

Reasons: Not used very often. Also makes the last menu tile narrower — good plan because this helps to prevent other cascades having to appear to the left of the menu tile.

Place frequently used commands at the top or bottom of a menu

So that users notice them first.

Although the first command on a menu has prominence, don't forget that users will also notice the command at the bottom of a menu just as much.

Actions (Zoom etc.) with a few settings should cycle not stop

All commands that move up and down a small number of settings — like Zoom in and Zoom out should cycle round the various settings, rather than coming to the end of the "list" and forcing the user to use the other command to go back.

It's annoying to the user who is trying to find the best setting to not be able to move between them all without changing keypress.

Users should not be confused by this because it mirrrors the action of the arrow keys when moving between menus, and between commands on a menu. The right arrow key, for example, takes you all the way to the right until it gets to the end, when it takes you to the far left.

For commands that cycle between settings...

...and that have no obvious effect on screen, display a message in the bottom left corner of the screen to let the user know that the action has taken place.

Apps should have separate Import and Export

Since the Save as and Merge in dialogs need to be kept as simple as possible, all the import/export stuff should be put in separate dialogs (rather than adding a File type line to the Save as and Merge in dialogs).

So the appropriate apps will need Import file and Export file commands. The commands should be at the end of the More choice list on the File menu.

All apps should have a Switch view command using Ctrl-Q

See **Standard menu items and their shortcuts** for where this command should appear.

"Q" was chosen as the shortcut key because of its proximity to Ctrl.

Preferences should use Ctrl-K.

The Switch view command will cycle you round your preferred views in the same way that the diamond key does on the Series 3 machines.

Switch view will also cascade, giving the list of views that you can move to:

Switch view >

>> Day view

>> Week view

etc.

etc.

Note — "Switch view" cascades from the menu when you select the menu command and cycles through the views when you press its shortcut.

Using ellipses for commands that display a dialog

All menu commands that display a dialog should have an ellipsis at the end, e.g. "Page setup...". This lets users know which commands perform an action as soon as they are selected and which ones don't.

To keep the menu command as short as possible, there shouldn't be a space between the last letter of the command name and the ellipsis character.

Note that commands that display a dialog to give information about the progress of something that has started as a result of selecting a menu command, e.g. Infrared Send and Receive, should not have ellipses. The action has already started before the dialog is displayed.

No more than 8 items on a menu, unless you really have to

If you find you have more than 8 items on a menu, try reorganising them, or move infrequently used options onto a cascade. If you really have to squeeze 9 options on a menu, you can, but you'll have to squish things up a bit and not use many dividing lines between options.

The difference between Settings, Options and Preferences

This is a bit of a murky area, so to clarify it, in general we should use the terms as follows:

Options

Try to avoid using this one if you can. If you have to use it, it should be the menu command that gives access to commands that are alternatives to each other. Usually, however, you'll be able to find an alternative wording, e.g. we originally had Options leading to the different types of drawing tools on the Tools menu in Sketch, however we have now renamed this "Drawing tools". Where you want to give access to additional commands on a menu that won’t fit on the top level, you should use the command "More" instead of "Options".

Settings

As a menu command, it’s used for setting the attributes of something, e.g. a communications program might have modem settings.

Preferences

Use for menu commands that allow the user to "customise" the application, ie change the way things are presented on screen, change the features that are available, change the default settings for things etc.

Capitalisation in command names, dialogs etc.

We should use capital letters on the first word in a menu command only, i.e. we should have:

Create new file

and not:

Create New File or Create new File or Create New file

We should capitalise only the first word in dialog line text, similarly.

THE EXCEPTION IS VIEW NAMES. These should always be capitalised, wherever they appear.

Giving clues about dialogs and cascaded menus

Commands which lead to dialogs should be shown ending in the "..." character.

Commands which have a cascaded menu should be shown ending in the right-pointing triangle character.

(There's no need to use these characters if/when referring to them elsewhere - e.g. in Help, InfoPrints etc.)

Graying out menu commands

'Gray-out' any commands whenever they are not appropriate. If the user selects one, an InfoPrint should explain why it's unavailable.

Note: where a menu command is a cascade, all its options should appear grayed when unavailable. Do not hide the commands under any circumstances, otherwise the user may think that something has gone wrong.

In general, don't change the names of menu commands when view changes

The menu contents should generally be fixed. For example, the commands should not change around when e.g. shifting to a new view. (If it's an entirely different view, requiring entirely new menus, then sure — like in Sheet — because the user's asked for a completely different "mode". But avoid subtle changes, as they only confuse.)

Some entries change their names slightly to reflect the state of the program. For example, "Undo" might become "Undo Deletion".

Avoid flip-flop command names - like "Wrap on" becoming "Wrap off" — for the obvious reason that users often aren't sure whether it's saying what the setting is, or what it will be if the command is used. Use tickboxes on menu commands instead. This example should just always be e.g. "Wrap to screen", and have a tickbox.

