Linux Standard Base Trial Use Specification 4.0

Linux Standard Base Trial Use Specification 4.0
LSB Trial Use Specification

Contents

List of Tables

Foreword

This is version 4.0 of the Trial Use Specification. This version is a preliminary version for review only. Conclusion of work on this version will result in version 3.2 of the Trial Use Specification.

This specification describes components which have Trial Use Specification status, and as such there is no formal compliance process for this specification. Implementations may claim to provide these components in a manner that agrees with this specification, but such a claim is not part of a conformance statement for the LSB version in which this module appears.

Applications may not assume that the components of this specification are present or operate as described in this specification on any given implementation.

Introduction

The Trial Use Specification describes components which may or may not be present on an otherwise conforming system. The purpose is to indicate that these components are on a Standards Track, that is, they are intended to become part of the LSB Specification in a future edition.

This document should be used in conjunction with the documents it references. Information referenced in this way is as much a part of this document as is the information explicitly included here.

I Introductory Elements

1 Scope

The Trial Use Specification defines components which are not required parts of the LSB Specification.

2 Normative References

The specifications listed below are referenced in whole or in part by the Trial Use Specification. Such references may be normative or informative; a reference to specification shall only be considered normative if it is explicitly cited as such. The Trial Use Specification may make normative references to a portion of these specifications (that is, to define a specific function or group of functions); in such cases, only the explicitly referenced portion of the specification is to be considered normative.

Table 2-1 Informative References

Name	Title	URL
ALSA Library API Reference	ALSA Library API Reference	http://www.alsa-project.org/alsa-doc/alsa-lib/
ISO C (1999)	ISO/IEC 9899: 1999, Programming Languages --C	
Java application launcher documentation	Java application launcher documentation	http://java.sun.com/javase/6/docs/technotes/tools/solaris/java.html
Java Platform SE 6 API	Java Platform, Standard Edition 6 API Specification	http://java.sun.com/javase/6/docs/api/index.html
Java VM Specification	Java Virtual Machine Specification	http://java.sun.com/docs/books/jvms/second_edition/html/VMSpecTOC.doc.html
Mozilla's NSS SSL Reference	Mozilla's NSS SSL Reference	http://www.mozilla.org/projects/security/pki/nss/ref/ssl/
NSPR Reference	Mozilla's NSPR Reference	http://refspecs.linuxfoundation.org/NSPR_API_Reference/NSPR_API.html
xdg-utils reference	Portland Project XDG Utilities Reference 1.0	http://portland.freedesktop.org/xdg-utils-1.0/

3 Requirements

3.1 Multimedia Libraries

The multimedia libraries listed in Table 3-1 shall be available on a Linux Standard Base system, with the specified runtime names. This list may be supplemented or amended by the architecture-specific specification.

Table 3-1 Standard Library Names

Library	Runtime Name
libasound	libasound.so.2

These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

3.2 Security Libraries

The security libraries listed in Table 3-2 shall be available on a Linux Standard Base system, with the specified runtime names. This list may be supplemented or amended by the architecture-specific specification.

Table 3-2 Standard Library Names

Library	Runtime Name
libnss3	libnss3.so
libssl3	libssl3.so
libnspr4	libnspr4.so

These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

4 Definitions

For the purposes of this document, the following definitions, as specified in the *ISO/IEC Directives, Part 2, 2001, 4th Edition*, apply:

can

be able to; there is a possibility of; it is possible to

cannot

be unable to; there is no possibility of; it is not possible to

may

is permitted; is allowed; is permissible

need not

it is not required that; no...is required

shall

is to; is required to; it is required that; has to; only...is permitted; it is necessary

shall not

is not allowed [permitted] [acceptable] [permissible]; is required to be not; is required that...be not; is not to be

should

it is recommended that; ought to

should not

it is not recommended that; ought not to

5 Terminology

For the purposes of this document, the following terms apply:

implementation-defined

Describes a value or behavior that is not defined by this document but is selected by an implementor. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations. The implementor shall document such a value or behavior so that it can be used correctly by an application.

Shell Script

A file that is read by an interpreter (e.g., awk). The first line of the shell script includes a reference to its interpreter binary.

undefined

Describes the nature of a value or behavior not defined by this document which results from use of an invalid program construct or invalid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

unspecified

Describes the nature of a value or behavior not specified by this document which results from use of a valid program construct or valid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

II ALSA sound library

6 Libraries

6.1 Interfaces for libasound

Table 6-1 defines the library name and shared object name for the libasound library

Table 6-1 libasound Definition

Library:	libasound
SONAME:	libasound.so.2

The behavior of the interfaces in this library is specified by the following specifications:
[ALSA] ALSA Library API Reference

6.1.1 ALSA Configuration Interface

6.1.1.1 Interfaces for ALSA Configuration Interface

An LSB conforming implementation shall provide the generic functions for ALSA Configuration Interface specified in Table 6-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-2 libasound - ALSA Configuration Interface Function Interfaces

snd_config_add(ALSA_0.9) [ALSA]	snd_config_copy(ALSA_0.9) [ALSA]	snd_config_delete(ALSA_0.9) [ALSA]
snd_config_get_ascii(ALSA_0.9) [ALSA]	snd_config_get_id(ALSA_0.9) [ALSA]	snd_config_get_integer(ALSA_0.9) [ALSA]
snd_config_get_integer64(ALSA_0.9) [ALSA]	snd_config_get_string(ALSA_0.9) [ALSA]	snd_config_get_type(ALSA_0.9) [ALSA]
snd_config_imake_integer(ALSA_0.9) [ALSA]	snd_config_imake_integer64(ALSA_0.9) [ALSA]	snd_config_imake_string(ALSA_0.9) [ALSA]
snd_config_iterator_end(ALSA_0.9) [ALSA]	snd_config_iterator_entry(ALSA_0.9) [ALSA]	snd_config_iterator_first(ALSA_0.9) [ALSA]
snd_config_iterator_next(ALSA_0.9) [ALSA]	snd_config_load(ALSA_0.9) [ALSA]	snd_config_make_compound(ALSA_0.9) [ALSA]
snd_config_make_integer(ALSA_0.9) [ALSA]	snd_config_make_integer64(ALSA_0.9) [ALSA]	snd_config_make_string(ALSA_0.9) [ALSA]
snd_config_save(ALSA_0.9) [ALSA]	snd_config_search(ALSA_0.9) [ALSA]	snd_config_searchv(ALSA_0.9) [ALSA]
snd_config_set_ascii(ALSA_0.9) [ALSA]	snd_config_set_integer(ALSA_0.9) [ALSA]	snd_config_set_integer64(ALSA_0.9) [ALSA]
snd_config_set_string(ALSA_0.9) [ALSA]	snd_config_top(ALSA_0.9) [ALSA]	snd_config_update(ALSA_0.9) [ALSA]
snd_config_update_free_global(ALSA_0.9)		

[ALSA]		

An LSB conforming implementation shall provide the generic data interfaces for ALSA Configuration Interface specified in Table 6-3, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-3 libasound - ALSA Configuration Interface Data Interfaces

snd_config(ALSA_0.9) [ALSA]		

6.1.2 ALSA Control Interface

6.1.2.1 Interfaces for ALSA Control Interface

An LSB conforming implementation shall provide the generic functions for ALSA Control Interface specified in Table 6-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-4 libasound - ALSA Control Interface Function Interfaces

snd_async_add_ctl_handler(ALSA_0.9) [ALSA]	snd_async_handler_get_ctl(ALSA_0.9) [ALSA]	snd_card_get_index(ALSA_0.9) [ALSA]
snd_card_get_longname(ALSA_0.9) [ALSA]	snd_card_get_name(ALSA_0.9) [ALSA]	snd_card_load(ALSA_0.9) [ALSA]
snd_card_next(ALSA_0.9) [ALSA]	snd_ctl_card_info(ALSA_0.9) [ALSA]	snd_ctl_card_info_clear(ALSA_0.9) [ALSA]
snd_ctl_card_info_copy(ALSA_0.9) [ALSA]	snd_ctl_card_info_free(ALSA_0.9) [ALSA]	snd_ctl_card_info_get_components(ALSA_0.9) [ALSA]
snd_ctl_card_info_get_driver(ALSA_0.9) [ALSA]	snd_ctl_card_info_get_id(ALSA_0.9) [ALSA]	snd_ctl_card_info_get_longname(ALSA_0.9) [ALSA]
snd_ctl_card_info_get_mixername(ALSA_0.9) [ALSA]	snd_ctl_card_info_get_name(ALSA_0.9) [ALSA]	snd_ctl_card_info_malloc(ALSA_0.9) [ALSA]
snd_ctl_card_info_sizeof(ALSA_0.9) [ALSA]	snd_ctl_close(ALSA_0.9) [ALSA]	snd_ctl_elem_add_boolean(ALSA_0.9) [ALSA]
snd_ctl_elem_add_iec958(ALSA_0.9) [ALSA]	snd_ctl_elem_add_integer(ALSA_0.9) [ALSA]	snd_ctl_elem_id_clear(ALSA_0.9) [ALSA]
snd_ctl_elem_id_copy(ALSA_0.9) [ALSA]	snd_ctl_elem_id_free(ALSA_0.9) [ALSA]	snd_ctl_elem_id_get_device(ALSA_0.9) [ALSA]
snd_ctl_elem_id_get_index(ALSA_0.9) [ALSA]	snd_ctl_elem_id_get_interface(ALSA_0.9) [ALSA]	snd_ctl_elem_id_get_name(ALSA_0.9) [ALSA]
snd_ctl_elem_id_get_numid(ALSA_0.9) [ALSA]	snd_ctl_elem_id_get_subdevice(ALSA_0.9) [ALSA]	snd_ctl_elem_id_malloc(ALSA_0.9) [ALSA]
snd_ctl_elem_id_set_de	snd_ctl_elem_id_set_in	snd_ctl_elem_id_set_in

