

Contents

- [1 Purpose](#)
- [2 Example code](#)
- [3 Example application](#)
- [4 Internal Link](#)

Purpose

While encrypting/decrypting contact database the common problem is that the static function 'CContactDatabase::DeleteDefaultFileL()' always returns error code -14 (**ErrInUse**).

Encrypt/Decrypt contact database is explained here: [Encrypt-Decrypt Contact Database](#)

The procedure to be followed while encrypting and decrypting contacts database:

1. Create a database of your own (MycontactDB) and copy all contacts from the default database (DefcontactDB) to MycontactDB.
2. Encrypt your database MycontactDB.
3. Delete all contacts from DefcontactDB (as we have a backup of all contacts in encrypted form in MycontactDB) instead of deleting default contact database using CContactDatabase::DeleteDefaultFileL().
4. Decrypt MycontactDB when required.
5. Copy all contacts from the MycontactDB to DefcontactDB.

Example code

The following code creates a database MycontactDB and copies all contacts from the default database (DefcontactDB) to MycontactDB:

```
// Copy all contacts from default contacts DB to My Db (creates own DB)
void CopyFromDefault()
{
    // open default contact database
    CContactDatabase* DefcontactDB= CContactDatabase::OpenL();
    CleanupStack::PushL(DefcontactDB);
    // create own database
    CContactDatabase* MycontactDB = CContactDatabase::ReplaceL(KFileName);
    CleanupStack::PushL(MycontactDB);
    //create iterator for DefcontactDB
    TContactIter contact_iterator(*DefcontactDB);
    TContactItemId contact_id;
    //go through contacts while there any info
    while( ( contact_id = contact_iterator.NextL() ) != KNullContactId)
    {
        // Read contact from default database
        CContactItem* contact_item = DefcontactDB->ReadContactL(contact_id);
        CleanupStack::PushL(contact_item);
    }
}
```

How_to_manage_contact_database_while_encryption

```
// Add it to own database
MycontactDB->AddNewContactL(*contact_item);
CleanupStack::PopAndDestroy(contact_item);
}
CleanupStack::PopAndDestroy(2);
}
```

The following code encrypt your database MycontactDB:

```
// Encrypt contacts in my database
void EncryptAll()
{
    CContactDatabase *MycontactDB = CContactDatabase::OpenL(KFileName);
    CleanupStack::PushL(MycontactDB);

    TContactIter iter(*MycontactDB);

    // Use Heap based descriptor for large/unknown size of contact items.
    TBuf16<128> aValue;

    const CContactIdArray* contactArray = MycontactDB->SortedItemsL();

    TInt cnt=contactArray->Count();

    for(TInt i=0;i<cnt;i++)
    {
        CContactItem* contactItem=NULL;

        contactItem= MycontactDB->OpenContactL((*contactArray)[i]);
        CleanupStack::PushL(contactItem);

        CContactItemFieldSet& fieldSet= contactItem->CardFields();
        TInt fieldCount=fieldSet.Count(); // This will give number
        // of contact fields.

        for(TInt index=0; index < fieldCount; index++)
        {
            CContactItemField& field = fieldSet[index];
            const CContentType& type = field.ContentType();
            if(!(type.ContainsFieldType(KUidContactFieldBirthday)))
            {
                TPtrC name =
                ->CardFields[index].TextStorage()->Text();
                aValue.Copy(name);
                Encrypt(aValue); // Call real encryption here
                contactItem->CardFields()[index].TextStorage()->SetTextL(aValue);
            }
        } //Inner for loop ends here
        MycontactDB->CommitContactL(*contactItem);
        CleanupStack::PopAndDestroy(contactItem);
    } //Outer for loop ends here
    CleanupStack::PopAndDestroy(MycontactDB);
}

// Actual encryption - can be changed into a complex one
void Encrypt(TDes& aValue)
{
    for(TInt iCount=0; iCount< aValue.Length();iCount++)
    {
        aValue[iCount]+=3;
    }
}
```

