SunXTL 1.1 Architecture Guide

The Network Is the Computer™

Sun Microsystems Computer Company
2550 Garcia Avenue
Mountain View, CA 94043 USA
415 960-1300 fax 415 969-9131

Part No.: 801-7048-11
Revision A, December 1995

Contents

Preface

The SunXTL 1.1 Architecture Guide describes the architecture of the SunXTL™ Teleservices for Solaris™ platform. This book serves as an introduction to the services and resources provided by SunXTL to enable developers to write teleservices applications and technology providers. Note that all references to Solaris in this book apply only to the SPARC version of Solaris.

Who Should Use This Book

This book is intended for readers interested in Sun Microsystems™ software architecture for desktop teleservices.

How This Book is Organized

Chapter 1, "Introduction to SunXTL Teleservices," provides background for teleservices on the desktop, defines the SunXTL platform audience, and introduces the general features of SunXTL.

Chapter 2, "Overview of SunXTL Teleservices," presents an overview of the SunXTL platform, its components, and its features.

Chapter 3, "The SunXTL API," goes into further detail to describe the SunXTL classes provided by the SunXTL API and MPI programming libraries.

Chapter 4, "SunXTL Provider Developer's Kit," describes the task of writing providers for the SunXTL platform.

Related Documentation

The following documents are part of the SunXTL documentation set:

- *Sun XTL 1.1 Administrator's Guide*
- *Sun XTL 1.1 Application Programmer's Guide*
- *Sun XTL 1.1 Provider Programmer's Guide*
- *Sun XTL 1.1 Remote Client Mgr Guide*

What the Typographic Changes and Symbols Mean

The following table describes the type changes and symbols used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `system% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`system% su`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename*`.`
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.

Code samples are included in boxes and may display the following:

%	UNIX C shell prompt	`system%`
$	UNIX Bourne and Korn shell prompt	`system$`
#	Superuser prompt, all shells	`system#`

Introduction to SunXTL Teleservices 1

Computer workstations capable of connecting to wide area networks are becoming increasingly common. As the hardware and software components for supporting modems, ISDN, Asynchronous Transfer Mode (ATM) and analog telephony increasingly proliferate onto the computer desktop, new applications will become available to exploit them.

The impact of these new applications will likely be felt throughout all areas of computing—from commercial users in the largest corporations, to home users in small, remote communities. The widespread proliferation of the telephone, followed recently by the personal computer, will be joined in the near future by integration between them. As result, the demand for telephony-based applications in the computer-telephony integration (CTI) market will be enormous.

Developers of teleservices applications will confront a large range of communication implementation technologies, hardware and software considerations and other complicating factors. To reduce the effort required by developers, while simultaneously freeing them to utilize virtually any communication medium, Sun Microsystems has developed a teleservices product delivery vehicle known as the *SunXTL Teleservices Platform for Solaris.* The SunXTL platform includes support for teleservices applications, as well as software providers for specific device implementations. The SunXTL platform provides a stable and robust platform for teleservices applications on the client-server, distributed computing desktop.

1

Workstation Computer-Telephony Integration

CTI brings together the two most widely used desktop tools (telephones and workstations), enhancing the power and utility of each. Workstations are empowered with access to the wide area, public switched network, while telephony applications are enhanced by access to end users environment and data bases.

The key to enabling CTI on workstations is a good architecture and programming platform to allow applications to access telephone technology—independent of the particular topology, telephony interface, or type of phone system being used. Ideally, an application can be written once, and it should work with various wiring configurations and (within reason) with any telephony technology.

SunXTL Applications and Their Market

SunXTL Teleservices is particularly suited for creating personal desktop telephony services: automatic dialing, answering machine, voice mail, automatic call transfers, and so on. With appropriate interface cards, SunXTL Teleservices can turn a workstation into a personal fax machine. Another common usage would be to use SunXTL Teleservices to manage connections from the workstation to other network servers for Internet (IP) connectivity.

As network technologies transition to ISDN and ATM, SunXTL Teleservices provides an ideal platform for multimedia applications. Because SunXTL Teleservices provides access to a variety of network technologies and data types, it is well positioned for developing multimedia applications.