vice(ALSA_0.9) [ALSA]	dex(ALSA_0.9) [ALSA]	terface(ALSA_0.9) [ALSA]
snd_ctl_elem_id_set_name(ALSA_0.9) [ALSA]	snd_ctl_elem_id_set_numid(ALSA_0.9) [ALSA]	snd_ctl_elem_id_set_subdevice(ALSA_0.9) [ALSA]
snd_ctl_elem_id_sizeof(ALSA_0.9) [ALSA]	snd_ctl_elem_iface_name(ALSA_0.9) [ALSA]	snd_ctl_elem_info(ALSA_0.9) [ALSA]
snd_ctl_elem_info_clear(ALSA_0.9) [ALSA]	snd_ctl_elem_info_copy(ALSA_0.9) [ALSA]	snd_ctl_elem_info_free(ALSA_0.9) [ALSA]
snd_ctl_elem_info_get_count(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_id(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_item_name(ALSA_0.9) [ALSA]
snd_ctl_elem_info_get_items(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_max(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_max64(ALSA_0.9) [ALSA]
snd_ctl_elem_info_get_min(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_min64(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_name(ALSA_0.9) [ALSA]
snd_ctl_elem_info_get_numid(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_step(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_step64(ALSA_0.9) [ALSA]
snd_ctl_elem_info_get_type(ALSA_0.9) [ALSA]	snd_ctl_elem_info_is_inactive(ALSA_0.9) [ALSA]	snd_ctl_elem_info_is_locked(ALSA_0.9) [ALSA]
snd_ctl_elem_info_is_readable(ALSA_0.9) [ALSA]	snd_ctl_elem_info_is_user(ALSA_0.9) [ALSA]	snd_ctl_elem_info_is_volatile(ALSA_0.9) [ALSA]
snd_ctl_elem_info_is_writable(ALSA_0.9) [ALSA]	snd_ctl_elem_info_malloc(ALSA_0.9) [ALSA]	snd_ctl_elem_info_set_id(ALSA_0.9) [ALSA]
snd_ctl_elem_info_set_item(ALSA_0.9) [ALSA]	snd_ctl_elem_info_sizeof(ALSA_0.9) [ALSA]	snd_ctl_elem_list(ALSA_0.9) [ALSA]
snd_ctl_elem_list_alloc_space(ALSA_0.9) [ALSA]	snd_ctl_elem_list_clear(ALSA_0.9) [ALSA]	snd_ctl_elem_list_copy(ALSA_0.9) [ALSA]
snd_ctl_elem_list_free(ALSA_0.9) [ALSA]	snd_ctl_elem_list_free_space(ALSA_0.9) [ALSA]	snd_ctl_elem_list_get_count(ALSA_0.9) [ALSA]
snd_ctl_elem_list_get_id(ALSA_0.9) [ALSA]	snd_ctl_elem_list_get_name(ALSA_0.9) [ALSA]	snd_ctl_elem_list_get_used(ALSA_0.9) [ALSA]
snd_ctl_elem_list_malloc(ALSA_0.9) [ALSA]	snd_ctl_elem_list_set_offset(ALSA_0.9) [ALSA]	snd_ctl_elem_list_sizeof(ALSA_0.9) [ALSA]
snd_ctl_elem_read(ALSA_0.9) [ALSA]	snd_ctl_elem_remove(ALSA_0.9) [ALSA]	snd_ctl_elem_type_name(ALSA_0.9) [ALSA]

snd_ctl_elem_value_clear(ALSA_0.9) [ALSA]	snd_ctl_elem_value_copy(ALSA_0.9) [ALSA]	snd_ctl_elem_value_free(ALSA_0.9) [ALSA]
snd_ctl_elem_value_get_boolean(ALSA_0.9) [ALSA]	snd_ctl_elem_value_get_byte(ALSA_0.9) [ALSA]	snd_ctl_elem_value_get_bytes(ALSA_0.9) [ALSA]
snd_ctl_elem_value_get_enumerated(ALSA_0.9) [ALSA]	snd_ctl_elem_value_get_id(ALSA_0.9) [ALSA]	snd_ctl_elem_value_get_iec958(ALSA_0.9) [ALSA]
snd_ctl_elem_value_get_integer(ALSA_0.9) [ALSA]	snd_ctl_elem_value_get_integer64(ALSA_0.9) [ALSA]	snd_ctl_elem_value_malloc(ALSA_0.9) [ALSA]
snd_ctl_elem_value_set_boolean(ALSA_0.9) [ALSA]	snd_ctl_elem_value_set_byte(ALSA_0.9) [ALSA]	snd_ctl_elem_value_set_enumerated(ALSA_0.9) [ALSA]
snd_ctl_elem_value_set_id(ALSA_0.9) [ALSA]	snd_ctl_elem_value_set_iec958(ALSA_0.9) [ALSA]	snd_ctl_elem_value_set_integer(ALSA_0.9) [ALSA]
snd_ctl_elem_value_set_integer64(ALSA_0.9) [ALSA]	snd_ctl_elem_value_sizeof(ALSA_0.9) [ALSA]	snd_ctl_elem_write(ALSA_0.9) [ALSA]
snd_ctl_event_clear(ALSA_0.9) [ALSA]	snd_ctl_event_copy(ALSA_0.9) [ALSA]	snd_ctl_event_elem_get_id(ALSA_0.9) [ALSA]
snd_ctl_event_elem_get_mask(ALSA_0.9) [ALSA]	snd_ctl_event_free(ALSA_0.9) [ALSA]	snd_ctl_event_malloc(ALSA_0.9) [ALSA]
snd_ctl_event_sizeof(ALSA_0.9) [ALSA]	snd_ctl_hwdep_info(ALSA_0.9) [ALSA]	snd_ctl_hwdep_next_device(ALSA_0.9) [ALSA]
snd_ctl_name(ALSA_0.9) [ALSA]	snd_ctl_nonblock(ALSA_0.9) [ALSA]	snd_ctl_open(ALSA_0.9) [ALSA]
snd_ctl_pcm_info(ALSA_0.9) [ALSA]	snd_ctl_pcm_next_device(ALSA_0.9) [ALSA]	snd_ctl_poll_descriptors(ALSA_0.9) [ALSA]
snd_ctl_poll_descriptors_count(ALSA_0.9) [ALSA]	snd_ctl_rawmidi_info(ALSA_0.9) [ALSA]	snd_ctl_rawmidi_next_device(ALSA_0.9) [ALSA]
snd_ctl_read(ALSA_0.9) [ALSA]	snd_ctl_subscribe_events(ALSA_0.9) [ALSA]	

6.1.3 ALSA Global defines and functions

6.1.3.1 Interfaces for ALSA Global defines and functions

An LSB conforming implementation shall provide the generic functions for ALSA Global defines and functions specified in Table 6-5, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-5 libasound - ALSA Global defines and functions Function Interfaces

snd_asoundlib_version(ALSA_0.9) [ALSA]	snd_async_add_handler(ALSA_0.9) [ALSA]	snd_async_del_handler(ALSA_0.9) [ALSA]

snd_async_handler_get_callback_private(ALSA_0.9) [ALSA]		

