Hypervisor Service API

Revision 0.45

March 23, 2005

Table of Contents

Revision	Author	Comment
0 1 – 0 42	David Redman	Created, undated for FWARC submission
0 42	David Redman David Kahn	Include State transition diagram from David Kahn
0 43	David Redman	Added the OBP device node information
0 44	David Redman David Kahn	Merged edits from David Kahn, added revision table
0,45	David Kahn	Comments from Hitendra and Sunit (FWARC approved version)

Revision Information

1 Introduction

This document describes a set of APIs that describe the data link layer used to connect two data bearing endpoints, typically a sun4v guest and a service processor. Example services are nvram, error and fma transports. The content of the connection (packet format) will be documented by the producer/consumers.

Services are identified and configured by nodes in the Machine Description.

Services that support *send* and *recv* are full-duplex. A service does not have to support both *send* and *recv*.

The transport is reliable, and only a single transaction is permitted to be outstanding per endpoint per direction. A service will not "deny service" to another service due to resource constraints, a packet is either delivered reliably or it is not delivered at all. The hypervisor implementation ensures that data integrity is maintained between the endpoints.

The transport is connection-less; the only delivery failure mechanism is the abnormal termination of the remote end of the connection, this would result in an abort (ABRT) condition. Upper software layers need to be able to recover from this if they have state outside the content of the data being transported.

These APIs are used in certain virtual device nodes in OBP and by Operating system device drivers to transport data in a virtualized fashion between a guest and the service processor.

2 Common definitions

2.1 Function numbers for FAST_TRAP

Function numbers for fast-traps are provided in %o5 as a 64-bit value. The following are the function numbers defined for the service API set:

```
SVC_SEND                0x80
SVC_RECV                0x81
SVC_GETSTATUS           0x82
SVC_SETSTATUS           0x83
SVC_CLRSTATUS           0x84
```

2.2 Data Definitions and Acronyms

SID — A unique identifier for each service, SID Value 0 is reserved, the values are opaque
Note: The SID value is defined by a property in the machine description as described in section 4.1.1
Implementation note: SID is restricted to 16 bits.

MTU — The largest number of bytes that may be transported using a single call. Typically this should be the *length* argument for an SVC_RECV call,.
Note: MTU is defined by a property in the machine description as described in section 4.1.1.

2.3 Service Status Register Definitions

This is the 64bit service status register used by the GET/SET/CLR Status APIs.

Bit	State	Symbol	Meaning
0	R/W1C	RX	'RECV' data available
1	R/W	RXE	RECV interrupt enabled
2	R/W1C	TX	SEND complete
3	R/W	TXE	SEND interrupt enabled
4	R/O	TB	Transmitter Busy
54 14	R/O		Reserved, 0
15	R/W1C	ABRT	Channel Error
16 63	R/O		Reserved, 0

R/W1C — Readable, Write 1 Clear this bit may only be cleared by CLRSTATUS.

R/W — Readable, Writeable, SETSTATUS and CLRSTATUS can modify this bit.

R/O — Read Only, SETSTATUS and CLRSTATUS do not modify this bit.

RX — Receive data available; This bit has side effects, when set and RXE is 1, an interrupt will be generated (for services that support interrupts). The RX bit must be cleared in order to clear the interrupt (CLRSTATUS).

TX — Transmit completed; This bit has side effects, when set and TXE is 1, an interrupt will be generated (for services that support interrupts). The TX bit must be cleared in order to clear the interrupt (CLRSTATUS).

3 API Call Descriptions

Registers %o0-%o5 are volatile across all service APIs unless specifically indicated otherwise.

On return, %o0 will contain the hypercall status as defined by FWARC/2005/116 (Core API).

3.1 Data Send and Receive

3.1.1 svc_send

```
trap#           FAST_TRAP
function#       SVC_SEND
arg0            SID
arg1            buffer
arg2            length

ret0            status
```

Send the content of *buffer* into the outgoing queue for delivery, the buffer may not be released or reused until the send is complete, indicated by a transmit done interrupt or the setting of the TX bit in the service status register (SVC_GETSTATUS). It is not necessary to clear the TX bit in order to send another packet.

3.1.1.1 Errors

```
EINVAL          length is larger than MTU or SID is invalid
ENORADDR        bad buffer address
EWOULDBLOCK     a packet is already queued for delivery (TB=1)
```

3.1.2 svc_recv

```
trap#           FAST_TRAP
function#       SVC_RECV
arg0            SID
arg1            buffer
arg2            length

ret0            status
ret1            actual-length
```

Copy a maximum *length* bytes of available data into *buffer*. If *length* is larger than the number of bytes available then *ret1* will return the actual number of bytes. SVC_RECV may be called many times, the same data will be returned and the service will remain 'BUSY' until the RX bit is cleared in the service status register (SVC_GETSTATUS). Typically *length* should be the MTU size associated with this SID.

3.1.2.1 Errors

```
EINVAL          length is larger than MTU or SID is invalid
ENORADDR        bad buffer address
EWOULDBLOCK     no data is available (RX=0)
```

3.2 Status APIs

The status APIs return information about the endpoints.

SVC_RECV will succeed and return data, if RX is set.

SVC_SEND will return EWOULDBLOCK, if TB is set.

if TX is set then the transmission is complete, and the SVC_SEND buffers may be reused.

