

White Paper



# The Advantages of the SGI® IRIX® 6.5 Release Process

SGI Developer Forum 2002

Casey Leedom

Since its initial release in May 1998, IRIX® 6.5 has successfully operated under a mandate of application compatibility, system reliability, and release predictability across the entire range of supported SGI® systems. This is especially remarkable given that, during this time, many major features, new hardware products, and an entirely new system platform line have been deployed.

This article explains how compliance with the IRIX 6.5 mandate is achieved while allowing for the development of important new software and hardware products. It describes the release process from start to finish, first principle goals, and engineering development methodology.

The IRIX 6.5 release process was designed to meet the needs of SGI customers, third-party partners, and internal development teams. Key members of each constituency were consulted to determine a core set of goals that releases would be required to meet. These goals, which are the metrics for judging the success of each release and which provide the foundation of the IRIX 6.5 release model, are as follows:

- IRIX 6.5 must run on all platforms. There will be no hardware-specific releases.
- IRIX 6.5 must maintain application compatibility forward and backward across all releases and all platforms. No release of IRIX 6.5 can break applications built on older IRIX 6.5 releases. Any application built on any particular release of IRIX 6.5 must be able to run on all previous IRIX 6.5 releases [unless new hardware or software features are required, of course]. An application must be able to run on all platforms [unless hardware-specific features are required].
- IRIX 6.5 must be reliable. Performance degradation and failures in existing functionality cannot be allowed. The decision to upgrade to the latest release of IRIX 6.5 must always be a easy one. An upgrade must provide significant benefit at negligible risk.
- IRIX 6.5 must be released on a predictable schedule so that all parties can plan appropriately. Also, new releases must occur often enough that SGI hardware releases can occur on a timely basis.
- Both small- and large-scale development efforts must be accommodated. IRIX is a complex system that supports many hardware and software products used by customers to solve diverse technical and creative problems. IRIX is under constant development to meet the needs of these customers, and to allow SGI's own engineers and independent partners to develop new products

and technologies. Engineers around the world must be able to make changes, both significant and less significant, on a daily basis.

The first two goals were discussed at length in "The Mandate of Application Compatibility in IRIX 6.5," by Casey Leedom, P/N 007-4405-001. [To read this white paper, see <http://docs.sgi.com/tpl.cgi/view/007-4405-001>.] This article addresses the last three goals and discusses additional advantages of the release model in the following sections:

- The IRIX 6.5 Release Process is Predictable and Standardized
- IRIX 6.5 Accommodates Large, Long-term Product Engineering Efforts
- IRIX 6.5 Accommodates New APIs
- The IRIX 6.5 Release Stream Offers Maintenance and Feature Branches
- IRIX 6.5 Has an Extensive and Solid Future

## The IRIX 6.5 Release Process is Predictable and Standardized

A new version of IRIX 6.5 is released once every three months. This reliable schedule allows customers to plan hardware and software releases, system upgrades, preventive maintenance, and service points.

To ensure application compatibility and system reliability, a daily build-and-test cycle is run across the entire release on all supported system platforms. The daily test cycle is augmented with weekly and monthly tests that are more extensive and rigorous. This testing includes both application and stress-test coverage as well as analytic tests that explicitly compare existing software interfaces to the official IRIX 6.5 application interface signatures.

All engineering changes require review and approval before modifications can be made. If a change breaks the build or a test, that change is removed unless it can be addressed by the developer or the release engineering staff before the build-and-test cycle on the following day.

However, no amount of testing and review can catch all problems. Therefore, the three-month release cycle is split into three phases [A, B, and C] of approximately one month each. During these phases, code changes are subject to increasingly strict requirements, as follows:

- During phase A, general changes are allowed after appropriate review and approval.
- During phase B, only changes that solve significant customer issues, and bug fixes for problems introduced in phase A, are allowed.

- During phase C, only bug fixes that are critical to the release are allowed.

At all times, regressions are given the highest priority. A core philosophy is that the system must be of sufficient quality to be released to customers at any time during the process.

Two “early-access” releases are generated during each release cycle, one near the beginning of phase B and one near the beginning of phase C. These early-access releases are installed on SGI’s own critical production systems and are provided to external partners who want to participate in the early-access program. This supports the philosophy that IRIX 6.5 must always be of sufficient quality to be released. It further ensures reliability and application compatibility, and it provides some of our most important customers with early access to new hardware and software features.

The composition of the IRIX 6.5 release team is another innovative feature of the release model. Each release team is coordinated by professional project management staff, and team members represent all parts of the company, including not only engineering, but also, for example, manufacturing, technical publications, and customer service.

Most of the project managers and release team members play long-term roles that span multiple releases and thus provide valuable continuity. The engineering representation is handled differently: with each release, two new engineering managers and two new technical lead engineers are chosen to form the core engineering portion of the release team. This serves two critical purposes:

- The incoming engineering managers and technical leads offer fresh perspectives and insights regarding the ongoing release process.
- The SGI engineering staff receives valuable exposure to the IRIX 6.5 release process and learns how to manage other software projects more effectively.