Shortcut keys

Section Contents

- **Standard menu items and their shortcuts**
- **Don't display the shortcut in a menu tile when..**
- **Don't include shortcuts for commands that cascade from buttons**
- **Choosing shortcuts — use the standard ones for standard commands, then...**
- **All commands should have shortcuts, if possible**
- **No numbers to be used in menu shortcut keys**

Standard menu items and their shortcuts

Here are the standard menu tiles and their contents:

File	Edit	View	Insert	Format	Tools
Create new file Ctrl-N Open file Ctrl-O	Undo Ctrl-Z Redo Ctrl-Y	Zoom in Ctrl-M Zoom out Sh-Ctrl-M	Sketch Graph Other object Sh-Ctrl-O	Bold Ctrl-B Italic Ctrl-I Underline Ctrl-U Paragraph > >Indents >Tab positions >Horizontal alignment >Vertical spacing	View preferences Ctrl-K General preferences Sh-Ctrl-K
Password Sh-Ctrl-Q	Cut Ctrl-X Copy Ctrl-C Paste Ctrl-V Delete/Delete all Ctrl-D Select all Ctrl-A	Show toolbar Ctrl-T Show top toolbar Sh-Ctrl-T Wrap to screen Ctrl-W			Spell check Ctrl-L Thesaurus Sh-Ctrl-L Word count Sh-Ctrl-W

Printing > >Page setup Sh-Ctrl-U >Print setup Sh-Ctrl-P >Print preview Sh-Ctrl-V >Print Ctrl-P	Find Ctrl-F > > Find next Ctrl-J > Replace Ctrl-H > Go to Ctrl-G	Switch view Ctrl-Q > > List of views > Name view			Infrared > > Send > Receive
(major app-specific)	(major app-specific)	(major app-specific)		(major app-specific)	* Help on program Sh-Ctrl-H * About program Sh-Ctrl-A
More> >Save as Sh-Ctrl-S >Save Ctrl-S >Revert to saved Ctrl-R >Merge in Sh-Ctrl-I > Import > Export >(minor app-specific)	Object> > Edit object... Sh-Ctrl-Z > Format object... Sh-Ctrl-J More> > (minor app-specific)	More> > (minor app-specific)		More> > (minor app-specific)	(major app-specific)
Close Ctrl-E					More> >(minor app-specific)

* = 3rd party applications only

Note: If your application has an Insert menu, it should be positioned between the View and Format menu tiles.

You should not have more than 8 commands on a menu — use cascaded commands to avoid this. This menu tile layout shows 9 on the Edit menu for the purposes of showing which commands should be included on this menu. In practice none of the apps use all the Edit commands that are listed here.

Below is a list of all the standard system shortcuts. You should avoid all of these shortcuts for your application specific functions and use the remaining unclaimed "shifted" shortcuts instead. Bear in mind that you need only supply shortcuts for the most commonly used functions — keyboard users always have the option of doing the three-key "<menu>-<tile>-<command>" keystroke to invoke a menu command — so it's not a big deal not to have a shortcut for everything.

The platform-wide "standards" (in English) will be:

Ctrl+	Normal keystroke	Shifted keystroke
A	Select all	* About Program name
B	Bold	
C	Copy	Insert special character
D	Delete	
E	Close	
F	Find	Font
G	Go to	
H	Replace	* Help on Program name
I	Italic	Merge in
J	Find next	Format object
K	Preferences	
L	Spell	Thesaurus
M	Zoom in	Zoom out
N	Create new	
O	Open	Insert object
P	Print	Print setup
Q	Switch view	Password
R	Revert to saved	
S	Save	Save as
T	Show toolbar	Show top toolbar
U	Underline	Page setup
V	Paste	Print preview
W	Wrap	
X	Cut	
Y	Redo	
Z	Undo	Edit object

* applies to 3rd party applications only.

Don't display the shortcut in a menu tile when..

.. you can't activate the command while the dialog is displayed because users are likely to try out the keypress straight away, and it won't appear to work for them.

In some special circumstances, you may decide you want to in order to advertise functionality that will be often used, eg Insert pagebreak keypress Ctrl-Enter in Word does not work while the menu bar is on-screen. However we decided to display the shortcut next to the command in the menu because inserting pagebreaks is an often-used command and it's useful to know the shortcut key for it.

Don't include shortcuts for commands that cascade from buttons

For example, don't include shortcuts for the commands that cascade from a View button in the Toolbar (See Agenda for an example), or that cascade from the Command icons to the left of the screen (Infrared Send and Receive etc.).

These commands are all pen activated when they are selected from the button, so they do not need keyboard shortcuts. They have shortcuts for the equivalent commands available from the menu bar.

Including shortcuts would also make the lists of commands too wide.