General Features of SunXTL Teleservices

SunXTL Teleservices for Solaris is the foundation library for applications that use or control telecom data streams. The SunXTL subsystem and API includes call control functions that establish a call or connection, and data stream access methods to control the flow of data over that connection.

On the workstation desktop SunXTL Teleservices can be used to control an ISDN line and manage phone calls for various voice applications like a personal answering machine with voice-mail/email integration, a user-friendly GUI for controlling the many features of a typical phone, or speed dialing

integrated with local databases. SunXTL Teleservices can also be used by non-voice applications like setting up desktop video conference calls, wide area datacomm, or fax.

SunXTL Teleservices is designed to allow multiple desktop applications to share access to workstation telephone interfaces and also to allow multiple applications to cooperate in managing an individual call without sacrificing data integrity or security. One application can create or receive a call and pass it to another application for voice processing. SunXTL Teleservices allows, but does not require, a single “call manager” application to own all calls and impose a user- or site-defined policy on the disposition of calls.

SunXTL Teleservices does not impose constraints on the content or format of the data stream; the stream is delivered to an application for interpretation. SunXTL Teleservices works equally well for audio / voice, FAX, video, IP, or other data communications protocols. SunXTL Teleservices is the control API for establishing connections on wide-area networks for future multimedia applications.

1

Overview of SunXTL Teleservices 2

The SunXTL Teleservices platform realizes a solid foundation for desktop telephone applications. Through object-oriented design, modular interfaces, and easy extensibility, the SunXTL platform provides standardized interfaces and services for software and hardware developers. This chapter describes the design goals of SunXTL Teleservices and how its architecture facilitates the development of desktop teleservices applications.

Goals of SunXTL Teleservices

The design and architecture of SunXTL Teleservices was driven by five goals:

- Provide an API for developing personal desktop applications. Applications such as an on-screen phone GUI, remote workstation access via DTMF, personal voice mail, etc. can share access to a workstation's telephony hardware.
- Provide transparent porting between analog, ISDN, ATM and other technologies. Applications should not need to be recompiled as network technology is upgraded.
- Provide easy access to common voice services. The building blocks of voice applications, like DTMF and silence detection must be available easily and efficiently, using "onboard" resources if available.
- Enable specialized or non-voice services. Many telephony boards also provide fax, modem, or video capabilities.

- Specify an open, third-party extensible, distributed-object model. The API and system capabilities must be dynamically extensible to utilize new technologies and new protocols as they become available.

The SunXTL system architecture achieves these goals by defining a clean separation between the application that uses telephony services and the provider of those services. The SunXTL system services provide object management, message passing and an event notification infrastructure that combine to bind an application to the appropriate teleservices provider.

Components of the SunXTL Subsystem

The SunXTL architecture consists of applications that use the SunXTL API, providers of SunXTL services, and the SunXTL libraries and system services; see Figure 2-1. The details of the system services are hidden from applications and providers by the API and Media Platform Interface (MPI) libraries. The system services include a message passing “server”, a data stream multiplexor streams driver, a provider configuration database, and a tool for administration of that database.

