

Open Print Assist Developer Guide

v0.3

WARNING: OPA is in early development. The current API is likely to change in future releases.

Contents

About.....	2
Overview of OPA.....	2
OPA Kernel.....	2
OPAManager Classes.....	3
OPAManager Functions.....	3
Module and OPAManager Interaction.....	4

About

This guide is aimed at those developers wishing to contribute to development of the OPA framework. Those wishing to *use* OPA to develop a separate application should refer to OPA User Guide.

Please note that at this stage the OPAKernel API (the interface to the OPAKernels services) is constantly evolving and is subject to change.

Overview of OPA

The OPA project is developing a Java based GUI application framework that provides services common to most GUI applications. The ultimate goal of OPA is to develop a print management system based on this framework.

The framework provides code useful to many applications so that developers have a head start. This saves valuable time and effort for new projects.

OPA is based on a modular architecture. The framework itself is generally used UNMODIFIED. That is, developers using the framework supply MODULES (for information on creating modules for OPA, see the OPA User Guide). These modules perform the business function of the application, the real work. The modules use the services provided by OPA such as window management (eg. creating dialog boxes), option management (eg. storing user preferences) and resource management (eg. providing access to localised text or icons common to GUI applications) and more.

By providing common services in a framework and using a modular architecture, application development can be easily distributed. Geographically isolated teams can work on separate modules with little need for interaction. To access the services that another module provides, a module only needs to know the interface that the other module implements.

When the interface is known a module can ask OPA to find a service (object instance – another module) implementing the interface, or even create a new instance if one doesn't exist.

OPA Kernel

The OPA framework is based around a single class, OPAKernel. OPAKernel holds other OPAManager classes. These manager classes provide more specific functionality. For example the OPASWindowManager interface is responsible for creating dialogs and windows, finding panels and creating the menus and toolbars at startup.

OPAKernel is basically just a run() method which executes statements to start the OPAManagers. OPAKernel also has getter methods so that modules can access (AND USE!!) the OPAManager classes.

OPAKernel also has an OPAModuleLoader class which, obviously enough, loads modules. It is responsible for reading the module list in the users home directory (ordinary Java Properties file), finding the modules (JAR files), mounting the modules (with java.net.URLClassLoader), then calling a special class (org.opa.kernel.Loader).

The Loader class is very simple. There is only ONE important method, load(). This method is

overridden by module developers. It should have instructions to use the OPAManagers in OPAKernel. For example, when a module is loading, it might ask OPAKernel (by using OPAOptionManager) to add an option panel to the main application option panel. Or, a module might ask OPA (by using OPAHelpManager) to add the module help to the applications main help file. Or again, it might ask OPA (by using OPALogManager) to log a message like: "Loading my module. Everything OK!"

OPAManager Classes

There are seven (7) OPAManager classes. They are:

- OPAHelpManager
- OPAOptionManager
- OPAServiceManager
- OPALogManager
- OPAWindowManager
- OPAActionManager
- OPAResourceManager.

Some of these interfaces are VERY simple. For example, OPAHelpManager uses the Java Help package from Sun Microsystems. The interface is a bridge to the Java Help interface. OPALogManager is similar.

Some of these interfaces have implementations. Some do not, or are very unstable. Below is some information about the kinds of tasks that each OPAManager class is expected to do.

OPAManager Functions

This section shows WHAT each manager is expected to do. These sequence diagrams:

DO

- Give a broad idea of the goals of each OPAManager, and
- Help explain where responsibility is allocated.

But they DO NOT:

- Give detail of the interfaces of each manager. For example, the single OPALogManager function "Log message" may need MANY method interfaces. One for each level (low, medium, high etc).
- Cover the implementation details of the OPAManager classes. OPAManagers may need to interact with OPAKernel, other managers or the JVM to perform these functions.
- Cover details of other subsystems, such as the relation between the JPanel subclasses used in OPA (JPanel --> OPAPanel --> OPAOptionPanel etc)

