

## Saving\_settings\_with\_Dictionary\_store

Reviewer Approved



Dictionary store nearly perfect solution for storing settings data, in general it is simpler to use than other stores, with it you can set your own UID for each stream in the store, thus you can find them easily when reading the setting back.

Following functions are illustrating how to store and load settings items from the store. With Dictionary stores there is small problem which causes the store file size to increase over time, as shown in the SaveValuesL()-function with this example this problem is fixed by deleting the store each time any changes are done.

Streams inside the Dictionary store are accessed by using Dictionary store read/write streams. For internalizing and externalizing you could use << and >> operator, as illustrated with the iName variable, for integers you should use the functions provided by read/write stream as shown with iIndex variable. There are separate functions for 8,16,32 and 64 bit wide signed/unsigned integers as well as for byte buffers.

As illustrated with the iBitmap variable you can also use InternalizeL/ExternalizeL functions provided by the class itself for reading and writing to the Dictionary store.

Note that when opening Dictionary store, you need to know the UID of the store (in example 0xDEADBEEF is used as store UID), using wrong value will cause a leave.

```
_LIT(KtxDicFileName ":\private\\<My SID>\\MyDicFile.ini" );
```

```
const TInt KIndexUID          = 0x1000;
const TInt KImageUID          = 0x2000;
const TInt KNameUID = 0x3000;
```

```
class CExampleItem : public CBase
{
public:
    CExampleItem(): iIndex(-1){};
    ~CExampleItem    () { delete iBitmap;};
public:
    TInt          ;    iIndex
    TBuf<100>     ;    iName
    CWsBitmap*   iBitmap
};
```

```
void LoadValuesL(CExampleItem& aItem)
```

```
{
    TFindFile AufCCoeEnv::Static()->FsSession();
    if(KErrNone == AufFolder.FindByDir(KtxDicFileName, KNullDesC))
    {
        CDictMngtFileStoreDictionaryFileStore::OpenLC(CCoeEnv::Static()->FsSession(), Auf
            = {0x00;FileUid
            iUid = KIndexUID;

    if(aDStore->IsPresentL(FileUid))
    {
        RDictionaryReadStream in
        OpenLC(*aDStore,FileUid);
        iIndex = in.ReadIn32();
        ::PopAndDestCObjFromStack
```

## Saving\_settings\_with\_Dictionary\_store

```

}

    iUid = KNameUID;
if (aDStore->IsPresentL(FileUid))
{
    RDictionaryReadStream in
    OpenLC(*aDStore, FileUid);
    >> aItem.iName;    in
        ::PopAndDestCpy4hppStack
}

    iUid = KImageUID;
if (aDStore->IsPresentL(FileUid))
{
    iBitmap = new (ELeave) CWsBitmap (CCoeEnv::Static()->WsSession());

    RDictionaryReadStream in
    OpenLC(*aDStore, FileUid);
    iBitmap->InternalizeL(in);
        ::PopAndDestCpy4hppStack
}

    ::PopAndDestCpy4hppStack1; // Store
}
}

void SaveValuesL (CExampleItem& aItem)
{
    TFindFile Auf (CCoeEnv::Static()->FsSession());
if (KErrNone == AufFolder.FindByDir (KtxDicFileName, KNullDesC))
{
    ::LeaveIfError (CCoeEnv::Static()->FsSession().Delete (AufFolder.File()));

    CDictionaryFileStore DictionaryFileStore::OpenLC (CCoeEnv::Static()->FsSession(), AufFolder)
    = {UID, FileUid};
    iUid = KImageUID;

    RDictionaryWriteStream out1
    AssignLC (*MyDStore, FileUid);
    WriteInt32L (aItem.iIndex);
    CommitL(); out1.
        ::PopAndDestCpy4hppStack1; // out1

    iUid = KNameUID;
    RDictionaryWriteStream out2
    AssignLC (*MyDStore, FileUid);
    << aItem.iName;
    CommitL(); out2.
        ::PopAndDestCpy4hppStack1; // out2

    iUid = KImageUID;
    RDictionaryWriteStream out3
    AssignLC (*MyDStore, FileUid);
    iBitmap->ExternalizeL (out3);
    CommitL(); out3.
        ::PopAndDestCpy4hppStack1; // out3

    ->CommitL(); // Store
        ::PopAndDestCpy4hppStack1; // Store
}
}

```

Saving\_settings\_with\_Dictionary\_store

}

## Other Related Links

[Using .ini Files](#) Insert non-formatted text here