Choosing shortcuts — use the standard ones for standard commands, then...

...allocate shortcuts so that you:

- give them to to the commonly used commands first. When all those have shortcuts, you can then consider the less often used commands.
- choose shortcuts that are easy to remember — using a letter from the command name is a often a good plan, or a letter that "sounds like" the command.
- are careful to account for users forgetting whether or not they are supposed to use shift with the shortcut. E.g. don't have Undo & Redo using the same key with one shifted and one not. Make sure that the exact opposite of the user's intention is not carried out if they press the wrong keypress.

It doesn't matter if you use a standard shortcut for a non-standard command in your program, as long as:

- you do not have the standard command
- no adverse actions are carried out if the user presses the shortcut thinking that they will get the standard command

---

All commands should have shortcuts, if possible

If not, make sure you allocate them to the most frequently used, then the most inaccessible commands (those on cascades) first.

No numbers to be used in menu shortcut keys

Help

Help for 3rd party programs will be available from the Tools menu

3rd party apps will not be able to capture the Help keypress. The Help keypress should always display the default machine help. They should supply a menu command called "Help on program name", and a further command to display the "About program name" dialog.

Toolbars

Chapter Contents

- Top toolbar
  - Order of buttons on top toolbar
  - Spacing of buttons on top toolbar must be regular
  - For cosmetic reasons, limit the number of sizes of top toolbar buttons
  - Buttons on the top toolbar shouldn't display dialogs
- Toolbars
  - Standard positions for commands on toolbars
  - Be sure to include a command for not showing toolbars
  - Include buttons that move between views if you can
  - Using different buttons in different views
  - Buttons on Toolbars should not have >
  - Buttons on Toolbars should not have ellipses (...)
  - Choice of toolbar buttons
  - Ideally just one button should latch down on the toolbar
  - Toolbars: general comments

Top toolbar

Section Contents

- Order of buttons on top toolbar
- Spacing of buttons on top toolbar must be regular
- For cosmetic reasons, limit the number of sizes of top toolbar buttons
- Buttons on the top toolbar shouldn't display dialogs

Order of buttons on top toolbar

Try to group related buttons together and separate the groups with spaces. E.g. in Word:

Style Font Font size | B I U | Alignment Borders Bullets

Buttons for changing font, then buttons for emphases, then buttons for formatting entire paragraphs.

Similarly in Sheet:

Font Font size | B I U | Alignment Borders Shading | Freeze Panes Sum Functions

Buttons for changing font, then buttons for emphases, then buttons for formatting highlighted blocks of cells, then special features.

Spacing of buttons on top toolbar must be regular

For cosmetic reasons, you should ideally make the spacing of buttons on top toolbar regular. You shouldn't have:

- all the buttons squished up to one end, leaving the other end empty
- buttons at either end of the top toolbar with a big gap in the middle
- irregular spacing between buttons

The best scheme we've come up with is to group buttons where possible (e.g. keep the font name and font size buttons together, the B I U ones together), then spread the groups of buttons out along the length of the top toolbar so that the gaps between the groups are the same.

For cosmetic reasons, limit the number of sizes of top toolbar buttons

Allowing many different sizes of buttons in the top toolbar makes them look untidy and "unprofessional". If it were possible, we would have liked to have just one size of button in the top toolbar, for neatness sake. However there are different sized icons and text to place on the buttons, and some buttons need a little down arrow to undicate that they drop down a list, so we've settled for providing 4 different sizes of button, and recommending that each program uses just 3 of the 4 sizes available as follows:

- The smallest size is a fixed size for buttons that either have small icons (e.g. bullet list icon in Word), or single letters (e.g. Bold, Italic and Underline B I and U). These are 35 pixels wide.
- The smaller of the middle sizes is for buttons that have an icon (e.g. paragraph alignment in Word) — 51 pixels.
- The larger of the middle sizes is a fixed size and will be used for point size buttons — 68 pixels wide. Note: we would have liked to make this button the same size as the alignment button, but couldn't because pt is clipped with 3 digit font sizes.
- The largest will be for buttons that have text and a drop down list (e.g. style and font buttons in Word) — 115 pixels.

Some programs may need to have "custom" buttons. E.g. Sheet Graph view has an 83 pixel sized button for Label type, Legends etc.

Buttons on the top toolbar shouldn't display dialogs

Buttons on the top toolbar should be reserved for actions that take immediate effect and should not be used for actions that require a dialog to be displayed first.

Toolbars

Section Contents

- **Standard positions for commands on toolbars**
- **Be sure to include a command for not showing toolbars**
- **Include buttons that move between views if you can**
- **Using different buttons in different views**
- **Buttons on Toolbars should not have >**
- **Buttons on Toolbars should not have ellipses (...)**
- **Choice of toolbar buttons**
- **Ideally just one button should latch down on the toolbar**
- **Toolbars: general comments**

Standard positions for commands on toolbars

There are standard positions for toolbar items which you should follow if you can - Spell 1, Sketch 2, Print 4, Find 2/3, New 3, Go to 4. Not a big deal — just a guide to use if you can.

