

The new CallInterim API has been introduced with the Symbian 9.1 releases to replace the old Agenda Model API.

Contents

- [1 Connecting](#)
- [2 Insert new Appointment](#)
- [3 Insert new Todo](#)
- [4 Add old AgendaModel Alarm data to the new CCalEntry Appointment item](#)
- [5 Add old AgendaModel Alarm data to the new CCalEntry Todo item](#)
- [6 Delete entry](#)
- [7 Update Appointment entry](#)
- [8 Update Todo item](#)
- [9 Convert item CCalEntry IDs to old AgendaModel entry IDs](#)
- [10 Copy entry to an other one without Global ID](#)

Connecting

The CCalSession is the interface to the Calendar file. The instantiation of CCalSession will result in a connection to the Calendar Server:

Library required:

```
LIBRARY          calinterimapi.lib
```

Header files:

```
#include <calsession.h>
#include <calprogresscallback.h>
#include <caliterator.h>
#include <calentry.h>
#include <calentryview.h>
#include <caltime.h>
#include <calalarm.h>
#include <calcategory.h>
#include <caluser.h>
#include <calrrule.h>
#include <calinstance.h>
#include <calinstanceview.h>
```

```
class CCalendar : public MCalProgressCallBack
{
    CCalendar();
    void ConstructL();
    ~CCalendar();
    [...]
private:
    void OpenCalendarL();
    [...]
    CCalSession*          iCalSession
    CCalIter*             ; iCalIter
```

CallInterimAPI

```
CCalEntryView*      iCalEntryView
HBufC8*             ; iNext
TBool               ; iReady;
[...]
}
```

Source file:

```
#include "Calendar.h"

CCalendar::CCalendar()
{
}

CCalendar::~~CCalendar()
{
    if(iCalIter)
    {
delete iCalIter;
    }
    if(iCalEntryView)
    {
delete iCalEntryView;
    }
    if(iCalSession)
    {
delete iCalSession;
    }
    if( iNext )
    {
delete iNext;
        = iNext;
    }
}

void CCalendar::ConstructL()
{
    iReady=EFalse;
    iNext = HBufC8::NewL(50);
    OpenCalendarL();
}

void CCalendar::OpenCalendarL()
{
    // allocate and construct server
    // Check that calendar exists, and if not, create it.
    // Calendar does not exist until it is created by calendar app - or by us.
    iCalSession = CCalSession::NewL();

    //Open the default calendar file
    TRAPD(aErr, iCalSession->OpenL(KNullDesC));
    if(aErr == KErrNotFound)
    {
        (errTRAPEdDalSession->CreateCalFileL(iCalSession->DefaultFileNameL() ));
        (aErrTRAPEdDiCalSession->OpenL(KNullDesC));
    }

    iCalEntryView = CCalEntryView::NewL(*iCalSession, *this);
    iCalIter = CCalIter::NewL(*iCalSession);

    //Reset to the begin of the Calendar entries
```

CallInterimAPI

```
TPtr8 aAddress( iNext->Des() );
aAddress = iCalIter->FirstL();
}
```

Do not forget implement other functions!

```
// Called during calendar entry view creation
void CCalendar::Progress(TInt aPercentageCompleted)
{
}

void CCalendar::Completed(TInt aError)
{
    if(aError==KErrNone)
        iReady=ETrue;
}

// Returns whether or not progress notification is required
TBool CCalendar::NotifyProgress()
{
    // Progress notification is required
    return ETrue;
}

// Utility function to handle CCalEntry insert
// Destroy the RPointerArray
void DestroyRPointerArray(TAny* aPtr)
{
    RPointerArray<CCalEntry>* self = static_cast<RPointerArray<CCalEntry>*> (aPtr);
    self->ResetAndDestroy();
}
}
```

Insert new Appointment

To insert new item with a given ID you should create a CCalEntry item and fill out with the desired data.

```
void CCalendar::NewApptL(TInt aID)
{
    if(!iReady) //if this function is called too soon then return. Calendar opening
                //must finish first.
        return;

    //Create entry with the given new GUID
    HBufC8* guidBuf;
    TBuf8<30> tmpGuid;
    tmpGuid.Num( aID );
    guidBuf = tmpGuid.AllocL();

    CCalEntry* appt = CCalEntry::NewL(CCalEntry::EAppt, guidBuf,
    CCalEntry::EMethodNone, 0);
    CleanupStack::PushL(appt);

    //Some data
    appt->SetSummaryL( _L("Summary") );
    appt->SetLocationL( _L("Location") );
    appt->SetDescriptionL( _L("Description") );
}
```

