Introduction to the API 1

A SunSoft goal is to define the Architectural Interfaces of Solaris. There are two reasons:

- The system interface is an effective "contract" with our customers. We tell our customers exactly what we offer them to use, and help ensure that only the official (intended) interface is used.
- We use the definition to ensure that we honor this contract. Interfaces we offer in a particular version of the product are preserved in future versions of the product. Thus, we maintain upward compatibility in subsequent releases of Solaris.

The Programming Interface

Solaris offers many kinds of "interface", such as: the programming interface, elements of the user interface, protocols, and rules about naming and the locations of objects in the file system. One of the most important interfaces to the system is the programming interface - the one offered to developers. The programming interface has two major parts: one seen by developers of applications, which we call the API, and one seen by developers of system components such as device drivers and platform support modules, which we call the SPI (system programming interface).

Each programming interface to Solaris is also "visible" to the developer at two levels, source level and binary. When we use the acronyms API and SPI, we indicate the source level programming interface to the system. We use the terms Application Binary Interface (ABI) and System Binary Interface (SBI) to

indicate the binary interfaces corresponding to the respective source level programming interfaces. (Because the phrase "the ABI" can be confused with other binary interfaces, we refer to the "Solaris ABI" only by name.)

Interface Functions

The Solaris 2.x functions discussed in this manual are the interfaces between the services provided by the kernel and application programs. `The` functions described in Sections 2 and 3 of the *Solaris 2.5 Reference Manual AnswerBook* are an application's interface to the Solaris 2.x operating system. These functions are how an application uses facilities such as the file system, interprocess communication primitives, and multitasking mechanisms. This manual is one of a set that describe major elements of the API. Other manuals in the set are *STREAMS Programming Guide*, *Multithreaded Programming Guide*, *Transport Interfaces Programming Guide*, etc.

When you use the library routines described in sections 2 and 3 of the *Solaris 2.5 Reference Manual AnswerBook*, the details of their implementation are transparent to the program. For example, the function `read` underlies the `fread` implementation in the standard C library.

A C program is automatically linked to the invoked functions when you compile the program. The procedure might be different for programs written in other languages. See the *Linker and Libraries Guide* for more information.

Libraries

Solaris provides both static and dynamic implementations of libraries. Static libraries do not provide an interface, they provide only an implementation. The application programming interface of Solaris is made available to developers through the shared libraries (also called shared objects). In the runtime environment, a dynamic executable and shared objects are processed by the runtime linker, to produce a runnable process. The official API to the system is the interface between an application and the dynamic shared libraries.

Static libraries

The traditional, static, implementation of libraries (.a files or archives), do not separate the application programming interface from its implementation (the contents of the library). When an application is linked to a static library, the

object code that implements that library is bound into the executable object resulting from the build. The source-level programming interface to the library may be preserved, but the application must be relinked to produce an executable that runs on a later version of an operating system. Future binary compatibility is only assured when shared libraries are used.

The presence of static libraries is a historical artifact and there is no mechanism to define their interfaces in a way that is separate from their implementation. For this reason, use of static libraries should be avoided by new applications.

Dynamic libraries

Unlike the static libraries, shared libraries do separate the application programming interface from the implementation. The interface is bound to an implementation of the library only at runtime. This allows SunSoft to evolve the library's implementation - such as changing internal interfaces, while maintaining the API and preserving binary compatibility with applications built against it.

Interface Taxonomy

The Interface Taxonomy classifies commitment level of an interface. The commitment level identifies who may, or how to, use the interface. Definitions:.

Open specification	An interface specification that we publish, which customers can use freely (build products that use our implementation of the interface). Others are free to provide alternative implementations without licensing or legal restrictions.
Closed specification	An interface specification that we do not publish. On which we do not want customers to build products and of which we do not want others to build alternative implementations.
Compatible change	A change to an interface or its implementation has no effect on previously valid programs.
Incompatible change	A change to an interface or its implementation that makes previously valid programs invalid. This may include bug fixes or performance degradation. This does not include programs which depend on unspecified "artifacts of the implementation".

Standard Classification

Specification:	Open
Incompatible Change:	major release (X.0)
Examples:	POSIX, ANSI-C, Solaris ABI, SCD, SVID, XPG, X11, DKI, VMEbus, Ethernet

Standard interfaces are those whose specification is controlled by a group outside of Sun. This includes standards such as POSIX and ANSI C, as well as industry specifications from groups such as X/Open, the MIT X-Consortium, and the OMG.

Public Classification

Specification:	Open
Incompatible Change:	major release (X.0)
Examples:	Sun DDI, XView, ToolTalk, NFS protocol, Sbus, OBP

These are interfaces whose specification is completely under Sun's control. We publish the specification of these interfaces and commit to remain compatible with them.

Classification

Specification:	Open
Incompatible Change:	minor release (X.Y)
Examples:	VFS interface, vm kernel interfaces, libkvm

Uncommitted interfaces are available for use by customers, but they lack the commitment that comes with a Public interface. We often publish specifications of the current versions of these interfaces in a "this is how it works in this release, but we may change it next release" form. Use these interfaces at your own risk in experiments.

Some of these interfaces are ones we would like to elevate to “Public” status, but at the moment we don’t feel confident enough in them to commit to the compatibility constraints of a Public interface. The VFS and vm interfaces are examples of this.

Obsolete Classification

Specification:	None
Incompatible Change:	Minor release (.X.0)
Examples:	RFS

An interface no longer in general use. An existing interface can be downgraded from some other status (such as Public or Standard) to Obsolete through a standard proactive program to communicate the change in commitment to customers.

A change in commitment requires one year’s notice to the customer base and the Sun product development community of the intended obsoleting of the interface. A full year must elapse before delivering a product that contains a change incompatible with the present status of the interface.

Acceptable means of customer notice includes letters to customers on support contracts, release notes or product documentation, or announcements to customer forums appropriate for the interface in question.

The notice of obsolescence is considered to be “public” information in that it is freely available to the customers. It is not intended that this require specific actions to “publish” the information, such as press releases or similar forms of publicity.

1