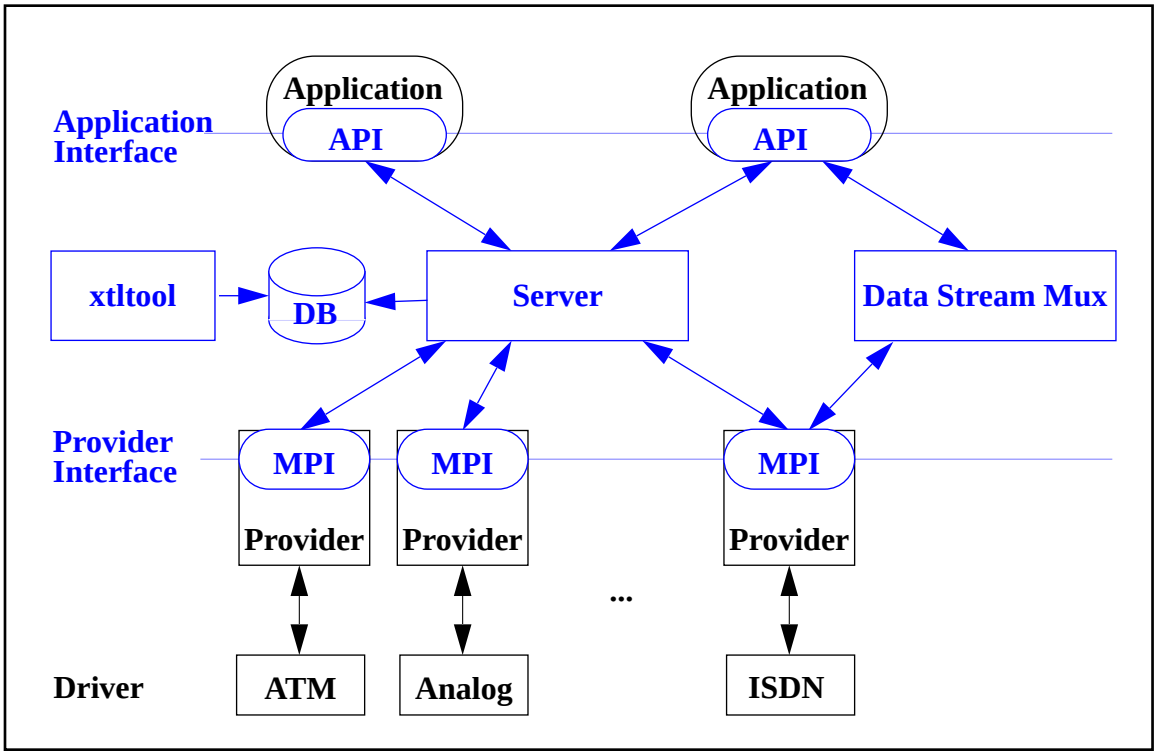

Figure 2-1 Components of the SunXTL Subsystem

Objects and Processes in the SunXTL Subsystem

Application developers use the SunXTL API to create C++ `XtlCall` objects. Each `XtlCall` object corresponds to a telephone call. An `XtlCall` object has command methods that query the current state of the call or request a change in state (for example, to put a call on hold). An `XtlCall` object also has callback methods for the asynchronous notification of state changes (for example, when a call is disconnected).

An `XtlCall` object includes accessor methods to determine the local and remote directory numbers associated with the call, the current call-progress state of the call, and the data type or media class associated with the call. Other

attributes can be associated with the `XtlCall` by either the provider or the controlling application. For example, an application may chose to annotate the call object with a customer name and account number.

The SunXTL API is primarily concerned with call control and the associated capabilities to provide security or shared management of calls when desired by the applications. A secondary aspect of the `XtlCall` object allows the application to gain access to the data stream associated with the call. The SunXTL API defines an interface for common usage of voice calls, like detecting DTMF tones or connecting a call to a convenient speaker/microphone. As other media types are utilized (like FAX or Video), the SunXTL API can be extended to provide access to the required features.

The SunXTL system defines the general semantics of a call object and the call establishment protocol, but the actual implementation of a call object depends on the particular telephony technology being used. For this reason, the bSunXTLsystem delegates the control of the call object to the `XtlProvider` object. The `XtlCall` objects follow the specifications and protocol of the SunXTL API. The provider executable is typically a UNIX process or driver supplied by the telephony hardware vendor. Because all SunXTL providers present the same interface to the application, they can be used interchangeably. The implementation characteristics (latency, bandwidth) may change, but an application can port to a new network Provider without recompilation.

The SunXTL system services act as the intermediary between the application view of a call object, and the provider's implementation of the call. The SunXTL server, along with the application and provider libraries, handles the interprocess message passing, object identification and creation, call ownership, security, and asynchronous event notification. The SunXTL subsystem also manages a database of available providers and helps manage data stream routing and access. A GUI tool, `xtltool(1)`, is supplied to browse and edit the provider configuration database.

Messaging

The SunXTL components communicate using asynchronous messages. In the vernacular of RPC, all messages are "one-way". Typically, when an application invokes a command method on a `XtlCall` object, a message is sent to the `XtlProvider` object. When the `XtlProvider` object detects a change of state in the call, it sends a message up to the application. When an application receives an event, the corresponding event method of the `XtlCall` object is

invoked. The event methods are C++ virtual functions that the application writer overrides to specify the appropriate processing. The API and provider libraries include routines to manage I/O signal handling, upcall and downcall dispatching, message marshalling, and the event wait loop.