CallInterimAPI

```
//Start / end date
TTime start;
start.UniversalTime();
TTime end;
end.UniversalTime();
TCalTime startCalTime;
startCalTime.SetTimeUtcL(start);
TCalTime endCalTime; //NullTTime()

//Comment out the next line if you do not want to set end time
endCalTime.SetTimeUtcL(end);

//Set it
appt->SetStartAndEndTimeL(startCalTime, endCalTime);

//Store this new Entry

RPointerArray<CCalEntry> entryArray;
CleanupStack::PushL(TCleanupItem(DestroyRPointerArray, &entryArray));
entryArray.AppendL(appt);
TInt success(0);
iCalEntryView->StoreL(entryArray, success);
entryArray.Reset();
CleanupStack::PopAndDestroy(&entryArray);
CleanupStack::PopAndDestroy(appt);
}
```

Insert new Todo

To insert new item with a given ID you should create a CCalEntry item and fill out with the desired data.

```
void CCalendar::NewTodoL(TInt aID)
{
    //Create an antry with the given GUID
    HBufC8* guidBuf;
    TBuf8<30> tmpGuid;
    tmpGuid.Num( aID );
    guidBuf = tmpGuid.AllocL();

    CCalEntry* todo = CCalEntry::NewL(CCalEntry::ETodo, guidBuf,
        CCalEntry::EMethodNone, 0);
    CleanupStack::PushL(todo);

    todo->SetSummaryL( _L("Summary") );

    //Priority
    TInt priority = 1;
    todo->SetPriorityL(priority);

    //Completed
    TCalTime calTime;
    TTime time; time.UniversalTime();
    calTime.SetTimeUtcL(time);
    todo->SetCompletedL(ETTrue, calTime);

    //End date
    TTime end;
    end.UniversalTime();
}
```

CallInterimAPI

```
TCalTime calEndDate;
calEndDate.SetTimeUtcL(end);
todo->SetStartAndEndTimeL(calStartDate, calEndDate);

//Store New Entry
RPointerArray<CCalEntry> entryArray;
CleanupStack::PushL(TCleanupItem(DestroyRPointerArray, &entryArray));
entryArray.AppendL(aTask);
TInt success(0);
iCalEntryView->StoreL(entryArray, success);
entryArray.Reset();
CleanupStack::PopAndDestroy(&entryArray);
CleanupStack::PopAndDestroy(todo);
}
```

Add old AgendaModel Alarm data to the new CCalEntry Appointment item

This code sample is useful if you port your application from the previous agenda model. In the old model you should use the `SetAlarm(TTimeIntervalDays aDaysWarning, TTimeIntervalMinutes aTime)`. In the new API you can set the alarm only by giving the offset time according to the start date of the entry.

```
//Get old alarm data
TTimeIntervalDays aDays = ...;
TTimeIntervalMinutes aMinutes= ...;

//tmp variables
TInt oneDayInMin = 1440;
TInt64 oneDayInMicroSec = 86400000000LL;
TInt64 oneMinInMicroSec = 60000000;
TCalTime startCalTime;

//Set the start time
startCalTime.SetTimeUtcL(start);
TInt offSet =
((startCalTime.TimeLocalL().Int64() % oneDayInMicroSec) / oneMinInMicroSec) -
aMinutes.Int();
if( offSet < 0 ) offSet = oneDayInMin - Abs(offSet);
TTimeIntervalMinutes tmp( offSet + aDays.Int() * oneDayInMin);
CCalAlarm* alarm = CCalAlarm::NewL();

if(alarm)
{
alarm->SetTimeOffset(tmp);
appt->SetAlarmL(alarm);
delete alarm;
}
```

Add old AgendaModel Alarm data to the new CCalEntry Todo item

This lines are similar to the appointment code, but now the alarm offset is calculated from the end date (due date).

