



<b>ID</b>	CS001291	<b>Creation date</b>	January 26, 2009
<b>Platform</b>	S60 5th Edition	<b>Tested on devices</b>	Nokia 5800 XpressMusic
<b>Category</b>	Flash Lite	<b>Subcategory</b>	PIM

**Keywords (APIs, classes, methods, functions):** Service.Calendar, calendar.GetList(), calendar.Add(), calendar.Delete()

## Overview

This code snippet demonstrates how to browse, update, and delete calendar entries of the default calendar using the Calendar Platform Service for Flash Lite supported from S60 5th Edition onwards.

## Preconditions

**Note:** The test device needs to have at least one calendar entry in the default calendar. Deleting an entry with this application will erase it from the default calendar, too.

## Source

```
// Import Platform Service Interface
import com.nokia.lib.Service;

/*****
** Set Device properties with fscommand2
*****/
// Enable Full Screen
fscommand2( "FullScreen", true );
// Set quality
fscommand2( "SetQuality", "high" );

/*****
** Define all variables and arrays
*****/
var outList;
var inParams;
var outParams;
var outputEntry;
var errorId;
```

## CS001291\_-\_Modifying\_calendar\_entries\_in\_Flash\_Lite

```
var type;
var summary;

var startYear;
var startMonth;
var startDate;
var startHours:Number;
var startMinutes:Number;
var startTime;

var endYear;
var endMonth;
var endDate;
var endHours:Number;
var endMinutes:Number;
var endTime;

var alarmYear;
var alarmMonth;
var alarmDate;
var alarmHours:Number;
var alarmMinutes:Number;
var alarmTime;

var entryId;
var entryLocalId;
var calendarEntry;
var resultValue;

var i = 0;

var idList:Array = new Array();
var localIdList:Array = new Array();
var typeList:Array = new Array();
var summaryList:Array = new Array();
var startList:Array = new Array();
var endList:Array = new Array();
var alarmList:Array = new Array();

// Create Listener Object.
var cbListener:Object = new Object();

// Assign function to Listener Object.
cbListener.change = function(event_obj:Object) {
    type = event_obj.target.selectedItem.label;
    error_txt.text = "Changed: "+event_obj.target.selectedItem.label;
}

type_cb.addEventListener("change", cbListener);

// Heading of the application
heading_txt.text = "Modify Calendar Entries";

// Create new Service object which has Contact data
var calendar = new Service("Service.Calendar", "IDataSource");

// Get list of the entries
listEntries();
// Define number of the last item
var lastItem = idList.length-1;
// Set information to the inputs
```

## CS001291\_-\_Modifying\_calendar\_entries\_in\_Flash\_Lite

```
setTexts();

/*****
** Function for setting the texts to inputs
*****/
function setTexts() {
    // Anniversary, DayEvent, Meeting, Reminder, ToDo

    nro_txt.text = i+1+"/"+idList.length;

    // Check which type of entry
    // Anniversary, DayEvent, Meeting, Reminder, ToDo
    if(typeList[i] != undefined) {
        type_txt.text = typeList[i];
    } else {
        type_txt.text = "";
    }

    if(summaryList[i] != undefined) {
        summary_txt.text = summaryList[i];
    } else {
        summary_txt.text = "";
    }

    if(startList[i] != undefined) {
        start_df.selectedDate = startList[i];
        startHours_txt.text = startList[i].getHours();
        startMinutes_txt.text = startList[i].getMinutes();
    } else {
        start_df.selectedDate = null;
        startHours_txt.text = "";
        startMinutes_txt.text = "";
    }

    if(endList[i] != undefined) {
        end_df.selectedDate = endList[i];
        endHours_txt.text = endList[i].getHours();
        endMinutes_txt.text = endList[i].getMinutes();
    } else {
        end_df.selectedDate = null;
        endHours_txt.text = "";
        endMinutes_txt.text = "";
    }

    if(alarmList[i] != undefined) {
        alarm_df.selectedDate = alarmList[i];
        alarmHours_txt.text = alarmList[i].getHours();
        alarmMinutes_txt.text = alarmList[i].getMinutes();
    } else {
        alarm_df.selectedDate = null;
        alarmHours_txt.text = "";
        alarmMinutes_txt.text = "";
    }
}

/*****
** Function for getting updated list of the entries
*****/
function listEntries() {
    // Define input parameters
    var inParams = {Type:"CalendarEntry"};
```