Be sure to include a command for not showing toolbars

You should provide a toolbar and let it be turned off (some people dont like having a clock pointing at them).

Include buttons that move between views if you can

It is important to include buttons for moving between views in the Toolbar because it advertises the fact that more views are available.

It may be appropriate to have a button for each view you can move to, e.g. Sheet, or one button that gives access to commands for each of the views, e.g. Agenda.

It depends how many views there are to move to (more than 4 and you have to have one button for moving between views), and how many buttons you want to include for oft-used commands.

Using different buttons in different views

With a program that has different views on the same information, and enough oft-used commands that are common to all views, e.g. Agenda, you should try to use the same buttons on the Toolbar in each view. This allows the user to perform the same actions by the same means in all the views (something that they will expect), and reduces the amount of things that change on screen when they switch view (cosmetically pleasing and results in less confusion..."I'm sure there was a button somewhere that let me do that, but I can't remember which screen it was on..." etc.).

With a program that has different information in each view, but that has some common and oft-used commands in all the views, e.g. Sheet, you should include as many of the common commands as buttons on the Toolbar as possible. Where there are not enough common commands, or where there are commands that are more often used than the ones that are common, you should use buttons that are specific to the individual views. Note that the buttons that are common to all views should appear in the same place on the toolbar in each view.

With a program that displays different information in each view and no commands that are common to all views, obviously there should be different buttons in each view.

Buttons on Toolbars should not have >

To show that there is a list of options. Reasons being:

- To leave room for as much text as possible.
- These buttons are not akin to menu commands and dialogs. The fact that they have text is for explanatory purposes; ideally, it would be better to just use graphics and then ellipses would not be necessary or appropriate.
- They are not cosmetically pleasing. On the top toolbar and menu commands, the arrows point to where the list will appear. Here

they would point off the screen or into the next button.

Buttons on Toolbars should not have ellipses (...)

Don't use ellipses on buttons on the Toolbars to show that they display a dialog. Reasons being:

- To leave room for as much text as possible.
- These buttons are not akin to menu commands and dialogs. The fact that they have text is for explanatory purposes; ideally, it would be better to just use graphics and then ellipses would not be necessary or appropriate.
- They are not cosmetically pleasing.

Choice of toolbar buttons

The principles behind the choice of buttons on the toolbar are to provide buttons for those actions that a user may want to do frequently, and to advertise functionality that may not be easy to spot.

For those programs that have different views:

- if they have 2 views there are two buttons for switching between them — this shows that there is more than one view available
- if they have more than two views, there is one button that cascades — ditto comment above
- Programs that have the same buttons ought to have them in the same position, e.g. Print appears at the bottom of the toolbar.

Ideally just one button should latch down on the toolbar

People will use these buttons to find out at a glance which view they are on, so you should avoid having buttons that can be latched down at the same time on the toolbar. E.g. In Sheet, the Sheet view started off having the Titles button on the Toolbar. Problem with this was that with the Titles on, you might wonder "which view am I on — the Sheet view or the Titles view?".

Toolbars: general comments

"Toolbars" consist of a group of buttons and a clock.

Relationship between toolbar actions and menu commands

The keyboard user must be able to duplicate each toolbar action by selecting a menu command (which has a shortcut key). (Because it's important that there doesn't appear to be 2 sets of functionality in the interface — menus and toolbars.) In a few cases, though, the keyboard user may then require an extra keystroke or two. Examples of the unusual cases:

- If you had a "Print now" toolbar button, a keyboard user could achieve this by selecting the normal Print command, followed by an Enter. That's OK — no separate "Print now" menu command is required. (People would wonder what we were thinking about if our menu contained near-identical "Print" and "Print now" commands.)
- In some apps e.g. Agenda a "View" toolbar button pops up a list of views — the menu should include a "Switch view" command that pops up a list of the views with accelerators for each view.

Text, icons, or both?

By default, place text to the right of an icon.

A few apps, such as Sketch, work best with icons but no text. But it's very rare to find a situation which works best with text but no icons. (Icons are in general more powerful reminders than text. Humans have great visual abilities — pattern recognition. To be really effective, though, icons have to represent things or ideas from the user's world, of course — and, ideally, be guessable.)

Actions of toolbar items

When a button is "unavailable" it will still depress as normal when pressed, then (typically) the app will display an Infoprint explaining why it is unavailable.

Latching icons are possible, to reflect the current state of something.

What sort of functions should go in the toolbar?