6.1.4 ALSA Hardware Dependant Interface

6.1.4.1 Interfaces for ALSA Hardware Dependant Interface

An LSB conforming implementation shall provide the generic functions for ALSA Hardware Dependant Interface specified in Table 6-6, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-6 libasound - ALSA Hardware Dependant Interface Function Interfaces

snd_hwdep_close(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_copy(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_free(ALSA_0.9) [ALSA]
snd_hwdep_dsp_image_get_image(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_get_index(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_get_length(ALSA_0.9) [ALSA]
snd_hwdep_dsp_image_get_name(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_malloc(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_set_image(ALSA_0.9) [ALSA]
snd_hwdep_dsp_image_set_index(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_set_length(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_set_name(ALSA_0.9) [ALSA]
snd_hwdep_dsp_image_sizeof(ALSA_0.9) [ALSA]	snd_hwdep_dsp_load(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status(ALSA_0.9) [ALSA]
snd_hwdep_dsp_status_copy(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_free(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_get_chip_ready(ALSA_0.9) [ALSA]
snd_hwdep_dsp_status_get_dsp_loaded(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_get_id(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_get_num_dsps(ALSA_0.9) [ALSA]
snd_hwdep_dsp_status_get_version(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_malloc(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_sizeof(ALSA_0.9) [ALSA]
snd_hwdep_info(ALSA_0.9) [ALSA]	snd_hwdep_info_copy(ALSA_0.9) [ALSA]	snd_hwdep_info_free(ALSA_0.9) [ALSA]
snd_hwdep_info_get_card(ALSA_0.9) [ALSA]	snd_hwdep_info_get_device(ALSA_0.9) [ALSA]	snd_hwdep_info_get_id(ALSA_0.9) [ALSA]
snd_hwdep_info_get_iface(ALSA_0.9) [ALSA]	snd_hwdep_info_get_name(ALSA_0.9) [ALSA]	snd_hwdep_info_malloc(ALSA_0.9) [ALSA]
snd_hwdep_info_set_device(ALSA_0.9) [ALSA]	snd_hwdep_info_sizeof(ALSA_0.9) [ALSA]	snd_hwdep_ioctl(ALSA_0.9) [ALSA]

snd_hwdep_open(ALSA_0.9) [ALSA]	snd_hwdep_poll_descriptors(ALSA_0.9) [ALSA]	snd_hwdep_read(ALSA_0.9) [ALSA]
snd_hwdep_write(ALSA_0.9) [ALSA]		

6.1.5 ALSA High level Control Interface

6.1.5.1 Interfaces for ALSA High level Control Interface

An LSB conforming implementation shall provide the generic functions for ALSA High level Control Interface specified in Table 6-7, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-7 libasound - ALSA High level Control Interface Function Interfaces

snd_hctl_close(ALSA_0.9) [ALSA]	snd_hctl_elem_get_callback_private(ALSA_0.9) [ALSA]	snd_hctl_elem_get_id(ALSA_0.9) [ALSA]
snd_hctl_elem_info(ALSA_0.9) [ALSA]	snd_hctl_elem_next(ALSA_0.9) [ALSA]	snd_hctl_elem_prev(ALSA_0.9) [ALSA]
snd_hctl_elem_read(ALSA_0.9) [ALSA]	snd_hctl_elem_set_callback(ALSA_0.9) [ALSA]	snd_hctl_elem_set_callback_private(ALSA_0.9) [ALSA]
snd_hctl_elem_write(ALSA_0.9) [ALSA]	snd_hctl_find_elem(ALSA_0.9) [ALSA]	snd_hctl_first_elem(ALSA_0.9) [ALSA]
snd_hctl_free(ALSA_0.9) [ALSA]	snd_hctl_get_callback_private(ALSA_0.9) [ALSA]	snd_hctl_get_count(ALSA_0.9) [ALSA]
snd_hctl_handle_events(ALSA_0.9) [ALSA]	snd_hctl_last_elem(ALSA_0.9) [ALSA]	snd_hctl_load(ALSA_0.9) [ALSA]
snd_hctl_nonblock(ALSA_0.9) [ALSA]	snd_hctl_open(ALSA_0.9) [ALSA]	snd_hctl_set_callback(ALSA_0.9) [ALSA]
snd_hctl_set_callback_private(ALSA_0.9) [ALSA]	snd_hctl_wait(ALSA_0.9) [ALSA]	

6.1.6 ALSA Input Interface

6.1.6.1 Interfaces for ALSA Input Interface

An LSB conforming implementation shall provide the generic functions for ALSA Input Interface specified in Table 6-8, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-8 libasound - ALSA Input Interface Function Interfaces

snd_input_buffer_open(ALSA_0.9) [ALSA]	snd_input_close(ALSA_0.9) [ALSA]	snd_input_stdio_attach(ALSA_0.9) [ALSA]
snd_input_stdio_open(ALSA_0.9) [ALSA]		

6.1.7 ALSA MIDI Sequencer

6.1.7.1 Interfaces for ALSA MIDI Sequencer

An LSB conforming implementation shall provide the generic functions for ALSA MIDI Sequencer specified in Table 6-9, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-9 libasound - ALSA MIDI Sequencer Function Interfaces

snd_seq_client_id(ALSA_0.9) [ALSA]	snd_seq_close(ALSA_0.9) [ALSA]	snd_seq_get_input_buffer_size(ALSA_0.9) [ALSA]
snd_seq_get_output_buffer_size(ALSA_0.9) [ALSA]	snd_seq_nonblock(ALSA_0.9) [ALSA]	snd_seq_open(ALSA_0.9) [ALSA]
snd_seq_poll_descriptors(ALSA_0.9) [ALSA]	snd_seq_poll_descriptors_count(ALSA_0.9) [ALSA]	snd_seq_poll_descriptors_revents(ALSA_0.9) [ALSA]
snd_seq_set_input_buffer_size(ALSA_0.9) [ALSA]	snd_seq_set_output_buffer_size(ALSA_0.9) [ALSA]	snd_seq_system_info(ALSA_0.9) [ALSA]
snd_seq_system_info_copy(ALSA_0.9) [ALSA]	snd_seq_system_info_free(ALSA_0.9) [ALSA]	snd_seq_system_info_get_clients(ALSA_0.9) [ALSA]
snd_seq_system_info_get_ports(ALSA_0.9) [ALSA]	snd_seq_system_info_get_queues(ALSA_0.9) [ALSA]	snd_seq_system_info_malloc(ALSA_0.9) [ALSA]
snd_seq_system_info_sizeof(ALSA_0.9) [ALSA]		

6.1.8 ALSA Mixer Interface

6.1.8.1 Interfaces for ALSA Mixer Interface

An LSB conforming implementation shall provide the generic functions for ALSA Mixer Interface specified in Table 6-10, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-10 libasound - ALSA Mixer Interface Function Interfaces

snd_mixer_attach(ALSA_0.9) [ALSA]	snd_mixer_close(ALSA_0.9) [ALSA]	snd_mixer_detach(ALSA_0.9) [ALSA]
snd_mixer_elem_get_callback_private(ALSA_0.9) [ALSA]	snd_mixer_elem_get_type(ALSA_0.9) [ALSA]	snd_mixer_elem_next(ALSA_0.9) [ALSA]
snd_mixer_elem_prev(ALSA_0.9) [ALSA]	snd_mixer_elem_set_callback(ALSA_0.9) [ALSA]	snd_mixer_elem_set_callback_private(ALSA_0.9) [ALSA]
snd_mixer_first_elem(ALSA_0.9) [ALSA]	snd_mixer_free(ALSA_0.9) [ALSA]	snd_mixer_get_callback_private(ALSA_0.9)

		[ALSA]
snd_mixer_get_count(ALSA_0.9) [ALSA]	snd_mixer_handle_events(ALSA_0.9) [ALSA]	snd_mixer_last_elem(ALSA_0.9) [ALSA]
snd_mixer_load(ALSA_0.9) [ALSA]	snd_mixer_open(ALSA_0.9) [ALSA]	snd_mixer_poll_descriptors(ALSA_0.9) [ALSA]
snd_mixer_poll_descriptors_count(ALSA_0.9) [ALSA]	snd_mixer_poll_descriptors_revents(ALSA_0.9) [ALSA]	snd_mixer_set_callback(ALSA_0.9) [ALSA]
snd_mixer_set_callback_private(ALSA_0.9) [ALSA]	snd_mixer_wait(ALSA_0.9) [ALSA]	snd_pcm_type_name(ALSA_0.9.0) [ALSA]

6.1.9 ALSA Output Interface

6.1.9.1 Interfaces for ALSA Output Interface

An LSB conforming implementation shall provide the generic functions for ALSA Output Interface specified in Table 6-11, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-11 libasound - ALSA Output Interface Function Interfaces

snd_output_buffer_open(ALSA_0.9) [ALSA]	snd_output_buffer_string(ALSA_0.9) [ALSA]	snd_output_close(ALSA_0.9) [ALSA]
snd_output_putc(ALSA_0.9) [ALSA]	snd_output_puts(ALSA_0.9) [ALSA]	snd_output_stdio_attach(ALSA_0.9) [ALSA]
snd_output_stdio_open(ALSA_0.9) [ALSA]		

6.1.10 ALSA PCM Interface - General Functions

6.1.10.1 Interfaces for ALSA PCM Interface - General Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - General Functions specified in Table 6-12, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-12 libasound - ALSA PCM Interface - General Functions Function Interfaces

snd_async_add_pcm_handler(ALSA_0.9) [ALSA]	snd_async_handler_get_pcm(ALSA_0.9) [ALSA]	snd_pcm_avail_update(ALSA_0.9) [ALSA]
snd_pcm_close(ALSA_0.9) [ALSA]	snd_pcm_delay(ALSA_0.9) [ALSA]	snd_pcm_drain(ALSA_0.9) [ALSA]
snd_pcm_drop(ALSA_0.9) [ALSA]	snd_pcm_forward(ALSA_0.9.0rc8) [ALSA]	snd_pcm_hw_free(ALSA_0.9) [ALSA]
snd_pcm_hw_params(ALSA_0.9) [ALSA]	snd_pcm_hw_params_current(ALSA_0.9) [ALSA]	snd_pcm_hwsync(ALSA_0.9) [ALSA]
snd_pcm_info(ALSA_0	snd_pcm_link(ALSA_0.	snd_pcm_name(ALSA_

