



Contents

- [1 Purpose](#)
- [2 Architectural relationships](#)
- [3 Description](#)
 - ◆ [3.1 General properties](#)
 - ◇ [3.1.1 Server](#)
 - ◇ [3.1.2 Session](#)
 - ◇ [3.1.3 Sub-session](#)
 - ◇ [3.1.4 Message](#)
- [4 Links](#)

Purpose

Provides the Symbian OS client-server framework, by which a program can offer services to multiple other programs. Servers also handle resources on behalf of multiple clients.

All Symbian OS developers should have a general understanding of this API in order to understand the design of many Symbian OS system APIs. In specialised circumstances, developers may also create their own server programs.

Architectural relationships

Many important Symbian OS system APIs use the client-server framework to provide services to client programs: for example, the *Windows Server*, *File Server*, and *Messaging*. In some cases, such APIs provide extensive client-side classes that hide the direct use of the client-server interface from the client program.

Description

The API has four key concepts: server (`CServer2`), session (`CSession2` and `RSessionBase`), sub-session (`RSubSessionBase`), and message (`RMessage2`, and `RMessagePtr2`).

General properties

A server program offers services to other processes through a client interface API that it defines. Clients and servers use a message passing protocol to communicate.

Client-server is usually chosen, rather than a conventional shared library, to provide services when one or more of the following is required:

- management of shared system resource
- asynchronous services
- the protection offered by running in a separate process from clients.

A client-server implementation supplies a server program executable, and a .DLL containing the client-side interface.

Server

The server is the central class of any server program. It is responsible for handling requests by clients to establish a connection to the server.

The base server interface is provided by `CServer2`.

Session

The session is the channel of communication between a client and a server.

The base client-side session interface is provided by `RSessionBase`. An implementation derives from this to define the functions that it wants to expose to clients.

The corresponding server-side session base classes is `CSession2`. A session can be shared between different client threads if the server marks the session as sharable. An implementation defines in a derived class how client messages should be handled.

Sub-session

The sub-session presents an efficient refinement of a session where a client wants multiple simultaneous uses of a server. For example, with the File Server, each opened file is handled through a separate sub-session.

The base client-side sub-session interface is provided by `RSubSessionBase`. An implementation derives from this to define the functions that it wants to expose to clients.

A server implements a corresponding sub-session class based on `CObject`, `CObjectCon` and `CObjectIx`, the Reference Counting Objects API.

Message

The message is the information passed between client and server. It consists of a code that identifies the type of request that the client is making, and up to four 32-bit data arguments together with information about each argument's type, width and accessibility. It is also possible to pass just the code, with no arguments.

Clients do not use messages directly; they use a `TIPCArgs` object to package the message information that is to be sent to the server.

Server-side sessions and subsessions access this information through an `RMessage2` object.

Links

- [Using Client-Server Architecture in symbian](#)
- [Inter Process Communication in Symbian](#)
- [Client side implementation of a server](#)
- [How to create a server from scratch](#)