The following critera should be used when choosing default toolbar buttons:

Criteria	Examples
1. Advertise the major functionality of the program	Find, Add, Change view, Spell, Print
2. Used frequently	E.g. "Today" in the Agenda
3. Are "valid" for a high proportion of the time	Most of the above
4. For "instant" operations.	e.g. Changing the view — things which don't need to go via dialogs and/or further keyboard actions
5. Reflect PC apps' buttons — which actions they are, in which order, and with what sort of icons	Both W4W and Excel have this order: New, Open, Save Print, Print Preview, Spell Cut, Copy, Paste, Format painter Undo, Redo/Repeat Bold, Italic, Underline Right, Cente, Left, Justified TipWizard, Help Any similarities between your toolbar and that of another app, should be entirely coincidental, of course.
6. Related to "pen-activity"	Navigation.

Things to avoid

Toolbars are to provide non-novices with the sort of actions mentioned above. Don't choose the 5 most basic operations if intermediate users aren't going to want them any more.

Avoid multiple button bars or a "More..." button. This defeats the purpose of the tool-bar: fast, instant, etc.

As far as possible, don't change the tool-bar buttons, e.g. when the user has switched to a different view.

If you've only a few things you want on a toolbar, don't fill it out with useless buttons — leave the buttons big, as it helps finger-use. Support finger-use wherever you can, but not everywhere — if you need small controls, use them.

Most programs should use one column only. Only use two columns if a large number of buttons really is necessary or useful (e.g. in Sketch) — ie worth losing the screenspace for.

Example

Agenda (in all views) is like this:

Button	Explanation
View	Tabs out to a list of all views — shows the major functionality
Sketch	The main purpose of a Sketch is to enter data quickly, and it's a pen activity.
Today	Most frequent operation
Goto	Calendar navigation — pen activity

Basic UI principles

Chapter Contents

- Using Arrow keys for navigation
- Filenames are case sensitive, and...
- Using filename extensions
- Clicks for selecting items and commencing actions
- Sequence of items in resource files
- Be consistent in use of capitalisation in UI
- Double-clicks on screen are provided in Eikon, but we don't use them anywhere
- Use Infoprints and Beeps when it isn't obvious what's happened
- Inserting/saving files when in an embedded object: use "Import file"/"Export to file"
- Dim Command icon commands and display an Infoprint when disabled
- Last file used — record this, not "last file opened", on closing the program
- Where to use dogears to indicate how to navigate
- Don't use "Please wait" messages in your app - be specific with infoprints
- Error messages (and status messages)
- Reducing the average amount of user confusion is a vital consideration
- Try to *misinterpret* your words
- Ideally, dialogs would only appear when the user asks for them
- Making apps intuitive takes a long time and many iterations
- You only succeed when all main features are good enough
- The ultimate goal: everything the user can do should be intuitive

Using Arrow keys for navigation

We've changed arrow keys slightly from 3a days.

Rather than "Ctrl moves all arrow keys by more, and Fn by even more, and Ctrl+Fn by the max", we change this for Fn+left/right (Home/END) where there is a Home/End action you might expect from PC usage.

So on the day view of the Agenda, Home/End go to the top/bottom of the view (instead of the 3a's move-by-week), while Ctrl+Fn+left/right still move by a bigger amount (which in this case is move-by-week).

Filenames are case sensitive, and...

Our filenames are case-sensitive and we don't auto-capitalise the user's filenames (though the pre-defined files we create have capitalised names).

If the user types "xyz" she gets it in lower case.

HOWEVER we don't let the user create two files whose names, case-independent, are the same (e.g "FREDA" and "freda"). This happens automatically.

Using filename extensions

We don't use file extensions except where required to differentiate files of the same name - e.g. an OPL program called FRED translates to FRED.OPO.

Clicks for selecting items and commencing actions

We use single-taps to select and two-taps (not just "double taps") to action a command, whenever you can select but not start an action. E.g. when selecting files in the System screen.

Sequence of items in resource files

For ease of localisation, all programs that have a single resource file should have items in the following order:

- Shortcut keys
- Toolbar
- Menu commands
- Dialog titles
- Dialog 1
- ...
- Dialog N
- Dialog 1 pages
- Dialog 1 page i
- Dialog 1 page i choice list a
- Dialog 1 page i choice list b
- Dialog 1 page ii
- Dialog 1 buttons
- ...
- Dialog N pages
- Dialog N page i
- Dialog N page i choice list a
- Dialog N page i choice list b
- Dialog N page ii
- Dialog N buttons
- Dialog text, etc.
- InfoMsgs
- Debugging messages and other text that is to be removed for the candidate release

---

Be consistent in use of capitalisation in UI

We have chosen to use a capital letter on the first word of resource strings only, except where a word is a proper noun. E.g. we will have:

"Day entry preferences" as a command, not "Day Entry Preferences", or "Day entry Preferences".

Inconsistent use of capitals makes the UI look messy and unpolished.