CallInterimAPI

```
//Get old alarm data
TTimeIntervalDays aDays = ...;
TTimeIntervalMinutes aMinutes= ...;

//tmp variables
TInt oneDayInMin = 1440;
TInt64 oneDayInMicroSec = 86400000000LL;
TInt64 oneMinInMicroSec = 60000000;
TCalTime endCalTime;

//Set end date
endCalTime.SetTimeUtcL(dueDate);
TInt offSet =
((endCalTime.TimeLocalL().Int64() % oneDayInMicroSec) / oneMinInMicroSec) -
aMinutes.Int();
TTimeIntervalMinutes tmp( offSet + aDays.Int() * oneDayInMin);
CCalAlarm* alarm = CCalAlarm::NewL();
if( alarm )
{
    alarm->SetTimeOffset(tmp);
    todo->SetAlarmL(alarm);
    delete alarm;
}
```

Delete entry

```
void CCalendar::DeleteItemL(TInt aID)
{
    CCalEntry* aEntry = NULL;
    TBuf8<30> tmpGuid;
    tmpGuid.Num( aID );
    HBufC8* guidBuf = tmpGuid.AllocL();
    RPointerArray<CCalEntry> tmpArray;
    CleanupStack::PushL(TCleanupItem(DestroyRPointerArray, &tmpArray));
    iCalEntryView->FetchL( guidBuf->Des(), tmpArray );
    delete guidBuf;

    if( tmpArray.Count() == 0 )
    {
        CleanupStack::PopAndDestroy(&tmpArray);
        return;
    }
    aEntry = tmpArray[0];
    iCalEntryView->DeleteL(*aEntry);
    CleanupStack::PopAndDestroy(&tmpArray);
}
```

Update Appointment entry

Because the CCalEntryView->UpdateL() can't update all the attributes of an entry i use a tmpEntry to store the copied data to store the new modified entry, there are some entry swapping to avoid the data loss if a break occur. You can also implement this function using the CopyCalEntryL() function, described at the end of this document.

```
void CCalendar::UpdateApptL(TInt aID)
{
```

CallInterimAPI

```

CCalEntry* aAppt = NULL;
CCalEntry* tmpEntry = NULL;
TInt aID = GetInt(*aIDText);
TBuf8<30> tmpGuid;
tmpGuid.Num(aID);
HBufC8* guidBuf = tmpGuid.AllocL();
HBufC8* guidTmp;

TBool aFound = EFalse;

RPointerArray<CCalEntry> array;
CleanupStack::PushL(TCleanupItem(DestroyRPointerArray, &array));
TRAPD(aErr, iCalEntryView->FetchL(guidBuf->Des(), array));
if (!aErr && array.Count() > 0)
{
    = aAppt[0];
if( aAppt->EntryTypeL() != CCalEntry::EAppt ) aAppt = NULL;
    =aFound;
}

if (aFound)
{
    //This will create new unique UID
    * guidTmp = CalenInterimUtils::GlobalUidL();
/*
    //Check the existence of the newly created tmp guid
    TInt itemCount = 0;
    do
    {
        TInt guidNum = Math::Random();
        TBuf8<30> tmpGuid;
        tmpGuid.Num( Abs(guidNum) );
        guidTmp = tmpGuid.AllocL();
        RPointerArray<CCalEntry> tmparray;
        CleanupStack::PushL(TCleanupItem(DestroyRPointerArray, &tmparray));
        iCalEntryView->FetchL(guidTmp->Des(), tmparray);
        itemCount = tmparray.Count();
        CleanupStack::PopAndDestroy(&tmparray);
        if( itemCount > 0 ) delete guidTmp;
    }
    while( itemCount > 0 );
*/
    tmpEntry
        CCalEntryL(CCalEntry::EAppt, guidTmp, CCalEntry::EMethodNone, 0);
    CleanupStack::PushL(tmpEntry);
    CopyCalEntryL(tmpEntry);
    CleanupStack::Pop(tmpEntry);
}

CleanupStack::PopAndDestroy(&array);
if( tmpEntry ) CleanupStack::PushL(tmpEntry);

//Set new data:
tmpEntry->SetSummaryL(_L("Summary"));
// [...]

//Store it

if( tmpEntry ) CleanupStack::Pop(tmpEntry);
if (aFound)
{
    * guidUpdate = guidBuf->AllocL();
    CCalEntryEntry =