Implementation Note – The SunXTL notification framework is based on the InterViews dispatcher library. In addition, the interprocess (over the wire) packets are generated by an asynchronous messaging protocol compiler based on an abstract definition of the message names and argument types. Once the message set is defined, the C++ code for marshalling and dispatch is generated automatically.

Call Ownership

The SunXTL system insures that only one application at a time is allowed to control a call. The controlling application is referred to as the “owner” of the call. When an application makes an outgoing call, that application becomes the owner. An incoming call is initially “owned” by the provider, but the provider immediately “offers” that call to any and all interested applications. An application gains control of an incoming call by issuing a “claim” request. If the claim is successful, the application becomes the owner and the application’s Call object is instantiated. The SunXTL system ensures that only the first claim request is honored for any offer (i.e. “first come, first served”).

Applications can share management of calls by passing ownership. The original owner “offers” the call, and another application can “claim” it. The same rules and protocol are used for application-to-application transfer of ownership as for claiming an incoming call.

Note – Claiming an incoming call is separate from answering the call. Once claimed, the unanswered call can be answered, dropped, or redirected. Or, the call may be annotated, and then offered to another application.

Security

The SunXTL system was designed to operate on networked personal workstations, and in that environment, security and authentication of users is critical. The SunXTL system uses the standard UNIX and UNIX File System

security features, making it easy to configure the system to restrict telephone usage to the authorized owner of the workstation and telephone. The SunXTL system can also be configured as a network resource, allowing multiple users to share access to telephony resources.

Several layers of security are provided, allowing the user or system administrator to tailor the system to the specific security needs of the site. The first layer of security is access to the SunXTL server. This is controlled by the UFS permissions on the named pipe used to make contact with the server. By default, only root applications can use the SunXTL server. A simple script is provided to allow access to particular UNIX group; typically that group would include the workstation owner/user, or a larger group if the workstation is configured as a departmental server. The server limits access to each provider based on an access control list maintained per provider. Therefore, individual line interfaces can be reserved for specific users. When used as a network server, security is provided by the usual ONC/RPC mechanisms.

Media Channel Access

For each `XtlCall` object there is a corresponding media channel that contains a call's data stream. The capabilities and utility of the media channel vary depending on the provider and the type of call. Some providers may perform only switching of calls, and not allow application access to the media channel. Other providers may have DSP capability and provide numerous services for voice/analog calls (e.g. DTMF tones, modem, ASR). Typically, the media channel provides a bidirectional UNIX stream that the application can read and write.

Extensibility

The SunXTL provider interface is open for third-party developers and Independent Hardware Vendors (IHVs). Using the SunXTL provider library ensures compliance with the basic system protocols. However, the SunXTL message set can be extended to allow access to provider-specific, technology-specific, or call-type-specific features.

Every API call and every message includes an optional argument that is a list of arbitrary key-value pairs (an `XtlKVlist`). The contents and use of the `XtlKVlist` is determined by the provider, and documented by the provider

for the benefit of the application developer. Portable applications may not use the `XtlKVlist`, but applications that use specific provider features can access them within the SunXTL framework using the `XtlKVlist` extensions.

The SunXTL API and message set also include a universal extension message, independent of any SunXTL protocol semantics. By specializing this message and its `XtlKVlist` argument, *any* provider-specific features can be accessed.

To avoid a proliferation of provider-specific enhancements and to increase the portability of applications, SunXTL Teleservices will be periodically updated to include standards for common features as they become known and used.

Provider Configuration Database

When providers are installed on a system, they are registered in a provider configuration database. The database shows the existence of each provider, how to invoke it (used by the SunXTL system internally) and describes the specific capabilities and features of that provider. For example, the database would indicate the number of simultaneous lines available, the bandwidth available for datacomm, the types of voice or DSP processing available, etc. If an application has specific requirements, it can query the database to determine the most appropriate provider for its needs.

