Dialogs

What is a dialog

Dialogs are used extensively by system and application user interfaces for simple notification through to highly sophisticated data presentation and capture

What does a dialog provide

- Dialogs provide a wide variety of ways to interact with a user.
- Dialogs can be used to notify, obtain a response, present fixed information, or to allow the user to enter data.

Types of dialogs

Standard dialogs

A simple dialog can be constructed by defining its layout and then writing a dialog class to handle the data. Data from the dialog can be validated and saved. More features can be added to a dialog, such as defining a menu for it or adding a custom control. Multipage dialogs can also be created.

Forms

A form can be used if you have a collection of related data that you want the user to edit.

Notes

Notes provide a convenient way to convey information to the user. Wrapped Notes offer a very simple way of communicating with the user, providing a standard format for common types of notes, such as a confirmation note or an information note. Custom notes can also be constructed.

Queries

Queries are a specialized type of dialog to be used when you wish to ask the user a question. In the simplest case, this is a confirmation query asking the user to say yes or no to a question. However, they can be more complex; for example, list queries can be constructed for the user to select items from.

List Dialogs

Two types of list dialogs are available in S60: selection lists, which allow one item to be selected, and markable lists, which allow multiple selections.

Dialog characteristics

- All dialogs share some basic properties. They are **window-owning** controls; virtually all dialog classes are ultimately derived from **CCoeControl**.
- A dialog framework manages many aspects of their behavior, including layout, drawing and the management of the user interaction with their component controls. Typically, most dialogs of any complexity are fully defined in a resource file and, after dynamic instantiation, their construction is completed by having the dialog framework load the definition from the resource file.
- The layout and positioning of all the elements of the dialog is usually automatic—the developer can influence the process, as described later. Full dynamic construction is possible for very simple dialogs.
- Dialogs can be **modal** or **modeless**, as well as **waiting** or **nonwaiting**. They have a number of lines arranged vertically, each containing one or more controls.

- A **modal dialog** prevents you from interacting with other parts of the application's UI until you dismiss it, while a **modeless dialog** does allow you to interact with other parts of the application's UI while it is active.

- A **nonwaiting dialog** allows the application to continue processing in the background, whereas a waiting dialog prevents an application from doing any further processing until the dialog is dismissed.

- **S60** dialogs are **modal and nonwaiting by default**. Modeless dialogs are **rarely** used in S60, as any dialog shown will generally have input focus until it is dismissed. Dialogs used as the main view in a Dialog-Based Application Architecture are the main exception.