.9) [ALSA]	9) [ALSA]	0.9) [ALSA]
snd_pcm_nonblock(ALSA_0.9) [ALSA]	snd_pcm_open(ALSA_0.9) [ALSA]	snd_pcm_open_lconf(ALSA_0.9) [ALSA]
snd_pcm_pause(ALSA_0.9) [ALSA]	snd_pcm_poll_descriptors(ALSA_0.9) [ALSA]	snd_pcm_poll_descriptors_count(ALSA_0.9) [ALSA]
snd_pcm_poll_descriptors_revents(ALSA_0.9) [ALSA]	snd_pcm_prepare(ALSA_0.9) [ALSA]	snd_pcm_readi(ALSA_0.9) [ALSA]
snd_pcm_readn(ALSA_0.9) [ALSA]	snd_pcm_recover(ALSA_0.9) [ALSA]	snd_pcm_reset(ALSA_0.9) [ALSA]
snd_pcm_resume(ALSA_0.9) [ALSA]	snd_pcm_rewind(ALSA_0.9) [ALSA]	snd_pcm_start(ALSA_0.9) [ALSA]
snd_pcm_state(ALSA_0.9) [ALSA]	snd_pcm_status(ALSA_0.9) [ALSA]	snd_pcm_stream(ALSA_0.9) [ALSA]
snd_pcm_sw_params(ALSA_0.9) [ALSA]	snd_pcm_sw_params_current(ALSA_0.9) [ALSA]	snd_pcm_type(ALSA_0.9) [ALSA]
snd_pcm_unlink(ALSA_0.9) [ALSA]	snd_pcm_wait(ALSA_0.9) [ALSA]	snd_pcm_writei(ALSA_0.9) [ALSA]
snd_pcm_writen(ALSA_0.9) [ALSA]		

6.1.11 ALSA PCM Interface - Access Mask Functions

6.1.11.1 Interfaces for ALSA PCM Interface - Access Mask Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Access Mask Functions specified in Table 6-13, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-13 libasound - ALSA PCM Interface - Access Mask Functions Function Interfaces

snd_pcm_access_mask_any(ALSA_0.9) [ALSA]	snd_pcm_access_mask_copy(ALSA_0.9) [ALSA]	snd_pcm_access_mask_free(ALSA_0.9) [ALSA]
snd_pcm_access_mask_malloc(ALSA_0.9) [ALSA]	snd_pcm_access_mask_none(ALSA_0.9) [ALSA]	snd_pcm_access_mask_set(ALSA_0.9) [ALSA]
snd_pcm_access_mask_sizeof(ALSA_0.9) [ALSA]	snd_pcm_access_mask_test(ALSA_0.9) [ALSA]	

6.1.12 ALSA PCM Interface - Debug Functions

6.1.12.1 Interfaces for ALSA PCM Interface - Debug Functions

An LSB conforming implementation shall provide the generic functions for

ALSA PCM Interface - Debug Functions specified in Table 6-14, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-14 libasound - ALSA PCM Interface - Debug Functions Function Interfaces

snd_pcm_dump(ALSA_0.9) [ALSA]	snd_pcm_hw_params_dump(ALSA_0.9) [ALSA]	snd_pcm_status_dump (ALSA_0.9) [ALSA]
snd_pcm_sw_params_dump(ALSA_0.9) [ALSA]		

6.1.13 ALSA PCM Interface - Description Functions

6.1.13.1 Interfaces for ALSA PCM Interface - Description Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Description Functions specified in Table 6-15, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-15 libasound - ALSA PCM Interface - Description Functions Function Interfaces

snd_pcm_access_name(ALSA_0.9) [ALSA]	snd_pcm_format_description(ALSA_0.9) [ALSA]	snd_pcm_format_name (ALSA_0.9) [ALSA]
snd_pcm_format_value (ALSA_0.9) [ALSA]	snd_pcm_state_name(ALSA_0.9) [ALSA]	snd_pcm_stream_name (ALSA_0.9) [ALSA]

6.1.14 ALSA PCM Interface - Direct Access (MMAP) Functions

6.1.14.1 Interfaces for ALSA PCM Interface - Direct Access (MMAP) Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Direct Access (MMAP) Functions specified in Table 6-16, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-16 libasound - ALSA PCM Interface - Direct Access (MMAP) Functions Function Interfaces

snd_pcm_mmap_begin (ALSA_0.9) [ALSA]	snd_pcm_mmap_commit(ALSA_0.9) [ALSA]	snd_pcm_mmap_readi(ALSA_0.9) [ALSA]
snd_pcm_mmap_readn (ALSA_0.9) [ALSA]	snd_pcm_mmap_writei (ALSA_0.9) [ALSA]	snd_pcm_mmap_writen(ALSA_0.9) [ALSA]

6.1.15 ALSA PCM Interface - Format Mask Functions

6.1.15.1 Interfaces for ALSA PCM Interface - Format Mask Functions

An LSB conforming implementation shall provide the generic functions for

ALSA PCM Interface - Format Mask Functions specified in Table 6-17, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-17 libasound - ALSA PCM Interface - Format Mask Functions Function Interfaces

snd_pcm_format_mask_any(ALSA_0.9) [ALSA]	snd_pcm_format_mask_copy(ALSA_0.9) [ALSA]	snd_pcm_format_mask_free(ALSA_0.9) [ALSA]
snd_pcm_format_mask_malloc(ALSA_0.9) [ALSA]	snd_pcm_format_mask_none(ALSA_0.9) [ALSA]	snd_pcm_format_mask_set(ALSA_0.9) [ALSA]
snd_pcm_format_mask_sizeof(ALSA_0.9) [ALSA]	snd_pcm_format_mask_test(ALSA_0.9) [ALSA]	

6.1.16 ALSA PCM Interface - Hardware Parameters

6.1.16.1 Interfaces for ALSA PCM Interface - Hardware Parameters

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Hardware Parameters specified in Table 6-18, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-18 libasound - ALSA PCM Interface - Hardware Parameters Function Interfaces

snd_pcm_hw_params_any(ALSA_0.9) [ALSA]	snd_pcm_hw_params_can_mmap_sample_resolution(ALSA_0.9) [ALSA]	snd_pcm_hw_params_can_pause(ALSA_0.9) [ALSA]
snd_pcm_hw_params_can_resume(ALSA_0.9) [ALSA]	snd_pcm_hw_params_can_sync_start(ALSA_0.9) [ALSA]	snd_pcm_hw_params_copy(ALSA_0.9) [ALSA]
snd_pcm_hw_params_free(ALSA_0.9) [ALSA]	snd_pcm_hw_params_get_access(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_access_mask(ALSA_0.9) [ALSA]
snd_pcm_hw_params_get_buffer_size(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_buffer_size_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_buffer_size_min(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_buffer_time(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_buffer_time_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_buffer_time_min(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_channels(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_channels_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_channels_min(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_format(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_format_mask(ALSA_0.9) [ALSA]	snd_pcm_hw_params_get_period_size(ALSA_0.9.0rc4) [ALSA]

snd_pcm_hw_params_get_period_size_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_period_size_min(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_period_time(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_period_time_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_period_time_min(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_periods(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_periods_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_periods_min(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_rate(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_rate_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_rate_min(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_rate_numden(ALSA_0.9) [ALSA]
snd_pcm_hw_params_get_rate_resample(ALSA_0.9) [ALSA]	snd_pcm_hw_params_get_sbits(ALSA_0.9) [ALSA]	snd_pcm_hw_params_is_double(ALSA_0.9) [ALSA]
snd_pcm_hw_params_is_half_duplex(ALSA_0.9) [ALSA]	snd_pcm_hw_params_is_joint_duplex(ALSA_0.9) [ALSA]	snd_pcm_hw_params_malloc(ALSA_0.9) [ALSA]
snd_pcm_hw_params_set_access(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_access_mask(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_buffer_size(ALSA_0.9) [ALSA]
snd_pcm_hw_params_set_buffer_size_near(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_set_buffer_time(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_buffer_time_near(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_set_channels(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_channels_near(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_set_format(ALSA_0.9) [ALSA]
snd_pcm_hw_params_set_format_mask(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_period_size(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_period_size_near(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_set_period_time(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_period_time_near(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_set_periods(ALSA_0.9) [ALSA]
snd_pcm_hw_params_set_periods_integer(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_periods_near(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_set_rate(ALSA_0.9) [ALSA]
snd_pcm_hw_params_set_rate_near(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_set_rate_resample(ALSA_0.9) [ALSA]	snd_pcm_hw_params_sizeof(ALSA_0.9) [ALSA]
snd_pcm_hw_params_test_access(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_buffer_size(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_buffer_time(ALSA_0.9) [ALSA]
snd_pcm_hw_params_test_channels(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_format(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_period_size(ALSA_0.9) [ALSA]
snd_pcm_hw_params_test_period_time(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_periods(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_rate(ALSA_0.9) [ALSA]