Double-clicks on screen are provided in Eikon, but we don't use them anywhere

We do provide for the use of double-clicks in programs, but have chosen not to utilise them in the built-in programs because it is an awkward action for users to achieve on the screen.

3rd party developers can use this feature if they wish, although if they want their programs to look like ours, they will choose not to.

Use Infoprints and Beeps when it isn't obvious what's happened

Users need feedback to know that the command they've selected has been carried out.

In most situations something will happen on the screen to mark the action, e.g a dialog will appear, or the text they've typed will change in some way. Other commands, however, cause no change to the contents of the screen; for these you need to provide an Infoprint to show the action has been carried out — Copied, Saved etc. — or sound a Beep when an Infoprint isn't appropriate.

Inserting/saving files when in an embedded object: use "Import file"/"Export to file"

If a program allows the user to insert a separate file within an embedded object, or save that object as a file — e.g. Word and Sheet may want their objects to allow this — we should do it with an Import file / Export to file command rather than an Open file / Insert file / Save as command.

Reason being, we want to retain the distinction between files and embedded objects, and try to ensure that the user does not consider that an embedded object is like a normal file.

Dim Command icon commands and display an Infoprint when disabled

The commands (for example Infrared Send and Infrared Receive) should be displayed dimmed, otherwise users may think the Command icons to the left of the screen have stopped working.

There should be an infoprint (otherwise users think they missed, or something).

The message should be:

"This item is not available"

Last file used — record this, not "last file opened", on closing the program

All programs that can automatically load the last used file on opening should record the file that was in use when the program was closed as the "last used" file. They should *not* record the file that was most recently opened as the last used file. This is the most intuitive behavior for the user. It means that the file she was last viewing is the one that is loaded when the program is next opened.

You'll see there's a difference if you imagine more than one copy of your program running: when they have all closed, the file last opened may not be the same as the last one closed.

Where to use dogears to indicate how to navigate

You should use dogears to indicate the area of the screen to tap to turn pages rather than single arrows when the screen display is a representation of a physical page e.g. Agenda.

Don't use "Please wait" messages in your app - be specific with infoprints

Instead of displaying "Please wait" (which gives the user no clue as to what she is waiting for), display a specific message. e.g. while a merge in is taking place, display "Merging in...".

And put "..." at the end — it implies that you have to wait without saying so specifically.

Error messages (and status messages)

Attention to detail in the UI hasn't traditionally been important in software development, whereas now it can make or break your product. Perhaps the area of UI which has had least attention paid to it is error messages. They've generally just been whatever programmers fancied saying, whenever they fancied saying it. So a little attention to detail here can make a big difference.

A basic principle:

Only display error messages when you have to, and word them carefully. Sometimes error messages just look like stupidity on the part of the program; most, even though they seem innocuous to the programmer, make any normal user think they're being told *they* are stupid. Alan Cooper says "The program is doo-doo, pond scum. It is less than zero. To be told by software that you have failed is humiliating and degrading. Users, quite justifiably, hate to be humiliated and degraded." Most people would rather that if they did make a mistake, it got lost in the noise, or ignored, or taken care of. They don't want a computer telling them all day.

There follow some more specific examples of this principle — most important first.

Avoid jargon — words that users may not know — and *avoid pidgin-speak* (typically missing out verbs) which means that users who don't know which words are jargon can't work out the syntax of the "sentence". *In almost all current software, "error messages" are full of these!* Re-read your "error messages" as if you were a novice with no knowledge of terminology etc, and try hard to misunderstand it any way you can. Ask yourself — is it jargon-free? If you have to use jargon, can you pick it out with e.g quote marks? Could it possibly be read the wrong way at all, if I pretend I don't know which words are jargon and which are not? Could any one noun be read as a verb, or vice versa?

- Cursor not on an entry

  (is "cursor" an 'imperative', ie "don't cursor on an entry!!!"?) "Cursor is not on an entry" is clearer.

- No system memory

  (Hmm, they said there was in the shop.) "Memory is full"?

- Read only file

  (Sounds like another Japanese-style demand. Was I reading something else?) "File has been marked 'read-only'", perhaps.

- File or device in use

  (Is "file" a verb here? Then what's it mean? Or is it saying I should select a "file" or a "device in use" and I selected something else?) Try "File (or device) is in use".

- This repeat occurrence disabled

  (What? Nurse!) No message at all, or "This is now a separate entry" would be a bit better.

- No records

  (Does it really? What does Yes do?) "There are no entries".

Check you're not being unnecessarily scary.

- "File does not exist" should be "File not found" or similar. The user may have mis-typed the filename (or be in the wrong folder) but not realise, and so think you're saying that the file has vanished.
- "There is an unrecoverable error on file XXX.DOC". To the user, "unrecoverable" means "you cannot now recover your file"! Should be something like "Problem while trying to save file. Check your file is OK".

Don't display messages you don't need. They stop the proceedings and interrupt the user's interaction.