```

CallInterimAPI

```
    CCalEntry *pCalEntry = new CCalEntry(guidUpdate, CCalEntry::EMethodNone, 0);
    CleanupStack(updateEntry);
    CopyCalEntryBy(updateEntry);
    CleanupStack(updateEntry);

    *HBuff->GuidTmp = guidTmp->AllocL();

//Insert the tmp entry
    RPoint@CCalEntry> insertArray;
    CleanupStackTCleanupItem(DestroyRPointerArray, &insertArray));
    insertArray.Append(updateEntry);
    TInt(success);
    iCalEntryView(insertArray, success);
    insertArray;
    CleanupStackDestroy(&insertArray);
delete tmpEntry;

//Delete old
    RPoint@CCalEntry> deleteArray;
    CleanupStackTCleanupItem(DestroyRPointerArray, &deleteArray));
    iCalEntryView(guidBuf->Des(), deleteArray);
    CCalEntry *deleteEntry = deleteArray[0];
    iCalEntryView(*deleteEntry);
    CleanupStackDestroy(&deleteArray);

//insert new update Entry
    RPoint@CCalEntry> updateArray;
    CleanupStackTCleanupItem(DestroyRPointerArray, &updateArray));
    updateArray.Append(updateEntry);
    success;
    iCalEntryView(updateArray, success);
    updateArray;
    CleanupStackDestroy(&updateArray);
delete updateEntry;

//Delete tmp Entry
    RPoint@CCalEntry> deleteTmpArray;
    CleanupStackTCleanupItem(DestroyRPointerArray, &deleteTmpArray));
    iCalEntryView(deleteGuidTmp->Des(), deleteTmpArray);
    CCalEntry *deleteTmpEntry = deleteTmpArray[0];
    iCalEntryView(*deleteTmpEntry);
    CleanupStackDestroy(&deleteTmpArray);
delete deleteGuidTmp;
}
    delete guidBuf;
}
```

Update Todo item

The Todo entry update can be made in similar mode than the Appointment entry.

Convert item CCalEntry IDs to old AgendaModel entry IDs

If you are porting your application from the old Agenda API to the new CallInterim API, presumably you have head-ache how to convert the new alphanumeric IDs to the old integer ID-s. Here is an example:

CallInterimAPI

```

////////////////////////////////////
/*
Because the Global Uid(22 char length &HBufC8) of an entry
is out of range of TInt32...
This function must be called after the & CCalIter creation
and before any other functions.
*/

// Update the Global uids to be in the range of TInt32
void CCalendar::ConvertL()
{
    HBufC8* next = HBufC8::NewL(50);
    TPtr8 aAddress( next->Des() );
    aAddress = iCalIter->FirstL();

    while( next->Des() != KNullDesC8 )
    {
        //Checking if the ( UID is string) || (bigger then 2147483646)
        TBool toConvert = EFalse;
        <100>TBuf8 des8 = _L8("");
        (err_fetch, des8.Copy( next->Des() ) );
        if( err_fetch == KErrNone )
        {
            TLex8 tlex8(des8);
            TInt id = tlex8.Val(id);
            if( err_lex != KErrNone ) toConvert = ETrue;
        }

        //Update
        if( toConvert != EFalse )
        {
            RPointerArray<CCalEntry> fetchArray;
            CleanupStack::CleanupItem(DestroyRPointerArray, &fetchArray);
            iCalIter->FetchL(next->Des(), fetchArray);
            CCalEntry = fetchArray[0];

            //////////////////////////////////////

            //Convert only the Appontment & Todo
            if( aEntry->EntryTypeL() == CCalEntry::EAppt || aEntry->EntryTypeL() ==
                CCalEntry::ETodo )
            {
                //Check the existence of an entry with the new GUID
                * guidBuf;HBufC8
                TInt itemCount
            do
            {
                = Mfht::BandNum();
                <30>TBuf8 Guid;
                Num( AbstractNum );
                guidBuf.Guid.AllocL();
                <CCalEntry> tmparray;
                ::PushCleanupItem(DestroyRPointerArray,
                    &tmparray);
                ->FetchL(next->Des(), tmparray);
                = tmparray.Count();
                ::PopCleanupStack(&tmparray);
                if( itemCount > 0 ) delete guidBuf;
            }
            while( itemCount > 0 );