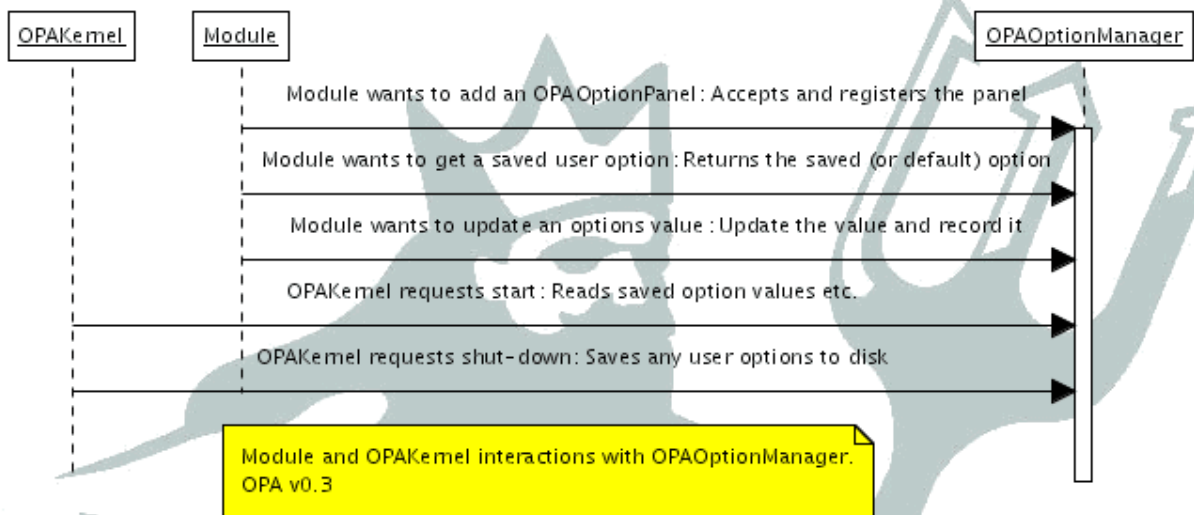
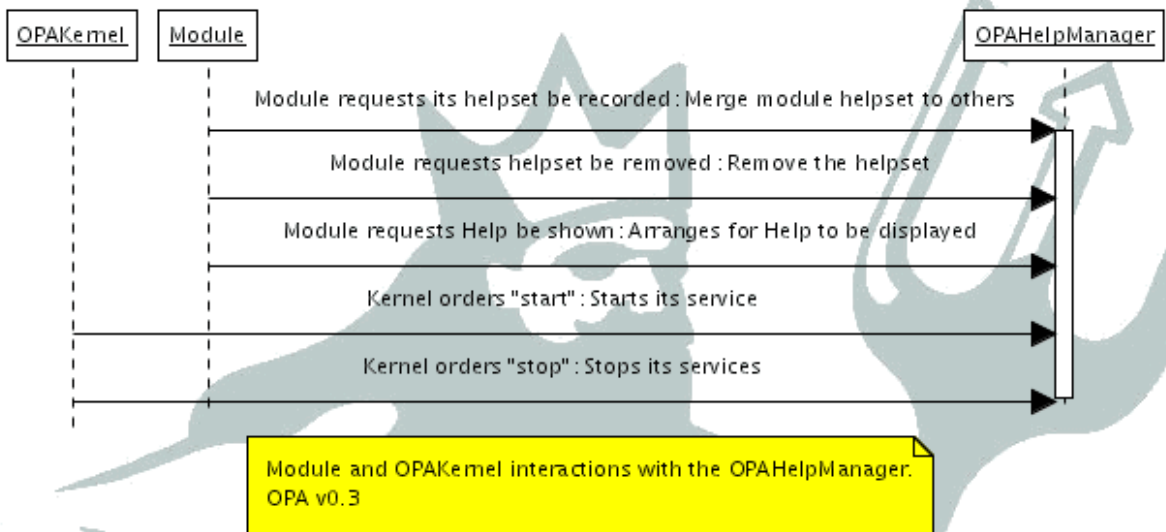
Module and OPAManager Interaction