3.2.1 svc_getstatus

```
trap#           FAST_TRAP
function#       SVC_GETSTATUS
arg0            SID

ret0            status
ret1            service status register
```

Return the service control register for service *SID.*

3.2.1.1 Errors

```
EINVAL          SID is invalid
```

3.2.2 svc_setstatus

```
trap#           FAST_TRAP
function#       SVC_SETSTATUS
arg0            SID
ret1            service status register

ret0            status
```

Set bits marked R/W in the service control register for service *SID,* this is a write-1-set operation, read-only bits are unchanged, reserved bits remain 0.

3.2.2.1 Errors

```
EINVAL          SID is invalid
```

3.2.3 svc_clrstatus

```
trap#           FAST_TRAP
function#       SVC_CLRSTATUS
arg0            SID
ret1            service status register

ret0            status
```

Clear bits marked R/W or W1C in the service control register for service SID, this is a write-1-clear operation, read-only bits are unchanged, reserved bits remain 0.

3.2.3.1 Errors

```
EINVAL          SID is invalid
```

4 Machine Description

Services are described and configured by nodes in the machine description.

4.1 Property Definitions

4.1.1 Platform_service node

Property	Type	Optional	Meaning/Purpose
name	PROP_STR	no	A human readable name to describe this service (nvram, fma etc)
flags	PROP_VAL	no	Configuration information about this service
ino	PROP_VAL	yes	Virtual INO generated, required if bit1 or 3 is set in flags
mtu	PROP_VAL	no	Maximum data payload in bytes for this service
sid	PROP_VAL	no	SID to identify this service
compatible	PROP_DATA	yes	An array of strings used by the operating system to bind devices drivers to this channel The property is only required for nodes that Solaris would bind drivers to

4.1.1.1 Flags Definitions

Bit	Meaning
0	SVC'_RECV available
1	RECV interrupt capable
2	SVC_SEND available
3	SEND interrupt capable

The purpose of the flags property is to permit lower resource utilization for services that are unidirectional or do not generate interrupts.

5 Virtual device bindings

This section describes the device bindings to Open Firmware for the sun4v devices using the SVC APIs, the property values are all extracted from the Machine Description, and nodes are created under the /virtual-devices node for each platform_service found in the Machine Description.

Note: Since these nodes are children of the 'virtual-devices' node, the reg property format is defined by the 'virtual-devices' binding, which is specified by FWARC/2005/111.

5.1 Open Firmware device node representation

There are no OBP drivers for most of these nodes, so no **device_type** property or methods are required for most nodes.

5.1.1 Open Firmware Defined Properties

"**reg**" standard propname defining the devices address space

Value: A single configuration space entry.

"name" standard propname defining the device name

Value: An encoded string, from the Machine Description property of the same name.

"compatible" standard propname defining driver compatibility

Value: An encoded string, from the Machine Description property of the same name.

Note: This property is optional.

"interrupts" standard propname defining the interrupts

Value: A single encoded int with the value 1

This property is only required for nodes that generate interrupts.

"sid" propname defining the service ID to use for this transport.

Value: A single encoded int, from the Machine Description property of the same name.

"mtu" propname defining the maximum payload in bytes for this transport.

Value: A single encoded int, from the Machine Description property of the same name.

"flags" propname defining the capabilities of this transport.

Value: A single encoded int, from the Machine Description property of the same name.

5.2 Virtual Device Nodes

This section defines the 'Great Lakes' platform-specific implementation of device nodes that use the Services API defined by this document.

The nodes all have the same set of properties as defined in section 4.1.1, For simplicity the properties and their expected values are placed in table forms in the next subsections. Unless stated otherwise these nodes have no device methods and no “device_type” property.

5.2.1 FMA connector

This service provides the operating system with the translated e-reports from the service processor and connects the OS fma engine to components on the service processor to manage fault status.

Property	Value
name	fma
flags	0xf
mtu	504
interrupts	1
Sid	(a non-zero integer value)

5.2.2 Explorer connector

This service provides the operating system with a transport to the service processor used to extract FRU and environment status from the service processor.

Property	Value
name	explorer
flags	0xf
mtu	128
interrupts	1
sid	(a non-zero integer value)

5.2.3 LED connector

This service provides the operating system with a mechanism to control the platform dependant LEDs (light emitting diodes) present in the systemchassis.

Property	Value
name	led
flags	0xf
mtu	128
interrupts	1
sid	(a non-zero integer value)

5.2.4 NVRAM

This is the virtual device transport used by OpenBoot to retain environment variables in a binary format, the content of the virtual device is private to OpenBoot (the same as it has always been. An OpenFirmware client uses the **setprop** interface on properties in the **options** node to set them.

Property	Value
name	nvram
flags	5
mtu	64
sid	(a non-zero integer value)
device_type	nvram

5.2.4.1 NVRAM Methods

These are documented for information purposes only; these are the 'standard' methods used by OBP4.X to abstract the interface between OpenBoot and the hardware implementation of the backing store.

open (-- true|false) Standard method

close (--) Standard method

read (buf, len –actual) Standard method

write (buf, len – actual) Standard method

size (-- rem) Method to return the number of bytes remaining in the device.

seek (offset type – error?) offset is a byte offset in the device

If type is 0, offset is relative to the beginning of the device. Offset must be zero or a positive value.

If type is 1, offset is relative to the current position, and offset may be zero or a positive or negative value.

All other types result in an error return.

sync (--) Flush any outstanding state to the backing store.

These methods will use the service APIs to transport blocks of data to the SP.

6 Transmitter/Receiver State transitions