6.1.17 ALSA PCM Interface - Helper Functions

6.1.17.1 Interfaces for ALSA PCM Interface - Helper Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Helper Functions specified in Table 6-19, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-19 libasound - ALSA PCM Interface - Helper Functions Function Interfaces

snd_pcm_area_copy(ALSA_0.9) [ALSA]	snd_pcm_area_silence(ALSA_0.9) [ALSA]	snd_pcm_areas_copy(ALSA_0.9) [ALSA]
snd_pcm_areas_silence(ALSA_0.9) [ALSA]	snd_pcm_build_linear_format(ALSA_0.9) [ALSA]	snd_pcm_bytes_to_frames(ALSA_0.9) [ALSA]
snd_pcm_bytes_to_samples(ALSA_0.9) [ALSA]	snd_pcm_format_big_endian(ALSA_0.9) [ALSA]	snd_pcm_format_cpu_endian(ALSA_0.9) [ALSA]
snd_pcm_format_float(ALSA_0.9) [ALSA]	snd_pcm_format_linear(ALSA_0.9) [ALSA]	snd_pcm_format_little_endian(ALSA_0.9) [ALSA]
snd_pcm_format_physical_width(ALSA_0.9) [ALSA]	snd_pcm_format_set_silence(ALSA_0.9) [ALSA]	snd_pcm_format_signed(ALSA_0.9) [ALSA]
snd_pcm_format_size(ALSA_0.9) [ALSA]	snd_pcm_format_unsigned(ALSA_0.9) [ALSA]	snd_pcm_format_width(ALSA_0.9) [ALSA]
snd_pcm_frames_to_bytes(ALSA_0.9) [ALSA]	snd_pcm_samples_to_bytes(ALSA_0.9) [ALSA]	

6.1.18 ALSA PCM Interface - Software Parameters

6.1.18.1 Interfaces for ALSA PCM Interface - Software Parameters

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Software Parameters specified in Table 6-20, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-20 libasound - ALSA PCM Interface - Software Parameters Function Interfaces

snd_pcm_sw_params_copy(ALSA_0.9) [ALSA]	snd_pcm_sw_params_free(ALSA_0.9) [ALSA]	snd_pcm_sw_params_get_avail_min(ALSA_0.9.0rc4) [ALSA]
snd_pcm_sw_params_get_boundary(ALSA_0.9) [ALSA]	snd_pcm_sw_params_get_silence_size(ALSA_0.9.0rc4) [ALSA]	snd_pcm_sw_params_get_silence_threshold(ALSA_0.9.0rc4) [ALSA]
snd_pcm_sw_params_get_start_threshold(ALSA_0.9.0rc4) [ALSA]	snd_pcm_sw_params_get_stop_threshold(ALSA_0.9.0rc4) [ALSA]	snd_pcm_sw_params_get_tstamp_mode(ALSA_0.9.0rc4) [ALSA]
snd_pcm_sw_params_	snd_pcm_sw_params_s	snd_pcm_sw_params_s

malloc(ALSA_0.9) [ALSA]	et_avail_min(ALSA_0.9) [ALSA]	et_silence_size(ALSA_0.9) [ALSA]
snd_pcm_sw_params_set_silence_threshold(ALSA_0.9) [ALSA]	snd_pcm_sw_params_set_start_threshold(ALSA_0.9) [ALSA]	snd_pcm_sw_params_set_stop_threshold(ALSA_0.9) [ALSA]
snd_pcm_sw_params_set_tstamp_mode(ALSA_0.9) [ALSA]	snd_pcm_sw_params_set_xfer_align(ALSA_0.9) [ALSA]	snd_pcm_sw_params_sizeof(ALSA_0.9) [ALSA]

6.1.19 ALSA PCM Interface - Status Functions

6.1.19.1 Interfaces for ALSA PCM Interface - Status Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Status Functions specified in Table 6-21, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-21 libasound - ALSA PCM Interface - Status Functions Function Interfaces

snd_pcm_status_copy(ALSA_0.9) [ALSA]	snd_pcm_status_free(ALSA_0.9) [ALSA]	snd_pcm_status_get_avail(ALSA_0.9) [ALSA]
snd_pcm_status_get_avail_max(ALSA_0.9) [ALSA]	snd_pcm_status_get_delay(ALSA_0.9) [ALSA]	snd_pcm_status_get_state(ALSA_0.9) [ALSA]
snd_pcm_status_get_trigger_tstamp(ALSA_0.9) [ALSA]	snd_pcm_status_get_tstamp(ALSA_0.9) [ALSA]	snd_pcm_status_malloc(ALSA_0.9) [ALSA]
snd_pcm_status_sizeof(ALSA_0.9) [ALSA]		

6.1.20 ALSA PCM Interface - Stream Information

6.1.20.1 Interfaces for ALSA PCM Interface - Stream Information

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Stream Information specified in Table 6-22, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-22 libasound - ALSA PCM Interface - Stream Information Function Interfaces

snd_pcm_info_copy(ALSA_0.9) [ALSA]	snd_pcm_info_free(ALSA_0.9) [ALSA]	snd_pcm_info_get_card(ALSA_0.9) [ALSA]
snd_pcm_info_get_class(ALSA_0.9) [ALSA]	snd_pcm_info_get_device(ALSA_0.9) [ALSA]	snd_pcm_info_get_id(ALSA_0.9) [ALSA]
snd_pcm_info_get_name(ALSA_0.9) [ALSA]	snd_pcm_info_get_stream(ALSA_0.9) [ALSA]	snd_pcm_info_get_subdevice(ALSA_0.9) [ALSA]
snd_pcm_info_get_subdevice_name(ALSA_0.9) [ALSA]	snd_pcm_info_get_subdevices_avail(ALSA_0.9) [ALSA]	snd_pcm_info_get_subdevices_count(ALSA_0.9) [ALSA]

snd_pcm_info_malloc(ALSA_0.9) [ALSA]	snd_pcm_info_set_device(ALSA_0.9) [ALSA]	snd_pcm_info_set_stream(ALSA_0.9) [ALSA]
snd_pcm_info_set_subdevice(ALSA_0.9) [ALSA]	snd_pcm_info_sizeof(ALSA_0.9) [ALSA]	

6.1.21 ALSA Sequencer Event Type Checks

6.1.21.1 Interfaces for ALSA Sequencer Event Type Checks

No external functions are defined for libasound - ALSA Sequencer Event Type Checks in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for ALSA Sequencer Event Type Checks specified in Table 6-23, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-23 libasound - ALSA Sequencer Event Type Checks Data Interfaces

snd_seq_event_types(ALSA_0.9) [ALSA]		

6.1.22 ALSA Error Handling

6.1.22.1 Interfaces for ALSA Error Handling

An LSB conforming implementation shall provide the generic functions for ALSA Error Handling specified in Table 6-24, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-24 libasound - ALSA Error Handling Function Interfaces

snd_lib_error_set_handler(ALSA_0.9) [ALSA]	snd_strerror(ALSA_0.9) [ALSA]	

6.1.23 ALSA RawMidi Interface

6.1.23.1 Interfaces for ALSA RawMidi Interface

An LSB conforming implementation shall provide the generic functions for ALSA RawMidi Interface specified in Table 6-25, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-25 libasound - ALSA RawMidi Interface Function Interfaces

snd_rawmidi_close(ALSA_0.9) [ALSA]	snd_rawmidi_drain(ALSA_0.9) [ALSA]	snd_rawmidi_drop(ALSA_0.9) [ALSA]
snd_rawmidi_info_free(ALSA_0.9) [ALSA]	snd_rawmidi_info_get_id(ALSA_0.9) [ALSA]	snd_rawmidi_info_get_name(ALSA_0.9) [ALSA]
snd_rawmidi_info_get_subdevice_name(ALSA_0.9) [ALSA]	snd_rawmidi_info_get_subdevices_count(ALSA_0.9) [ALSA]	snd_rawmidi_info_malloc(ALSA_0.9) [ALSA]
snd_rawmidi_info_set_device(ALSA_0.9)	snd_rawmidi_info_set_stream(ALSA_0.9)	snd_rawmidi_info_set_subdevice(ALSA_0.9)

