



Following are some of the tips for reducing the code size by eliminating unnecessary things from your code

Contents

- [1 Avoid Excessive TRAP Harnesses](#)
- [2 Avoid Debug Code In Release](#)
- [3 Avoid Unnecessary Exported Functions](#)
- [4 Avoid Unnecessary Virtual Functions](#)
- [5 Decomposable Functions](#)
- [6 Use Common Controls](#)

Avoid Excessive TRAP Harnesses

Trap harnesses use up **space** when they are compiled. Code that contains **many TRAP macros** (e.g., more than five in a class) is using up too much space. It is also probably **incorrectly designed**, as the TRAP harness is not intended for use **extensively** in normal code. It is there to allow advanced development of **special error handling** and **recovery** routines.

Avoid Debug Code In Release

If there is any code for **logging, debugging, or testing**, it needs to be **excluded** in release builds. The compiler directive `#ifdef _DEBUG` can be used for this purpose.

Avoid Unnecessary Exported Functions

When functions are exported using `IMPORT_C` and `EXPORT_C` from a DLL, they use up space for the **export table**. Only functions that need to be used outside of the DLL **should be exported**.

Avoid Unnecessary Virtual Functions

Unnecessary virtual functions are **bad**, for reasons similar to exports, as they create extra **vtable** functions.

Decomposable Functions

There are many places where a **number** of functions that perform very **similar tasks** are present in a class. Often this common code can be **abstracted** out into a **single function**, which is parameterized to perform the different tasks required. A common example of this kind of thing is a class that implements both **NewL** and **NewLC**. Rather than **duplicate** the code in both functions, NewL can just call NewLC, performing a CleanupStack::Pop afterwards.

Use Common Controls

If possible, **use framework controls** that are available in the system (or other shared DLLs) instead of developing new ones.