A hammer has a good UI. Use it well, all is well, but use it badly, you don't get such a good end result. It does not stop and say: "You Bent The Nail Five Degrees (You Idiot)! [Abort]".

If it's serious then say so. On the 3a the "Battery too low to write to Flash" message sounds harmless, but ignoring the problem

means the possibility of corrupting your disk.

If you can, give likely explanations and/or courses of action. Ideally if printing fails don't say "Invalid printer" or some such, say what it might be (the printer's off-line?) and e.g have buttons for actions like "print it when possible" (ie spool it) etc. Try not just to say "wrong!". "Matching requires sorted file" — well why not tell them how to sort the file then? "Use the 'Sorting' option before matching"?

Always tell the user if you have to go deaf for a bit. At the same time, tell them it's OK for this to happen (ie you're not issuing a warning), and when you do go deaf, tell them how long left if possible, and let them cancel it if at all possible. If you're, say, copying multiple files, what use is a progress meter for each file??? Almost none at all. If you do want a progress meter, look at the files, add up their sizes, and you can then make a pretty good guess at the progress, as you copy.

Only validate data that you have to validate. Error messages have in the past been used as much for the programmers' own convenience as anything else. It's easy to make the user fill in a dialog with "valid" data (ie the few things which your particular program was equipped to handle), and to say "Invalid Data" if it wasn't "valid". It's harder to handle a variety of possible input yourself.

If it's the program's limitation, say so. Don't make it sound like it's the user's own stupid fault for not knowing. Don't say "Invalid data" if the data may look perfectly valid to the user.

Reducing the average amount of user confusion is a vital consideration

..and it outranks most other considerations.

For example, an early "comms setup" dialog which, after lots of lines to do with various settings, ended with this line:

"Save current settings as default? [Y/N]"

There were two ways to cause this dialog to be displayed:

1. By tabbing out a line from the "Connect" dialog. In this case, the "Save current settings as default" line was set to "No" by default — the thinking being that if you wanted to change the settings when about to connect, it would most likely be because of some temporary circumstance.
2. By choosing a "comms setup" menu option. In this case, the line was *fixed* to "Yes" (the entire point of a setup dialog, after all, being to save the settings as the "default" for all future connections).

There are problems with both the above:

In case 1), the dialog might be better with "Yes" (or "All future connections" or whatever) as its default. The user may not have seen the standalone "setup" dialog and may just be trying to get on with a connection. Or the user may be making a permanent change to the settings, and be doing it from the "Connect" dialog. Also, it avoids the user going in, checking the settings are correct, but then being scared to hit Enter because that last line is saying "these settings won't be saved as default"... err will they be saved at all? What as? What about the next time — will they have gone back to some other values?

(People often do hit Enter in spite of having made no changes to a dialog, by the way. They look at the dialog, think "yes that's what I want", and press Enter to say so).

But even if you think that users in this case would more often want "Just this connection" than "All future connections", reducing the average number of keypresses required is not as important as reducing the average amount of user confusion.

In case 2), it would have been better to make the effort to remove this line from the dialog. Permanently disabled dialog lines are almost always for the convenience of the programmer, and usually at the user's expense: avoid them if you can. Users may think some or all of the following: "Why is it saying 'This item cannot be changed'? Which line of the dialog do I have to change in order to 'enable' this line? What's this 'not saving as default' feature which I can't currently do? I'm sure I was able to do that last time in this dialog [e.g user last got here through route (1), not route (2)]- what I have done to disable this line? Have I done something wrong since last time?"

If the line wasn't there, the only potential for confusion is a few users who notice that a line which they saw before is missing, and wondering where it's gone. You hope most of them would remember that that line was to do with the fact that they were making a connection at the time. Not perfect, but less confusion overall. The vast majority now just see a "setup" dialog which makes perfect sense.

(The above are more examples of "putting yourself in the user's shoes", of course.)

Try to *misinterpret* your words

Take time with wording — error messages, dialog titles, prompts and so on. Before you accept each word or phrase, "put your end user shoes on", read it as an end user and actively try to find a way to misunderstand it, or not understand it, or misread it, or be scared by it. If you succeed, try to re-word it.

Often each phrase takes a few goes before you're happy. But it scores well on a cost:benefit ratio — a few seconds work (even a minute or two, maybe) to reduce the chance of a misunderstanding. You can never be perfect for everyone, and there will always be the odd misunderstanding, but few users give up when they meet the first one. The more unintelligible or misinterpretable things in an interface, the sooner people give up on it.