[ALSA]	[ALSA]	[ALSA]
snd_rawmidi_info_sizeof(ALSA_0.9) [ALSA]	snd_rawmidi_nonblock(ALSA_0.9) [ALSA]	snd_rawmidi_open(ALSA_0.9) [ALSA]
snd_rawmidi_poll_descriptors(ALSA_0.9) [ALSA]	snd_rawmidi_poll_descriptors_count(ALSA_0.9) [ALSA]	snd_rawmidi_poll_descriptors_revents(ALSA_0.9) [ALSA]
snd_rawmidi_read(ALSA_0.9) [ALSA]	snd_rawmidi_write(ALSA_0.9) [ALSA]	

6.1.24 ALSA Sequencer Client Interface

6.1.24.1 Interfaces for ALSA Sequencer Client Interface

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Client Interface specified in Table 6-26, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-26 libasound - ALSA Sequencer Client Interface Function Interfaces

snd_seq_client_info_copy(ALSA_0.9) [ALSA]	snd_seq_client_info_free(ALSA_0.9) [ALSA]	snd_seq_client_info_get_client(ALSA_0.9) [ALSA]
snd_seq_client_info_get_name(ALSA_0.9) [ALSA]	snd_seq_client_info_get_num_ports(ALSA_0.9) [ALSA]	snd_seq_client_info_get_type(ALSA_0.9) [ALSA]
snd_seq_client_info_malloc(ALSA_0.9) [ALSA]	snd_seq_client_info_set_client(ALSA_0.9) [ALSA]	snd_seq_client_info_set_name(ALSA_0.9) [ALSA]
snd_seq_client_info_sizeof(ALSA_0.9) [ALSA]	snd_seq_get_any_client_info(ALSA_0.9) [ALSA]	snd_seq_get_client_info(ALSA_0.9) [ALSA]
snd_seq_query_next_client(ALSA_0.9) [ALSA]	snd_seq_set_client_info(ALSA_0.9) [ALSA]	

6.1.25 ALSA Sequencer Event API

6.1.25.1 Interfaces for ALSA Sequencer Event API

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Event API specified in Table 6-27, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-27 libasound - ALSA Sequencer Event API Function Interfaces

snd_seq_drain_output(ALSA_0.9) [ALSA]	snd_seq_drop_output(ALSA_0.9) [ALSA]	snd_seq_drop_output_buffer(ALSA_0.9) [ALSA]
snd_seq_event_input(ALSA_0.9) [ALSA]	snd_seq_event_input_pending(ALSA_0.9) [ALSA]	snd_seq_event_length(ALSA_0.9) [ALSA]
snd_seq_event_output(ALSA_0.9) [ALSA]	snd_seq_event_output_direct(ALSA_0.9)	snd_seq_free_event(ALSA_0.9) [ALSA]

	[ALSA]	

6.1.26 ALSA Sequencer Middle Level Interface

6.1.26.1 Interfaces for ALSA Sequencer Middle Level Interface

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Middle Level Interface specified in Table 6-28, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-28 libasound - ALSA Sequencer Middle Level Interface Function Interfaces

snd_seq_connect_from(ALSA_0.9) [ALSA]	snd_seq_connect_to(ALSA_0.9) [ALSA]	snd_seq_control_queue(ALSA_0.9) [ALSA]
snd_seq_create_simple_port(ALSA_0.9) [ALSA]	snd_seq_delete_simple_port(ALSA_0.9) [ALSA]	snd_seq_disconnect_from(ALSA_0.9) [ALSA]
snd_seq_disconnect_to(ALSA_0.9) [ALSA]	snd_seq_parse_address(ALSA_0.9) [ALSA]	snd_seq_set_client_name(ALSA_0.9) [ALSA]
snd_seq_sync_output_queue(ALSA_0.9) [ALSA]		

6.1.27 ALSA Sequencer Port Interface

6.1.27.1 Interfaces for ALSA Sequencer Port Interface

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Port Interface specified in Table 6-29, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-29 libasound - ALSA Sequencer Port Interface Function Interfaces

snd_seq_create_port(ALSA_0.9) [ALSA]	snd_seq_delete_port(ALSA_0.9) [ALSA]	snd_seq_get_any_port_info(ALSA_0.9) [ALSA]
snd_seq_get_port_info(ALSA_0.9) [ALSA]	snd_seq_port_info_copy(ALSA_0.9) [ALSA]	snd_seq_port_info_free(ALSA_0.9) [ALSA]
snd_seq_port_info_get_addr(ALSA_0.9) [ALSA]	snd_seq_port_info_get_capability(ALSA_0.9) [ALSA]	snd_seq_port_info_get_client(ALSA_0.9) [ALSA]
snd_seq_port_info_get_name(ALSA_0.9) [ALSA]	snd_seq_port_info_get_port(ALSA_0.9) [ALSA]	snd_seq_port_info_get_type(ALSA_0.9) [ALSA]
snd_seq_port_info_malloc(ALSA_0.9) [ALSA]	snd_seq_port_info_set_capability(ALSA_0.9) [ALSA]	snd_seq_port_info_set_client(ALSA_0.9) [ALSA]
snd_seq_port_info_set_midi_channels(ALSA_0.9) [ALSA]	snd_seq_port_info_set_name(ALSA_0.9) [ALSA]	snd_seq_port_info_set_port(ALSA_0.9) [ALSA]
snd_seq_port_info_set_port_specified(ALSA_0	snd_seq_port_info_set_timestamp_queue(ALS	snd_seq_port_info_set_timestamp_real(ALSA_

.9) [ALSA]	A_0.9) [ALSA]	0.9) [ALSA]
snd_seq_port_info_set_timestamping(ALSA_0.9) [ALSA]	snd_seq_port_info_set_type(ALSA_0.9) [ALSA]	snd_seq_port_info_sizeof(ALSA_0.9) [ALSA]
snd_seq_query_next_port(ALSA_0.9) [ALSA]	snd_seq_set_port_info(ALSA_0.9) [ALSA]	

6.1.28 ALSA Sequencer Port Subscription

6.1.28.1 Interfaces for ALSA Sequencer Port Subscription

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Port Subscription specified in Table 6-30, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-30 libasound - ALSA Sequencer Port Subscription Function Interfaces

snd_seq_get_port_subscription(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_copy(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_free(ALSA_0.9) [ALSA]
snd_seq_port_subscribe_get_dest(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_get_exclusive(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_get_queue(ALSA_0.9) [ALSA]
snd_seq_port_subscribe_get_sender(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_get_time_real(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_get_time_update(ALSA_0.9) [ALSA]
snd_seq_port_subscribe_malloc(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_set_dest(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_set_exclusive(ALSA_0.9) [ALSA]
snd_seq_port_subscribe_set_queue(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_set_sender(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_set_time_real(ALSA_0.9) [ALSA]
snd_seq_port_subscribe_set_time_update(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_sizeof(ALSA_0.9) [ALSA]	snd_seq_query_port_subscribers(ALSA_0.9) [ALSA]
snd_seq_query_subscribe_copy(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_free(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_get_addr(ALSA_0.9) [ALSA]
snd_seq_query_subscribe_get_exclusive(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_get_index(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_get_queue(ALSA_0.9) [ALSA]
snd_seq_query_subscribe_get_root(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_get_time_real(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_get_time_update(ALSA_0.9) [ALSA]
snd_seq_query_subscribe_malloc(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_set_index(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_set_root(ALSA_0.9) [ALSA]
snd_seq_query_subscribe_set_type(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_sizeof(ALSA_0.9) [ALSA]	snd_seq_subscribe_port(ALSA_0.9) [ALSA]

snd_seq_unsubscribe_port(ALSA_0.9) [ALSA]		

6.1.29 ALSA Sequencer Queue Interface

6.1.29.1 Interfaces for ALSA Sequencer Queue Interface

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Queue Interface specified in Table 6-31, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-31 libasound - ALSA Sequencer Queue Interface Function Interfaces

snd_seq_alloc_named_queue(ALSA_0.9) [ALSA]	snd_seq_alloc_queue(ALSA_0.9) [ALSA]	snd_seq_free_queue(ALSA_0.9) [ALSA]
snd_seq_get_queue_status(ALSA_0.9) [ALSA]	snd_seq_get_queue_tempo(ALSA_0.9) [ALSA]	snd_seq_queue_status_copy(ALSA_0.9) [ALSA]
snd_seq_queue_status_free(ALSA_0.9) [ALSA]	snd_seq_queue_status_get_real_time(ALSA_0.9) [ALSA]	snd_seq_queue_status_get_tick_time(ALSA_0.9) [ALSA]
snd_seq_queue_status_malloc(ALSA_0.9) [ALSA]	snd_seq_queue_status_sizeof(ALSA_0.9) [ALSA]	snd_seq_queue_tempo_copy(ALSA_0.9) [ALSA]
snd_seq_queue_tempo_free(ALSA_0.9) [ALSA]	snd_seq_queue_tempo_get_ppq(ALSA_0.9) [ALSA]	snd_seq_queue_tempo_get_tempo(ALSA_0.9) [ALSA]
snd_seq_queue_tempo_malloc(ALSA_0.9) [ALSA]	snd_seq_queue_tempo_set_ppq(ALSA_0.9) [ALSA]	snd_seq_queue_tempo_set_tempo(ALSA_0.9) [ALSA]
snd_seq_queue_tempo_sizeof(ALSA_0.9) [ALSA]	snd_seq_set_queue_tempo(ALSA_0.9) [ALSA]	