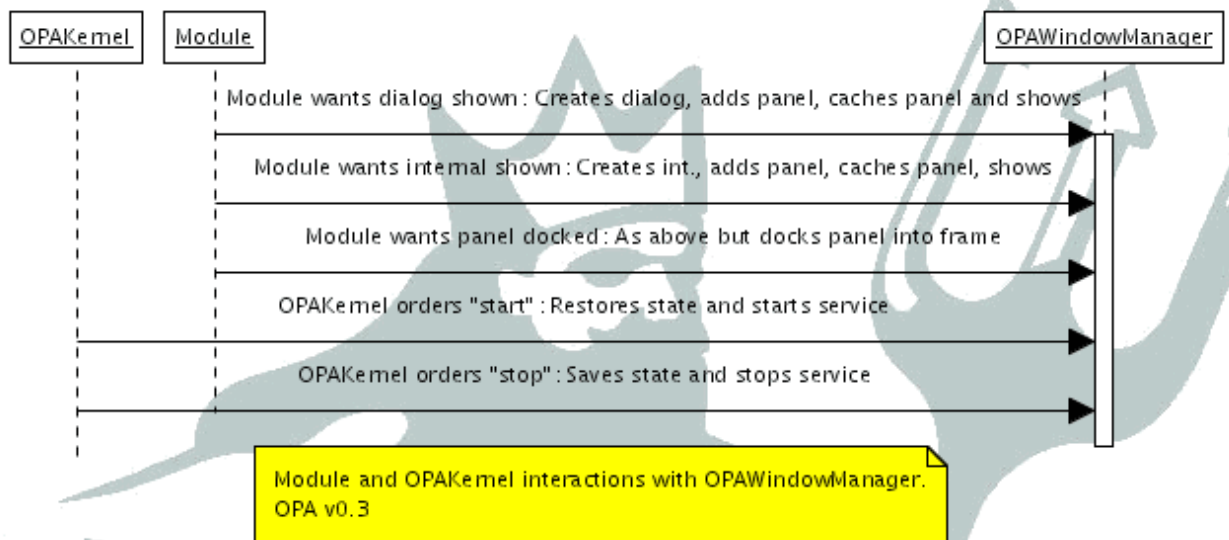
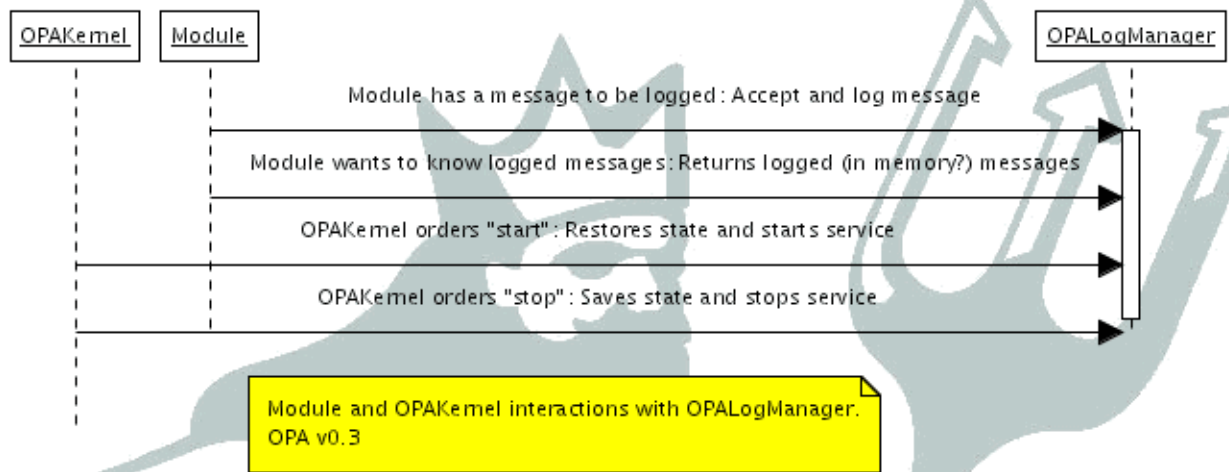
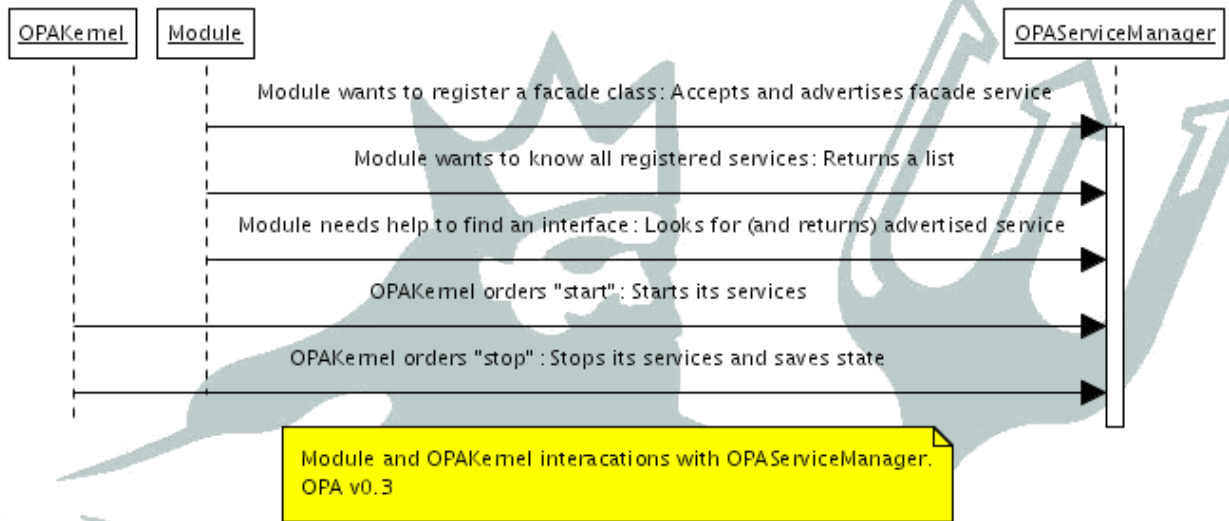
In the following sequence diagrams, stimuli (causes) are shown with arrowed lines. Each line has a name, then a colon (":"), then an action performed (or delegated) by the manager class. For example, the OPAHelpManager diagram shows:

"Module requests its helpset be recorded: Merge module helpset to others".

The *"Module requests its helpset be recorded"* is the starting action. When this happens, the OPAHelpManager will respond by doing...*"Merge module helpset to others"*.

Note: These diagrams were developed with Poseidon for UML Community Edition 2.1.1







Module and OPAKemel interactions with OPAActionManager.
OPA v0.3



Module and OPAKemel interactions with OPAResourceManager.
OPA v0.3