2

The SunXTL API 3

The SunXTL API is an object-oriented interface accessed using the C++ language. The API includes `XtlProvider`, `XtlCall`, `XtlCallState`, and `XtlMonitor` objects. These base objects define the command and callback methods of SunXTL Teleservices. There are three types of methods on these objects: synchronous requests, asynchronous commands, and asynchronous events. Synchronous requests modify or query information cached in the local C++ object. Command methods send messages to the provider. Results or status changes are reported back to the application via the event methods.

An `XtlProvider` object is the application's representation of an SunXTL provider. The application registers with the provider object to receive notification of incoming calls, and specifies the provider when creating outgoing calls.

`XtlCallState` objects represent the identity and a snapshot of the state of a call. When a provider has an incoming call, the registered applications receive a `XtlCallState` object corresponding to that call. The `XtlCallState` object is not "active"; it does not have command or event methods. If an application wants to monitor or control a call, it creates a `XtlMonitor` or `XtlCall` object based on the `XtlCallState`. In addition to incoming call notifications, an application can get a list of `XtlCallState` objects corresponding to all the calls currently active on a provider.

Monitor objects allow an application to track the status of a call (for logging, statistics, billing, etc.) without actually controlling the call. The `XtlMonitor` object has notification methods that track the state changes of the call, but there are no command methods to control the call.

3

An `XtlCall` object represents a call that is “owned” by the application, with complete command repertoire and event/status notification. In addition to controlling call progress, a `XtlCall` object enables an application to configure or get direct access to the data stream. Call objects and call ownership can be passed between cooperating applications.

The SunXTL API also allows the application to manage the data stream for a call. This can be as simple as providing a UNIX stream for the data being sent and received, or can be a request to connect the call to various hardware or software resources to, for example, detect DTMF tones. The particular data stream capabilities available depend on the provider and the type of data being transmitted by the call. All data stream configuration requests are handled by the provider, allowing it to do data routing and signal processing as efficiently as it can. For example, the provider may use onboard DSPs, special data busses, or other directly connected hardware resources without the application knowing the specifics.

Distributed Object Activation

When an application creates an `XtlCall` object, a C++ object is allocated and returned to the application. The actual state and implementation of an `XtlCall` object is maintained in the provider. Until the provider has created its portion of the object, the command methods of the `XtlCall` object in the application cannot be meaningfully used.

As part of object creation in the API, a creation request message is sent to the `XtlProvider` object. When the `XtlProvider` object has completed its object creation and initialization, the application is notified via the `activation_ind()` method of the `XtlCall` object. The application’s implementation of the activation method typically begins processing of the call.

If the remote aspect of an object should become unreachable (e.g. network or server failure) the application is notified that the object is no longer accessible via the `deactivation_ind()`. The application’s implementation of the deactivation method typically cleans up references and deletes the C++ object.

Select SunXTL Objects and Methods

This section shows a few of the important methods of the `XtlProvider` and `XtlCall` objects. For complete information on the SunXTL API and the SunXTL MPI, see the *Sun XTL 1.1 Application Programmer's Guide* and the *Sun XTL 1.1 Provider Programmer's Guide* respectively.

Methods of the Provider Object

Command Methods:

```
list_calls_req();
enable_offer_event_req(boolean_t on_off, KVList& args);
listen_req(CallEvent listenFor, KVList& args)
```

Notification Methods:

```
list_calls_ind(CallState *const * call_list, u_int call_count);
call_event_ind(
    CallState& call, CallEvent event, Cause cause, KVList& args);
offer_ind(CallState& offeredCall, KVList& args);
```

Methods of the Call Object

Command Methods:

```
connect_req(Address& local, Address& remote, Format& media_format,
           KVList& args);
answer_req(KVList& args);
disconnect_req(KVList& args);
hold_req(KVList& args);
unhold_req(KVList& args);
transfer_req(CallState& transfer_to, KVList& args);
redirect_req(Address& redirect_to, KVList& args);
conference_req(CallState& conferenc_in, KVList& args);
drop_req(KVList& args);
offer_req(KVList& args);
configuration_req(KVList& requested_configuration);
application_info_req(KVList& info);
```

Notification Methods:

```
event_ind(CallEvent event, Cause cause, KVList& args);
```

`CallEvent` is an enum of possible state changes:

```
PROCEEDING, ALERTING, CONNECT, FAILURE, DISCONNECT, INFO, TRANSFER,
REDIRECT, CONFERENCE, DROP
```