For example:

The "comms setup" dialog (already mentioned in the section immediately preceding this one), which, after lots of lines to do with various settings, ended with the line:

"Save current settings as default? [Y/N]"

At first glance, that's fine. But with your "idiot user" shoes on, you can mis-read it. It uses the phrase "current settings". The whole reason you're in this dialog in the first place is because your *current* settings (ie the pre-dialog ones you're trying to change) are wrong. I'd hardly want *them* as my default, overriding the ones I'm entering here?! So why's it saying such a thing? It might be better as e.g:

"Save these as default settings? [Y/N]"

But, then again, what's a "default"? It's largely industry jargon - lots of people won't understand it. (If your mum says "Actually I'll have *two* sugars today please dear" do you ask her "is that the default"?) So maybe something like:

"Use settings for: [Just this connection] [All future connections]"

(Can I mis-read or misunderstand this? Can I improve on it? Probably, given a few minutes! But it's certainly better.)

There are lots and lots of examples (many far, far worse) all over the place in software. They're the main reason people don't understand computers — because *we* don't take the time to communicate with people with unambiguous, intelligible words. (It's *our* fault, and no-one else's.) The only way round it is to learn to put "idiot user" shoes on and look harshly at your own program, as in the above example. Think for a few seconds about whether each phrase/setting is intelligible and unambiguous, and take a minute or two to fix it, if it isn't.

Ideally, dialogs would only appear when the user asks for them

Traditionally dialogs have been a tool for the programmer to throw up all the settings he/she knows about, at a point convenient to them, as if to say "OK, *I* understand these settings. You can try and set any of these things. (If you know what they mean, that is.)"

This is an appalling thing to do in a UI.

In practice the user doesn't usually *want* to see a "dialog" — they just want to tell the program to get on and do a nice simple thing!

Two things to bear in mind:

1. Occasionally you can just do what the user's going to want, 95% of the time, and if it turns out not to be quite right have some other setting/command for the user to change the action;
2. When you do need to show a dialog (and, indeed, it's only rarely in practice that you can obviate the need for one), multi-page dialogs often let you make dialogs much simpler to use. See **Multi-page dialogs can make your app much easier to use**.

Making apps intuitive takes a long time and many iterations

Seemingly simple questions or designs often involve complex discussions in order to come to the best decisions.

You have to think carefully about the variety of things normal users will want to do, and which words and ideas they'll understand.

You have to keep asking for comment from others. No spec ever covers the hundreds of different scenarios/situations the software could be in, of different requirements which different users may have for it, of similarities/inconsistencies with other areas of the software. Be prepared for repeated tweaking as you come across more and more of these. It's not exactly pleasant, but you can't make great software without doing it.

Occasionally you come up with an approach which breaks the norm; be prepared for lots of extra resistance/debate when you have such an idea. Ideas like these:

- Making ToDos appear anywhere you want them on the Agenda's "day" screen ("What? ToDo lists are ToDo lists!")
- Allowing overlapping entries, so that Repeating entries work sensibly ("What?? How can you be in two meetings at once!!")
- Keeping MS-DOS (disk letters, colons, slashes, directories and file extensions) hidden throughout the main UI ("What??? You can't hide the entire filing system, they're bound to come up against it soon!!!")

It took large amounts of discussion just to generate these ideas in the first place, then much more afterwards before they were accepted.

You only succeed when all main features are good enough

You don't succeed because your app has whizzo features. You succeed when nothing's broken; when all the day-to-day features are powerful enough and easy enough to use for normal people.

Only *then* do you worry about the whizzo features.

Key things to consider:

- Missing functionality
- Missing connectivity
- User-friendliness
- Ability to change the way data is structured, e.g in a database
- Limitations on file sizes
- Ability to compress files which can grow very large
- WYSIWYG display in apps
- Password-protection for databases or their entries
- Adequate Help/Tips

Time is much better spent on things like the above than on fine-tuning whizzo features.

The ultimate goal: everything the user can do should be intuitive

Inside, the program works one way. The user has a set of desires, concepts, things they understand. The UI's job is to get the former across in terms as close to the latter as possible.

To do this, you must put yourself in the users' shoes, and think only of the things they want to do, and in the terms they understand. It is crucial to understand that in the use of computers we are advanced beyond the dreams of normal humanity, and we must fight this if we are going to do good product specs and UI design for such people.

Why bother with all this UI stuff? Because if you write a powerful app but no-one can work out what it does or how to use it, you have wasted all that effort.

With each new app we design, we try to get closer to the above goal. (Typically it means that more and more of each app is UI related.) The attitude to app design has changed as the industry has matured. In the not-so-distant past you would feel OK if someone asked "how do I?" as long as you could say "you use Tools|Options|Whatever". Now, if they have to ask how, you've failed. Modern PC software leads users through many processes with "wizards" - a succession of dialogs asking for all the required info *to do something the user understands*. This is the sort of benchmark at the moment. In five years, *it* will look clunky and useless.

Also, if the overall impression is one of passion for the product design it invites the user to explore for hidden functionality. You should emphasise the wonder of the product as well as providing the expected. Soon the expected will be "no big deal" or "seen it

all before".