How to copy Bitmap Handles

Using the Handle() and Duplicate() methods of the CFbsBitmap class we can easily duplicate a bitmap object. The resulting object points to the same bitmap object but can be independently deleted from the original object.

This is due to the fact that CFbsBitmap is really a handle to a Bitmap managed by font and bitmap server. The managed Bitmap object is reference counted and only deleted when reference count reaches zero. So when handle to the Bitmap is duplicated the resulting CFbsBitmap object refereces the same bitmap, only with its reference count incremented.

Usage Scenarios

1. Use these API to share CFbsBitmap instances across threads/processes as you cannot share instances of CFbsBitmap directly between threads.

2. An object can take an independent ownership a CFbsBitmap instances contained by another object without fearing deletion by the containing object.

3. You can also use this technique for making a copy of the bitmap(in the same thread) rather than copying them pixel by pixel for quick,temporary usage.

Code Snippet

The following code snippet demonstrates duplicating bitmap handles within a single thread.

```
//Duplicates a bitmap creating and returning a new CFbsBitmap object.
//Ownership of newly created object vest with the caller.

CFbsBitmap* DuplicateBitmapL(CFbsBitmap & aSourceBitmap)
{
        //Target bitmap.
        CFbsBitmap* dstbitmap = new (ELeave) CFbsBitmap;
        CleanupStack::PushL(dstbitmap);

        //Get the handle to source bitmap
        TInt srchandle=aSourceBitmap.Handle();

        //Duplicate the bitmap handle. Increases the RefCount of bitmap
        //managed but Fbs Server
        User::LeaveIfError(dstbitmap->Duplicate(srchandle));

        //It should be better to track the error returned by Duplicate()
        //to get the notification about whether the bitmap is copied successfully or
not.

        CleanupStack::Pop(dstbitmap);
        return dstbitmap;
}
```

Code Snippet

The following code snippet demonstrates duplicating CGulIcon.

```
CGulIcon* DuplicateBitmapL(CGulIcon& aIcon)
{
       CFbsBitmap* pBitmap = new (ELeave) CFbsBitmap();
       CleanupStack::PushL(pBitmap);
       CFbsBitmap* pBitMask = new (ELeave) CFbsBitmap();
```


(C) Copyright Nokia 2014. All rights reserved.

```
        CleanupStack::PushL(pBitMask);

        User::LeaveIfError(pBitmap->Duplicate(aIcon.Bitmap()->Handle()));
        User::LeaveIfError(pBitMask->Duplicate(aIcon.Mask()->Handle()));

        CGulIcon* Icon = CGulIcon::NewL(pBitmap, pBitMask);
        //Both pBitmap and pBitMask are now owned by Icon
        CleanupStack::Pop(2);
        return Icon ;
}
```

NOTE: This way we are not copying the bitmap to another, we are just creating a (duplicate) handle for the bitmap, in the font and bitmap server.

Also, deleting one bitmap (thus handle) does not affect the other one.

(C) Copyright Nokia 2014. All rights reserved.