## CS001291\_-\_Modifying\_calendar\_entries\_in\_Flash\_Lite

```
// Define result value
var outParams = calendar.GetList(inParams);
if (outParams.ErrorCode == 0) {
    var outList = outParams.ReturnValue;
    var outputEntry = null;
    // Format arrays
    idList = [];
    localIdList = [];
    typeList = [];
    summaryList = [];
    startList = [];
    endList = [];
    alarmList = [];
    // Go through all calendar events and print them to the textbox on the
    // scene
    do {
        outputEntry = outList.next();
        if (null != outputEntry) {
            idList.push(outputEntry.id);
            localIdList.push(outputEntry.LocalId);
            typeList.push(outputEntry.Type);
            summaryList.push(outputEntry.Summary);
            startList.push(outputEntry.StartTime);
            endList.push(outputEntry.EndTime);
            alarmList.push(outputEntry.AlarmTime);

            } else {
                break;
            }
        } while (true);
    } else {
        // if errors trace them to the textfield
        errorId = outParams.ErrorCode;
        error_txt.text = "Error while listing: "+errorId+"\r";
    }
}

/*****
** Function for pressing the Prev button
*****/
prev_mc.onPress = function() {
    if(i == 0) {
        i = lastItem;
    } else {
        i--;
    }
    setTexts();
    error_txt.text = "";
}

/*****
** Function for pressing the Next button
*****/
next_mc.onPress = function() {
    if(i == lastItem) {
        i = 0;
    } else {
        i++;
    }
    setTexts();
    error_txt.text = "";
}
```

## CS001291\_-\_Modifying\_calendar\_entries\_in\_Flash\_Lite

```
}

/*****
** Fuction for updating the calendar entry when Save
** button is pressed.
** NOTE! Updating entry with this application won't erase
** other information of the entry.
*****/
save_mc.onPress = function() {

    type = typeList[i];
    summary = summary_txt.text;

    // Define Start Time for the entry
    startInput = start_df.selectedDate;
    startYear = startInput.getFullYear();
    startMonth = startInput.getMonth();
    startDate = startInput.getDate();
    startHours = startHours_txt.text;
    startMinutes = startMinutes_txt.text;
    startTime = new Date(startYear, startMonth, startDate, startHours,
        startMinutes, 0, 0);
    // Get universal time to set it to device
    startYearUTC = startTime.getUTCFullYear();
    startMonthUTC = startTime.getUTCMonth();
    startDateUTC = startTime.getUTCDate();
    startHoursUTC = startTime.getUTCHours();
    startTimeUTC = new Date(startYearUTC, startMonthUTC, startDateUTC,
        startHoursUTC, startMinutes, 0, 0);

    // Define End Time for the entry
    endInput = end_df.selectedDate;
    endYear = endInput.getFullYear();
    endMonth = endInput.getMonth();
    endDate = endInput.getDate();
    endHours = endHours_txt.text;
    endMinutes = endMinutes_txt.text;
    endTime = new Date(endYear, endMonth, endDate, endHours, endMinutes, 0, 0);
    endYearUTC = endTime.getUTCFullYear();
    endMonthUTC = endTime.getUTCMonth();
    endDateUTC = endTime.getUTCDate();
    endHoursUTC = endTime.getUTCHours();
    endTimeUTC = new Date(endYearUTC, endMonthUTC, endDateUTC, endHoursUTC,
        endMinutes, 0, 0);

    // Define Alarm Time for the entry
    alarmInput = alarm_df.selectedDate;
    if(!alarmInput && startInput){
        alarmInput = startInput;
    } else if(!alarmInput && endInput){
        alarmInput = endInput;
    }
    alarmYear = alarmInput.getFullYear();
    alarmMonth = alarmInput.getMonth();
    alarmDate = alarmInput.getDate();

    alarmHours = alarmHours_txt.text;
    if(!alarmHours && startHours){
        alarmHours = startHours;
    } else if(!alarmHours && endHours){
        alarmHours = endHours;
    }
}
```