6.1.30 ALSA Sequencer event - MIDI byte stream coder

6.1.30.1 Interfaces for ALSA Sequencer event - MIDI byte stream coder

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer event - MIDI byte stream coder specified in Table 6-32, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-32 libasound - ALSA Sequencer event - MIDI byte stream coder Function Interfaces

snd_midi_event_decode(ALSA_0.9) [ALSA]	snd_midi_event_encode(ALSA_0.9) [ALSA]	snd_midi_event_encode_byte(ALSA_0.9) [ALSA]

snd_midi_event_free(ALSA_0.9) [ALSA]	snd_midi_event_init(ALSA_0.9) [ALSA]	snd_midi_event_new(ALSA_0.9) [ALSA]
snd_midi_event_reset_decode(ALSA_0.9) [ALSA]	snd_midi_event_reset_encode(ALSA_0.9) [ALSA]	

6.1.31 ALSA Simple Mixer Interface

6.1.31.1 Interfaces for ALSA Simple Mixer Interface

An LSB conforming implementation shall provide the generic functions for ALSA Simple Mixer Interface specified in Table 6-33, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-33 libasound - ALSA Simple Mixer Interface Function Interfaces

snd_mixer_find_selem(ALSA_0.9) [ALSA]	snd_mixer_selem_channel_name(ALSA_0.9) [ALSA]	snd_mixer_selem_get_capture_group(ALSA_0.9) [ALSA]
snd_mixer_selem_get_capture_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_get_capture_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_get_capture_volume_range(ALSA_0.9) [ALSA]
snd_mixer_selem_get_enum_item(ALSA_0.9) [ALSA]	snd_mixer_selem_get_enum_item_name(ALSA_0.9) [ALSA]	snd_mixer_selem_get_enum_items(ALSA_0.9) [ALSA]
snd_mixer_selem_get_id(ALSA_0.9) [ALSA]	snd_mixer_selem_get_index(ALSA_0.9) [ALSA]	snd_mixer_selem_get_name(ALSA_0.9) [ALSA]
snd_mixer_selem_get_playback_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_get_playback_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_get_playback_volume_range(ALSA_0.9) [ALSA]
snd_mixer_selem_has_capture_channel(ALSA_0.9) [ALSA]	snd_mixer_selem_has_capture_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_has_capture_switch_exclusive(ALSA_0.9) [ALSA]
snd_mixer_selem_has_capture_switch_joined(ALSA_0.9) [ALSA]	snd_mixer_selem_has_capture_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_has_capture_volume_joined(ALSA_0.9) [ALSA]
snd_mixer_selem_has_common_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_has_common_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_has_playback_channel(ALSA_0.9) [ALSA]
snd_mixer_selem_has_playback_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_has_playback_switch_joined(ALSA_0.9) [ALSA]	snd_mixer_selem_has_playback_volume(ALSA_0.9) [ALSA]
snd_mixer_selem_has_playback_volume_joined(ALSA_0.9) [ALSA]	snd_mixer_selem_id_copy(ALSA_0.9) [ALSA]	snd_mixer_selem_id_free(ALSA_0.9) [ALSA]
snd_mixer_selem_id_get_index(ALSA_0.9) [ALSA]	snd_mixer_selem_id_get_name(ALSA_0.9) [ALSA]	snd_mixer_selem_id_malloc(ALSA_0.9) [ALSA]
snd_mixer_selem_id_se	snd_mixer_selem_id_se	snd_mixer_selem_id_si

t_index(ALSA_0.9) [ALSA]	t_name(ALSA_0.9) [ALSA]	zeof(ALSA_0.9) [ALSA]
snd_mixer_selem_is_active(ALSA_0.9) [ALSA]	snd_mixer_selem_is_capture_mono(ALSA_0.9) [ALSA]	snd_mixer_selem_is_enum_capture(ALSA_0.9) [ALSA]
snd_mixer_selem_is_enum_playback(ALSA_0.9) [ALSA]	snd_mixer_selem_is_enumerated(ALSA_0.9) [ALSA]	snd_mixer_selem_is_playback_mono(ALSA_0.9) [ALSA]
snd_mixer_selem_register(ALSA_0.9) [ALSA]	snd_mixer_selem_set_capture_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_set_capture_switch_all(ALSA_0.9) [ALSA]
snd_mixer_selem_set_capture_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_set_capture_volume_all(ALSA_0.9) [ALSA]	snd_mixer_selem_set_capture_volume_range(ALSA_0.9) [ALSA]
snd_mixer_selem_set_enum_item(ALSA_0.9) [ALSA]	snd_mixer_selem_set_playback_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_set_playback_switch_all(ALSA_0.9) [ALSA]
snd_mixer_selem_set_playback_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_set_playback_volume_all(ALSA_0.9) [ALSA]	snd_mixer_selem_set_playback_volume_range(ALSA_0.9) [ALSA]

6.1.32 ALSA Timer Interface

6.1.32.1 Interfaces for ALSA Timer Interface

An LSB conforming implementation shall provide the generic functions for ALSA Timer Interface specified in Table 6-34, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-34 libasound - ALSA Timer Interface Function Interfaces

snd_timer_close(ALSA_0.9) [ALSA]	snd_timer_continue(ALSA_0.9) [ALSA]	snd_timer_id_copy(ALSA_0.9) [ALSA]
snd_timer_id_free(ALSA_0.9) [ALSA]	snd_timer_id_get_card(ALSA_0.9) [ALSA]	snd_timer_id_get_class(ALSA_0.9) [ALSA]
snd_timer_id_get_device(ALSA_0.9) [ALSA]	snd_timer_id_get_sclass(ALSA_0.9) [ALSA]	snd_timer_id_get_subdevice(ALSA_0.9) [ALSA]
snd_timer_id_malloc(ALSA_0.9) [ALSA]	snd_timer_id_set_card(ALSA_0.9) [ALSA]	snd_timer_id_set_class(ALSA_0.9) [ALSA]
snd_timer_id_set_device(ALSA_0.9) [ALSA]	snd_timer_id_set_sclass(ALSA_0.9) [ALSA]	snd_timer_id_set_subdevice(ALSA_0.9) [ALSA]
snd_timer_id_sizeof(ALSA_0.9) [ALSA]	snd_timer_info(ALSA_0.9) [ALSA]	snd_timer_info_copy(ALSA_0.9) [ALSA]
snd_timer_info_free(ALSA_0.9) [ALSA]	snd_timer_info_get_card(ALSA_0.9) [ALSA]	snd_timer_info_get_id(ALSA_0.9) [ALSA]
snd_timer_info_get_name(ALSA_0.9) [ALSA]	snd_timer_info_get_resolution(ALSA_0.9) [ALSA]	snd_timer_info_malloc(ALSA_0.9) [ALSA]

snd_timer_info_sizeof(ALSA_0.9) [ALSA]	snd_timer_open(ALSA_0.9) [ALSA]	snd_timer_params(ALSA_0.9) [ALSA]
snd_timer_params_get_ticks(ALSA_0.9) [ALSA]	snd_timer_params_malloc(ALSA_0.9) [ALSA]	snd_timer_params_set_auto_start(ALSA_0.9) [ALSA]
snd_timer_params_set_ticks(ALSA_0.9) [ALSA]	snd_timer_poll_descriptors(ALSA_0.9) [ALSA]	snd_timer_poll_descriptors_count(ALSA_0.9) [ALSA]
snd_timer_read(ALSA_0.9) [ALSA]	snd_timer_start(ALSA_0.9) [ALSA]	snd_timer_status(ALSA_0.9) [ALSA]
snd_timer_stop(ALSA_0.9) [ALSA]		

6.2 Data Definitions for libasound

This section defines global identifiers and their values that are associated with interfaces contained in libasound. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