SunXTL Provider Developer's Kit

4

SunXTL Teleservices for Solaris provides an Internet Protocol (IP) provider and a provider simulator as part of its distribution. Other network or telephony technologies may easily be integrated into SunXTL Teleservices. Developers of teleservices hardware devices to be integrated into Solaris-based systems typically supply two components: a device driver and provider; see Figure 4-1 on page 17. The device driver handles low-level details of the underlying network transfer scheme or telephony device (e.g. ISDN, ATM, FDDI, analog telephony, etc.), and is written to conform to UNIX System V Release 4 standards. The provider is the code necessary to manage the interface between the SunXTL subsystem and the associated device-specific driver.

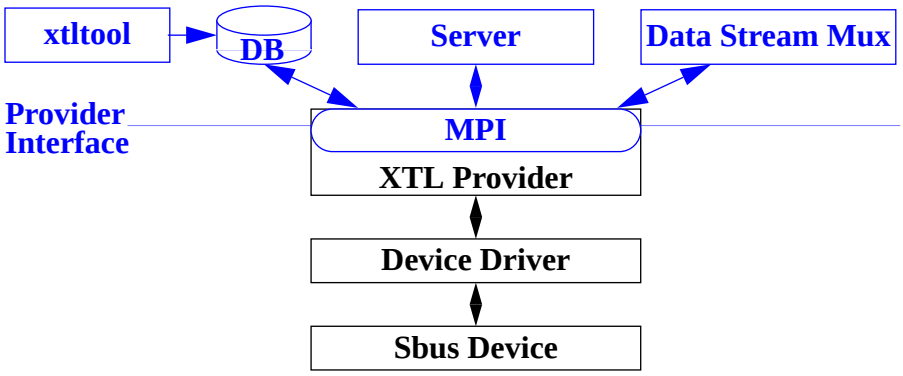

Figure 4-1 SunXTL Provider and Device Driver

4

Providing Access to Additional Teleservices Technologies

The task of writing an SunXTL provider is simplified by using the SunXTL Media Platform Interface (MPI) library. The provider process uses the MPI to handle all communication and messaging to the SunXTL system components. The provider developer's sole concern is to translate between the SunXTL protocol and whatever interface is supplied by the device driver.

When devices have additional features not defined by the SunXTL API/MPI, the provider extension messages can be used to define the mechanism to allow an application to access the device's features. This extension mechanism is described in detail in the *Sun XTL 1.1 Provider Programmer's Guide*.

SunXTL Provider Library

The SunXTL provider library (MPI) isolates the provider code from the intricacies of the SunXTL system services. Specifically the library supplies interfaces to the provider database, the server messaging system for commands and callbacks, and the mechanisms for making device data streams accessible to applications. For complete information, see the *Sun XTL 1.1 Provider Programmer's Guide*.

Index

Reader Comments

We welcome your comments and suggestions to help improve this manual. Please let us know what you think about the *SunXTL 1.1 Architecture Guide*, part number *801-7048-11*.

- The procedures were well documented.

Strongly Agree	Agree	Disagree	Strongly Disagree	Not Applicable
❑	❑	❑	❑	❑

Comments _____

- The tasks were easy to follow.

Strongly Agree	Agree	Disagree	Strongly Disagree	Not Applicable
❑	❑	❑	❑	❑

Comments _____

- The illustrations were clear.

Strongly Agree	Agree	Disagree	Strongly Disagree	Not Applicable
❑	❑	❑	❑	❑

Comments _____

- The information was complete and easy to find.

Strongly Agree	Agree	Disagree	Strongly Disagree	Not Applicable
❑	❑	❑	❑	❑

Comments _____

- Do you have additional comments about the *SunXTL 1.1 Architecture Guide*?

_____

_____

_____

_____

Name: _____

Title: _____

Company: _____

Address: _____

_____

Telephone: _____

Email address: _____

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 1 MOUNTAIN VIEW, CA

POSTAGE WILL BE PAID BY ADDRESSEE

SUN MICROSYSTEMS, INC.
Attn: Manager, Publications
MS MPK 14-101
2550 Garcia Avenue
Mt. View, CA 94043-9850