

## Symbian OS Essentials

### Details

Level:	Introduction
Duration:	3 days
Development language:	C++
Experience:	None, reasonable understanding of C++ and OO/UML experience is beneficial. Familiarity with the essential basics of Carbide.c++ IDE is also beneficial.



### Target audience

This course is for developers new to Symbian OS. The candidates gain a wide understanding of the Symbian OS platform and learn the basic concepts to write software for Symbian OS phones using native C++. The course acts as a foundation course to more specialised Symbian OS courses (Symbian OS Engine, Symbian OS AppUI and Symbian OS Internals).

### Description

The course introduces the candidate to the Symbian OS development tools of choice and explains how to develop applications using these tools for Symbian.

During the course UML class diagrams are used to illustrate C++ coding concepts. The course consists of approximately 50% exercises and practical work and 50% lectures. The exercises include C++ coding and use the TechView and Series 60 emulator to verify the results.

The course trains the essential and generic Symbian OS programming methodologies, all of which are applicable to all phone types, all emulators, all UI systems and all versions of the OS.

These methodologies include:

- Symbian system overview
- E32 kernel architecture
- resource management
- descriptors and arrays
- file management (file server, streams & store, persistent objects)
- active objects and using client/server architecture

Within the system overview device families are introduced, the design of Symbian OS is motivated and the developer learns the philosophy behind the application framework and generic technology (GT) layers. While developing Symbian applications developers will heavily re-use components in the Symbian OS.

The E32 kernel architecture presents an overview of the multitasking capabilities in Symbian OS. We cover processes and threads, priorities, context switching, kernel service calls, stacks and heaps.

As memory and other resources are limited on a mobile phone it is crucial that developers handle the management of these resource carefully. Within the resource management section the developer gets familiar with Symbian concepts like cleanup stack, 2 phase construction, memory leak detection, etc to ensure your applications will run stable and free of memory leaks.

Descriptors and arrays are introduced as the Symbian way to deal with string handling (Unicode) and binary data. Besides saving on memory descriptors will help developer to identify the out-of-bounds problem easily.

Almost any application has to deal with data and files. In the file management section different options are presented to write and read data to files. The candidate gets an overview of the file server, streams & stores, object embedding and how to create persistent objects within Symbian OS.

Besides memory processor speed is also limited on mobile phones. The active object framework is presented in order to create single threaded applications. Active objects will save context switches and give better performance at lower clock rates and ultimately also save battery power. Active objects are also used as an introduction to client/server architecture. This course mainly explains how to use the existing servers in Symbian OS as a client.

## Objectives

- Learn the differences between the three types of applications in Symbian: test or console applications and server (EXE), GUI application (EXE) and engines (DLL).
- Understand the functionality of fundamental components of Symbian OS (kernel, system layer, application framework, design philosophy)
- Learn to develop simple application within the emulator and to compile them for target phones.
- Use the development environment of choice competently
- Debug and test code (plus strategies)
- Use the memory management model
- Use descriptors, arrays and the file system
- Use Active object framework within your own applications
- Use the standard Symbian servers as a client

## Tools and platform

During the course we will use the Nokia Carbide.c++ OEM v2 IDE environment. Carbide.c++ can be downloaded from Forum Nokia free of charge.

Exercises and course materials are adapted for the Series 60 5<sup>th</sup> Edition SDK emulator.

A number of exercises will be run in the Symbian TechView UI. These exercises require features which are not present in the Series 60 environment or malfunction in the Series 60 environment.

## Agenda

### *Day 1*

#### Introduction to Symbian OS

- Key features
- Key platform components and their organisation
- Application architecture: Engine and Uis
- Overview of Symbian OS Development:
  - BAKs, DevKits, CustKits and SDKs: what they're used for.

#### Symbian OS Development Tools

- DevKit, CustKit, SDK installation, usage and documentation
- Basic CodeWarrior/Carbide commands
- Symbian platform build tools
- Unique identifiers
- Differences between target and emulated platforms

#### Symbian OS Application Development

- Building programs: GUI vs. console apps
- Using simple programs as templates
- Debugging and testing techniques
- Special Symbian OS coding conventions

### *Day 2*

#### System structure

- Architecture and implications
- Kernel mode vs user mode operation
- Types of function calls (user, executive and server request)

#### Resource Management

- Overview of memory leaks
- Detecting memory leaks using the memory leak and alloc failure tools
- Two phase construction pattern
- Exception handling using TRAPD
- Using the cleanup stack for resource management

#### Descriptors

- Motivation for descriptors (c.f. C strings)
- Descriptor classes and their key features

#### Arrays

- Dynamic array overview
- Use of segmented vs flat arrays
- Arrays for fixed/variable sized elements

## *Day 3*

### Active Objects

- Why AOs
- Life cycle of AOs
- CActive and CActiveScheduler classes
- Implementing simple AOs

### Client/Server architecture

- Client Server architecture overview
- Sessions and subsessions
- Associated cleanup issues

### File & Store

- Using file server sessions
- Basic direct stream store functionality
- Overview of different stream stores