```

CallInterimAPI

```
        CCalEntry* updateEntry;
if( aEntry->EntryTypeL() == CCalEntry::EAppt )
    {
        = updateEntry
        ::NewL(CCalEntry::EAppt, guidBuf,
              CCalEntry::EMethodNone, 0);
    }
else
    {
        = updateEntry
        ::NewL(CCalEntry::ETodo, guidBuf,
              CCalEntry::EMethodNone, 0);
    }

//Copy: because updateEntry->CopyFromL( *aEntry, EDontCopyId );
//doesn't work :(
(aEntry->CopyFrom(*updateEntry));

<CCalEntry* modifyingArray;
::PushL(CCalEntryItem(DestroyRPointerArray,
&modifyingArray));
AppendL(*updateEntry);
TInt (numberOfEntries
->SetEntryViewingArray, numberOfEntries);
CleanupSpaceDestroy(&modifyingArray);

//Delete old
<CCalEntry* deleteArray;
CleanupSpaceDestroyItem(DestroyRPointerArray,
&deleteArray));
->FetchEntryViewingArray(UidL(), deleteArray);
* deleteEntry = deleteArray[0];
iCalendarView->Delete(*deleteEntry);
::PopAndDestroy(&deleteArray);
}

CleanupSpaceDestroy(&fetchArray);
}

aCalendar->NextL();
}

delete next;
= NH&K;
}
```

Copy entry to an other one without Global ID

This function is useful if CCalEntry CopyFromL(*aEntry, EDontCopyId) doesn't work. aSource and aTarget should exist before calling this function. All 2 entries should be the same type!

```
void CCalendar::CopyCalEntryL(CCalEntry* aSource, CCalEntry* aTarget)
{
    aTarget->SetSummaryL( aSource->SummaryL() );
    aTarget->SetDescriptionL( aSource->DescriptionL() );
    aTarget->SetLastModifiedDateL();
    aTarget->SetDTStampL( aSource->DTStampL() );
    aTarget->SetLocationL( aSource->LocationL() );
    aTarget->SetStatusL( aSource->StatusL() );
    aTarget->SetReplicationStatusL( aSource->ReplicationStatusL() );
    aTarget->SetPriorityL( aSource->PriorityL() );
}
```

CallInterimAPI

```
//On Nokia E60 - sometimes at Todo entries, getting startTime/endTime Leaves!  
//In this case create a new NullTTime Enty  
TCalTime startCalTime;  
TRAPD(err, startCalTime = aSource->StartTimeL());  
if( err != KErrNone )  
{  
    timeTTime::NullTTime();  
    startSetTimeUtcL(time);  
}  
  
TCalTime endCalTime;  
TRAPD(err1, endCalTime = aSource->EndTimeL());  
if( err1 != KErrNone )  
{  
    timeTTime::NullTTime();  
    endSetTimeUtcL(time);  
}  
  
aTarget->SetStartAndEndTimeL(startCalTime, aSource->EndTimeL());  
  
if( aSource->EntryTypeL() == CCalEntry::ETodo )  
{  
    TCalTime aSource->CompletedTimeL();  
if( time.TimeUtcL() != Time::NullTTime() ) aTarget->SetCompletedL  
    (ETrue time);  
    -aSource->CompletedTimeL().TimeUtcL(), ETrue, 3);  
}  
  
TCalRRule rule;  
if( aSource->GetRRuleL( rule ) != EFalse ) aTarget->SetRRuleL( rule );  
  
RArray<TCalTime> array;  
aSource->GetRDatesL( array );  
if( array.Count() > 0 ) aTarget->GetRDatesL( array );  
  
array.Reset();  
aSource->GetExceptionDatesL( array );  
if( array.Count() > 0 ) aTarget->SetExceptionDatesL( array );  
array.Close();  
  
CCalAlarm* alarm = aSource->AlarmL();  
aTarget->SetAlarmL( alarm );  
if( alarm )  
{  
    -aSource->AlarmL()->TimeOffset().Int(), 3);  
delete alarm;  
}  
  
//If,  
//Category, Attendee, Organizer, PhoneOwner, Method,  
//SequenceNumber, RecurrenceID, TzRules, LocalUid are not needed  
}
```

--Egeri 09:51, 8 June 2007 (UTC)