

## Cleanup\_stack\_support\_for\_non-CBase\_classes

The discussion of clean up so far has assumed objects to be cleaned up are derived from CBase, with clean up by invoking delete. Other classes need explicit cleanup support to be provided by the programmer.

The cleanup\_stack supports other types of object by two further overloaded versions of CleanupStack::PushL(). You may push:

- a **TAny\***: such objects are destroyed by invoking **User::Free()** on the pointer pushed ? note that this is less powerful: it simply frees the memory, without calling the C++ destructor
- a **TCleanupItem**: an object of this type encapsulates a pointer to the object to be pushed, and a pointer to a function that provides cleanup for that object.

Some utility functions are provided that make construction of a suitable TCleanupItem easy.

## Example: RPointerArray

**RPointerArray** can be pushed in the **CleanupStack** and safely destroyed by making a **TCleanupItem** and a static function for it. In this example the array contains **HBufC** descriptors.

```
// static cleanup function
static void PointerArrayCleanup( TAny* aArray )
{
    static_cast<RPointerArray<HBufC*>>( aArray )->ResetAndDestroy();
}

RPointerArray<HBufC> array;
...
TCleanupItem arrayCleanup( PointerArrayCleanup, &array );
CleanupStack::PushL( arrayCleanup );
...
CleanupStack::PopAndDestroy(); // arrayCleanup
```