How_to_manage_contact_database_while_encryption

```
}
```

The following code deletes all contacts from Default contact Database:

```
// Deletes all contacts from default contacts DB as we have
// back up in encrypted form
void DelAllCnts()
{
    // open default contact database
    CContactDatabase* DefcontactDB= CContactDatabase::OpenL();
    CleanupStack::PushL(DefcontactDB);
    // An array to hold all contact items IDs
    CContactIdArray* cntArray = CContactIdArray::NewL();
    CleanupStack::PushL(cntArray);
    //create iterator for DefcontactDB
    TContactIter contact_iterator(*DefcontactDB);
    TContactItemId contact_id;
    //go through contacts while there any info
    while( ( contact_id = contact_iterator.NextL() ) != KNullContactId)
    {
        cntArray->AddL(contact_id);
    }
    // deletes all contacts
    DefcontactDB->DeleteContactsL(*cntArray);
    CleanupStack::PopAndDestroy(2);
}
```

The following code decrypts MycontactDB:

```
// Decrypt all contacts in my database
void DecryptAll()
{
    CContactDatabase *MycontactDB = CContactDatabase::OpenL(KFileName);
    CleanupStack::PushL(MycontactDB);

    TContactIter iter(*MycontactDB);
    TBuf16<70> aValue;

    const CContactIdArray* contactArray = MycontactDB->SortedItemsL();

    TInt cnt=contactArray->Count();

    for(TInt i=0;i<cnt;i++)
    {
        CContactItem* contactItem=NULL;

        contactItem= MycontactDB->OpenContactL((*contactArray)[i]);
        CleanupStack::PushL(contactItem);

        CContactItemFieldSet& fieldSet= contactItem->CardFields();
        TInt fieldCount=fieldSet.Count(); // This will give number
        // of contact fields.

        for(TInt index=0; index < fieldCount; index++)
        {
            CContactItemField& field = fieldSet[index];
            const CContentType& type = field.ContentType();
            if(!(type.ContainsFieldType(KUidContactFieldBirthday)))
            {
                TPtrC name =
                ->CardFieldsItem(index).TextStorage()->Text();
            }
        }
    }
}
```

Example code

How_to_manage_contact_database_while_encryption

```
        aValue.Copy(name);
        Decrypt(aValue);
        contactItem->CardFields()[index].TextStorage()->SetTextL(aValue);
    }
    } //Inner for loop ends here
    MycontactDB->CommitContactL(*contactItem);
    CleanupStack::PopAndDestroy(contactItem);
} //Outer for loop ends here
CleanupStack::PopAndDestroy(MycontactDB);
}

// Actual decryption
void Decrypt(TDes& aValue)
{
    for(TInt iCount=0; iCount< aValue.Length();iCount++)
    {
        aValue[iCount]-=3;
    }
}
```

The following code copies back all the contacts from the MycontactDB to Default contact database and deletes MycontactDB:

```
// Copy all contacts back to default contacts DB from My Db
void CopyBackToDefault()
{
    // open default contact database
    CContactDatabase* DefcontactDB= CContactDatabase::OpenL();
    CleanupStack::PushL(DefcontactDB);
    // open My contact database
    CContactDatabase* MycontactDB = CContactDatabase::OpenL(KFileName);
    CleanupStack::PushL(MycontactDB);
    //create iterator for MycontactDB
    TContactIter contact_iterator(*MycontactDB);
    TContactItemId contact_id;
    //go through contacts while there any info
    while( ( contact_id = contact_iterator.NextL() ) != KNullContactId)
    {
        // Read contact from own database
        CContactItem* contact_item = MycontactDB->ReadContactL(contact_id);
        CleanupStack::PushL(contact_item);
        // Add it to default database
        DefcontactDB->AddNewContactL(*contact_item);
        CleanupStack::PopAndDestroy(contact_item);
    }
    CleanupStack::PopAndDestroy(2);
    // delete own database file - no more needed
    CContactDatabase::DeleteDatabaseL(KFileName);
}
```

Example application

- [An example application on contact database](#)

Internal Link

- [Encrypt-Decrypt Contact Database](#)