## CS001291\_-\_Modifying\_calendar\_entries\_in\_Flash\_Lite

```
};

alarmMinutes = alarmMinutes_txt.text;
if(!alarmMinutes && startMinutes){
    alarmMinutes = startMinutes;
} else if(!alarmMinutes && endMinutes){
    alarmMinutes = endMinutes;
}
alarmTime = new Date(alarmYear, alarmMonth, alarmDate, alarmHours,
    alarmMinutes, 0, 0);
alarmYearUTC = alarmTime.getUTCFullYear();
alarmMonthUTC = alarmTime.getUTCMonth();
alarmDateUTC = alarmTime.getUTCDate();
alarmHoursUTC = alarmTime.getUTCHours();
alarmTimeUTC = new Date(alarmYearUTC, alarmMonthUTC, alarmDateUTC,
    alarmHoursUTC, alarmMinutes, 0, 0);

entryId = idList[i];
entryLocalId = localIdList[i];

// Anniversary, DayEvent, Meeting, Reminder, ToDo
if(type=="Anniversary" || type=="Reminder") {
    calendarEntry = {id:entryId, LocalId:entryLocalId, Type:type,
        StartTime:startTimeUTC, AlarmTime:alarmTimeUTC, Summary:summary
    };
} else if (type=="DayEvent" || type=="Meeting") {
    calendarEntry = {id:entryId, LocalId:entryLocalId, Type:type,
        StartTime:startTimeUTC, EndTime:endTimeUTC, AlarmTime:alarmTimeUTC,
        Summary:summary
    };
} else if (type=="ToDo") {
    calendarEntry = {id:entryId, LocalId:entryLocalId, Type:type,
        EndTime:endTimeUTC, AlarmTime:alarmTimeUTC, Summary:summary
    };
};

// Define input parameters
var inParams = {Type:"CalendarEntry", Item:calendarEntry};

// Define result value
var outParams = calendar.Add(inParams);

// Check if update success
if (outParams.ErrorCode == 0) {
    resultValue = outParams.ReturnValue;
    error_txt.text = resultValue;
} else {
    errorId = outParams.ErrorCode;
    error_txt.text = "Error while updating: "+errorId;
}
// Get updated list of the entries and set texts to the inputs
listEntries();
setTexts();
}

/*****
** Fuction for deleting the entry when Delete button
** is pressed. calendar.Delete() method used synchronously.
*****/
delete_mc.onPress = function() {

    // Define ID of the entry
```

## CS001291\_-\_Modifying\_calendar\_entries\_in\_Flash\_Lite

```
var idDeleteList = [idList[i]];
entryId = {IdList:idDeleteList};

// Define input parameters for the deletion
inParams = {Type:"CalendarEntry", Data:entryId};

// Define result data of the deletion
var outParams = calendar.Delete(inParams);
if (outParams.ErrorCode == 0) {
    error_txt.text = "Deletion success";
} else {
    errorId = outParams.ErrorCode;
    error_txt.text = "Error while deleting: "+errorId;
}

listEntries();
i = 0;
lastItem = idList.length-1;
setTexts();
}

/*****
** Function for pressing the Exit button.
*****/
exit_mc.onPress = function() {
    status = fscommand2("Quit");
    trace("QUIT");
}

stop();
```

## Postconditions

Start time, End time, alert time, and summary of the entry are displayed. You can browse, update, and delete entries.

## Example application

The following sample application has been tested in the Nokia 5800 XpressMusic (S60 5th Edition, Flash Lite 3.0).

[File:FlashLite Modifying Calendar Entries.zip](#)

## See also

- [Flash Lite Developer's Library](#)
- [CS001214 - Adding a calendar entry in Flash Lite](#)
- [CS001216 - Deleting a calendar entry in Flash Lite](#)
- [CS001222 - Listing calendar entries in Flash Lite](#)
- [CS001220 - Importing calendar entries in Flash Lite](#)
- [CS001218 - Exporting calendar entries to a text file in Flash Lite](#)