

# Open Print Assist Application Framework Kernel Architecture Proposal

## Summary

The OPA framework's goal is to cut development time and therefore project costs by providing core services common to most GUI based applications in a flexible, re-usable kernel. This core is supported by a modular architecture that allows additional code, developed separately, to be plugged in and executed at run-time. The kernel is highly scalable, supporting small desktop applications to powerful distributed systems.

## Motivation

Object oriented programming aims to maximise code reuse and reduce the time to develop business applications. Unfortunately, often this reuse is not maximised and complicated, difficult to maintain systems are the norm. It is imperative that, from the beginning, an application allows concurrent development to be conducted without serious complication. The use of well-defined interfaces and a directory service that allows clients to locate services exported by servers can facilitate the achievement of this goal.

## Requirements

GUI based client applications share many requirements that are often developed from scratch with each new project. A re-usable and flexible framework can provide these core services so that application development may have a head start. At present, OPA has identified the following as candidate services:

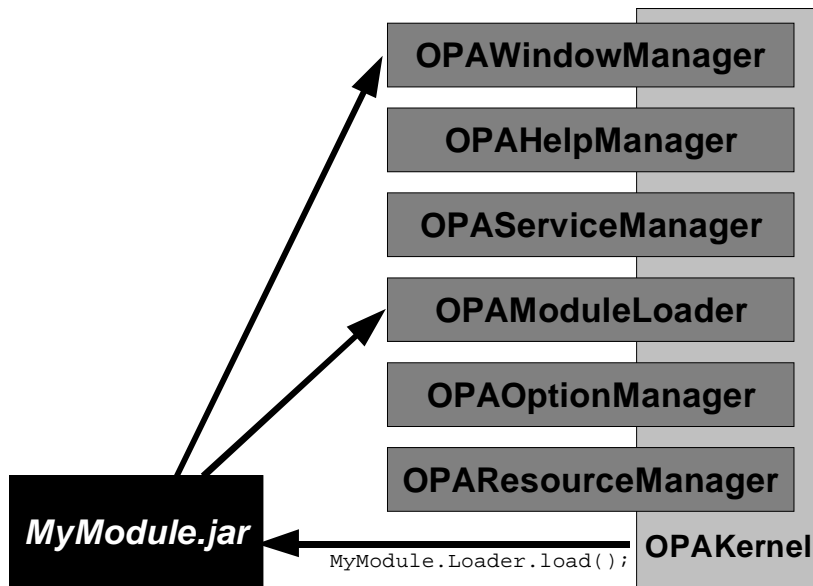
- Options management
- Help management
- Window management
- Persistence management
- Security management
- Resource management
- GUI construction
- Directory service, and
- Logging management.

## Design

The kernel is the central point of the application. The services noted above are contained in 'manager' objects, available from the kernel. The framework services are available through the manager APIs.

The managers implement the interfaces defined in the `org.opa.kernel` package. Clients, other kernel services or installed user supplied modules, can access the kernel and the manager objects it contains through a reference supplied to modules when they are loaded into the application.

User modules may also use the services exported by other user modules. When a module is loaded into the application, a specially supplied user `Loader` object performs any work required by the module, such as registering help, options or directory services with the kernel. Once a module is loaded, others can browse the directory service locating implementations of the interfaces they require. The interface to a module should be provided in as few classes as possible, ideally a single facade class.



☒ *OPAFramework overview*

NB. This diagram represents only the *planned* interfaces at the time of publication of this proposal. Actual planned and/or implemented interfaces may vary. Please check the latest documentation on the project page at: <http://openprintassist.sourceforge.net>