All release team members have well-defined roles and responsibilities. Regular meetings and goal-oriented discussion forums, such as newsgroups and e-mail lists, facilitate communication and provide frequent “reality checks” as to the state of the release.

## **IRIX 6.5 Accommodates Large, Long-term Product Engineering Efforts**

Not all product engineering efforts can be handled within the quarterly framework of the IRIX 6.5 release

model. Some products or features require months, or even years, of development effort. This is incompatible with the central tenet that IRIX 6.5 must remain ready or almost ready to be released at all times.

Therefore, long-term IRIX 6.5 engineering efforts occur in cloned environments [“engineering trees”] that are synchronized with the main line IRIX 6.5 product. These engineering trees are subject to the IRIX 6.5 release goals, such as compatibility and reliability, and an appropriate build-and-test plan is followed, but the IRIX 6.5 release timeline is not a constraint.

When a product in an engineering tree is ready to be released, the owners of the engineering tree must work with the central IRIX 6.5 release team to plan the release of the product. This planning includes choosing an appropriate release of IRIX 6.5 into which the product will be merged, establishing a merge schedule, setting testing criteria, and choosing the responsible parties for each phase of the merge. Before the merge, the engineering tree is fully tested. After the merge, the entire main line IRIX 6.5 product is tested yet again. If any functional failures or unacceptable performance degradations are noted, the merge is cancelled.

## **IRIX 6.5 Accommodates New APIs**

Much of the release work involves creating new software functionality, including new application programming interfaces [APIs]. Yet the introduction of a new API can create compatibility problems, because an application that uses a new interface cannot run on a release of IRIX 6.5 that existed prior to the deployment of that interface. This conflict between the need for new APIs and the need for cross-release compatibility is addressed as follows:

- Once a new API is introduced, it is never removed. Thus, once an API is supported in IRIX 6.5, it will always be supported in IRIX 6.5.
- The frequent, reliable releases of IRIX 6.5 allow safe upgrades to releases that support needed software and hardware capabilities.
- All new APIs are introduced with an “optional symbol facility.” This facility allows an application to determine easily if a feature that it requires is present. If the application finds this feature is missing, it can exit gracefully, perhaps providing an informative error message, or it can operate in a way that does not depend on that feature. Without the optional symbol facility, the application would simply abort if it tried to use the missing feature.

The optional symbol facility is demonstrated in the following two examples. IRIX 6.5.10 introduced `utimets()`, a nanosecond-accurate version of the traditional `utime()` system call. An application may want to use `utimets()` if it is available, but it can use the older `utime()` interface if it has no choice. Sample application code that implements this choice would be similar to the following:

```
if (_MIPS_SYMBOL_PRESENT(utimets)){
    utimets(file, ts);
} else {
    struct utimbuf tb;
    tb.actime = ts[0].tv_sec;
    tb.modtime = ts[1].tv_sec;
    utime(file, &tb);
}
```

The following code fragment was written for an application that needed to use the Job Limits facility, which was introduced in IRIX 6.5.7f.

```
if (_MIPS_SYMBOL_PRESENT(getjid) ==
0) {
    fprintf(stderr,
        "This application uses Job Limits
        facilities and thus \n"
        "can only be run under IRIX 6.5.7f
        or later. Sorry.\n");
    exit(EXIT_FAILURE);
}
```

For more information about the optional symbol facility, see the following online man page:  
`optionalsym(1)`

## The IRIX 6.5 Release Stream Offers Maintenance and Feature Branches

The IRIX 6.5 release stream offers the following parallel release branches, which are both held to the same stringent reliability and compatibility standards. These branches fulfill the differing needs of customers, without compromising any of the core goals of the IRIX 6.5 release model.

- **Maintenance Branch.** This branch contains bug fixes, performance enhancements (including new hardware platform support), and standards-based changes. This branch allows customers to remain aware of important changes. It is available at the following website: <http://support.sgi.com/colls/patches/tools/relstream/index.html>
- **Feature Branch.** This branch is a superset of the maintenance branch and it contains new functional subsystems. Optional IRIX features, such as the Job Limits facility and Shift-JIS character-encoding support, are available only in the feature branch. This branch is available only to customers with active support contracts.

## IRIX 6.5 Has an Extensive and Solid Future

Occasionally someone asks, "When will SGI stop releasing IRIX 6.5 and release a new version of IRIX?" The short answer is, "When customers and third-party developers want SGI to stop pursuing any of the core goals of the IRIX 6.5 release model."

The IRIX 6.5 release model has allowed SGI to release new hardware and software functionality in a simple, reliable manner. It allows SGI customers and partners to use a single version of IRIX across all SGI systems with a common applications interface. It allows internal and external developers to use a rich set of evolving APIs, without fear that old interfaces will disappear or that new interfaces will cause incompatibilities. It ensures that customers, whether internal or external, need not worry that upgrading an installed version of IRIX will cause more problems than it fixes, nor debate whether an upgrade is desirable.

In short, IRIX 6.5 is a strong, reliable, and well-built foundation upon which SGI customers and developers can base their most demanding work and solve their most challenging technical and creative problems.