6.2.1 alsa/conf.h

```
typedef struct _snd_config_iterator *snd_config_iterator_t;
typedef struct _snd_config snd_config_t;
typedef enum _snd_config_type {
    SND_CONFIG_TYPE_INTEGER,
    SND_CONFIG_TYPE_INTEGER64 = 1,
    SND_CONFIG_TYPE_REAL = 2,
    SND_CONFIG_TYPE_STRING = 3,
    SND_CONFIG_TYPE_POINTER = 4,
    SND_CONFIG_TYPE_COMPOUND = 1024
} snd_config_type_t;
typedef struct _snd_config_update snd_config_update_t;
typedef struct snd_devname {
    char *name;
    char *comment;
    snd_devname_t *next;
} snd_devname_t;
extern snd_config_t *snd_config;
extern int snd_config_add(snd_config_t * config, snd_config_t *
leaf);
extern int snd_config_copy(snd_config_t * *dst, snd_config_t *
src);
extern int snd_config_delete(snd_config_t * config);
extern int snd_config_get_ascii(const snd_config_t * config, char
```

```
**value);
extern int snd_config_get_id(const snd_config_t * config,
                             const char **value);
extern int snd_config_get_integer(const snd_config_t * config,
                                  long int *value);
extern int snd_config_get_integer64(const snd_config_t * config,
                                    long long int *value);
extern int snd_config_get_string(const snd_config_t * config,
                                 const char **value);
extern snd_config_type_t snd_config_get_type(const snd_config_t *
config);
extern int snd_config_imake_integer(snd_config_t * *config,
                                    const char *key, const long
int value);
extern int snd_config_imake_integer64(snd_config_t * *config,
                                      const char *key,
                                      const long long int value);
extern int snd_config_imake_string(snd_config_t * *config, const
char *key,
                                   const char *ascii);
extern     snd_config_iterator_t     snd_config_iterator_end(const
snd_config_t *
                                                          node);
extern        snd_config_t        *snd_config_iterator_entry(const
snd_config_iterator_t
                                                    iterator);
extern    snd_config_iterator_t    snd_config_iterator_first(const
snd_config_t *
                                                           node);
extern snd_config_iterator_t snd_config_iterator_next(const
                                                           snd_config_
iterator_t
                                                          iterator);
extern int snd_config_load(snd_config_t * config, snd_input_t *
in);
extern int snd_config_make_compound(snd_config_t * *config,
                                    const char *key, int join);
extern int snd_config_make_integer(snd_config_t * *config,
                                   const char *key);
extern int snd_config_make_integer64(snd_config_t * *config,
                                     const char *key);
extern int snd_config_make_string(snd_config_t * *config, const
char *key);
extern int snd_config_save(snd_config_t * config, snd_output_t *
out);
extern int snd_config_search(snd_config_t * config, const char
*key,
                             snd_config_t * *result);
extern int snd_config_searchv(snd_config_t * config,
                              snd_config_t * *result, ...);
extern int snd_config_set_ascii(snd_config_t * config, const char
*ascii);
extern int snd_config_set_integer(snd_config_t * config, long int
value);
extern int snd_config_set_integer64(snd_config_t * config,
                                    long long int value);
extern int snd_config_set_string(snd_config_t * config, const
char *value);
extern int snd_config_top(snd_config_t * *config);
extern int snd_config_update(void);
extern int snd_config_update_free_global(void);
```

6.2.2 alsa/control.h

```
#define SND_CTL_EVENT_MASK_VALUE        (1<<0)
#define SND_CTL_EVENT_MASK_INFO (1<<1)
#define SND_CTL_EVENT_MASK_ADD  (1<<2)
#define SND_CTL_EVENT_MASK_TLV  (1<<3)
#define SND_CTL_POWER_D3hot     (SND_CTL_POWER_D3|0x0000)
#define SND_CTL_POWER_D3cold    (SND_CTL_POWER_D3|0x0001)
#define SND_CTL_EVENT_MASK_REMOVE       (~0U)
#define SND_CTL_TLV_DB_GAIN_MUTE        -9999999
#define SND_CTL_POWER_D0        0x0000
#define SND_CTL_TLVT_CONTAINER  0x0000
#define SND_CTL_NONBLOCK        0x0001
#define SND_CTL_TLVT_DB_SCALE   0x0001
#define SND_SCTL_NOFREE 0x0001
#define SND_CTL_ASYNC   0x0002
#define SND_CTL_TLVT_DB_LINEAR  0x0002
#define SND_CTL_TLVT_DB_RANGE   0x0003
#define SND_CTL_READONLY        0x0004
#define SND_CTL_POWER_D1        0x0100
#define SND_CTL_POWER_D2        0x0200
#define SND_CTL_POWER_D3        0x0300
#define SND_CTL_POWER_MASK      0xff00

typedef struct snd_aes_iec958 {
    unsigned char status[24];
    unsigned char subcode[147];
    unsigned char pad;
    unsigned char dig_subframe[4];
} snd_aes_iec958_t;
typedef struct _snd_ctl_card_info snd_ctl_card_info_t;
typedef struct sndrv_ctl_elem_id snd_ctl_elem_id_t;
typedef enum _snd_ctl_elem_iface {
    SND_CTL_ELEM_IFACE_CARD,
    SND_CTL_ELEM_IFACE_HWDEP = 1,
    SND_CTL_ELEM_IFACE_MIXER = 2,
    SND_CTL_ELEM_IFACE_PCM = 3,
    SND_CTL_ELEM_IFACE_RAWMIDI = 4,
    SND_CTL_ELEM_IFACE_TIMER = 5,
    SND_CTL_ELEM_IFACE_SEQUENCER = 6,
    SND_CTL_ELEM_IFACE_LAST = 6
} snd_ctl_elem_iface_t;
typedef struct _snd_ctl_elem_info snd_ctl_elem_info_t;
typedef struct sndrv_ctl_elem_list snd_ctl_elem_list_t;
typedef enum _snd_ctl_elem_type {
    SND_CTL_ELEM_TYPE_NONE,
    SND_CTL_ELEM_TYPE_BOOLEAN = 1,
    SND_CTL_ELEM_TYPE_INTEGER = 2,
    SND_CTL_ELEM_TYPE_ENUMERATED = 3,
    SND_CTL_ELEM_TYPE_BYTES = 4,
    SND_CTL_ELEM_TYPE_IEC958 = 5,
    SND_CTL_ELEM_TYPE_INTEGER64 = 6,
    SND_CTL_ELEM_TYPE_LAST = 6
} snd_ctl_elem_type_t;
typedef struct _snd_ctl_elem_value snd_ctl_elem_value_t;
typedef struct sndrv_ctl_event snd_ctl_event_t;
typedef enum _snd_ctl_event_type {
    SND_CTL_EVENT_ELEM,
    SND_CTL_EVENT_LAST
} snd_ctl_event_type_t;
typedef struct _snd_ctl snd_ctl_t;
typedef enum _snd_ctl_type {
    SND_CTL_TYPE_HW,
    SND_CTL_TYPE_SHM = 1,
    SND_CTL_TYPE_INET = 2,
    SND_CTL_TYPE_EXT = 3
} snd_ctl_type_t;
typedef struct _snd_hctl snd_hctl_t;
```

```
typedef struct _snd_sctl snd_sctl_t;
typedef struct _snd_hctl_elem snd_hctl_elem_t;
typedef int (*snd_hctl_compare_t) (const snd_hctl_elem_t *,
                                   const snd_hctl_elem_t *);
typedef int (*snd_hctl_callback_t) (snd_hctl_t *, unsigned int,
                                    snd_hctl_elem_t *);
typedef   int   (*snd_hctl_elem_callback_t)   (snd_hctl_elem_t   *,
unsigned int);
extern    int    snd_async_add_ctl_handler(snd_async_handler_t    *
*handler,
                                     snd_ctl_t * ctl,
                                             snd_async_callback_t
callback,
                                     void *private_data);
extern snd_ctl_t *snd_async_handler_get_ctl(snd_async_handler_t *
handler);
extern int snd_card_get_index(const char *name);
extern int snd_card_get_longname(int card, char **name);
extern int snd_card_get_name(int card, char **name);
extern int snd_card_load(int card);
extern int snd_card_next(int *card);
extern int snd_ctl_card_info(snd_ctl_t * ctl, snd_ctl_card_info_t
* info);
extern void snd_ctl_card_info_clear(snd_ctl_card_info_t * obj);
extern void snd_ctl_card_info_copy(snd_ctl_card_info_t * dst,
                                   const snd_ctl_card_info_t *
src);
extern void snd_ctl_card_info_free(snd_ctl_card_info_t * obj);
extern const char *snd_ctl_card_info_get_components(const
                                                    snd_ctl_card_
info_t *
                                                   obj);
extern     const     char     *snd_ctl_card_info_get_driver(const
snd_ctl_card_info_t *
                                                 obj);
extern      const      char      *snd_ctl_card_info_get_id(const
snd_ctl_card_info_t *
                                              obj);
extern    const    char    *snd_ctl_card_info_get_longname(const
snd_ctl_card_info_t
                                                  * obj);
extern const char *snd_ctl_card_info_get_mixername(const
                                                    snd_ctl_card_i
nfo_t *
                                                   obj);
extern      const      char      *snd_ctl_card_info_get_name(const
snd_ctl_card_info_t *
                                               obj);
extern int snd_ctl_card_info_malloc(snd_ctl_card_info_t * *ptr);
extern size_t snd_ctl_card_info_sizeof(void);
extern int snd_ctl_close(snd_ctl_t * ctl);
extern int snd_ctl_elem_add_boolean(snd_ctl_t * ctl,
                                    const snd_ctl_elem_id_t * id,
                                    unsigned int count);
extern int snd_ctl_elem_add_iec958(snd_ctl_t * ctl,
                                   const snd_ctl_elem_id_t * id);
extern int snd_ctl_elem_add_integer(snd_ctl_t * ctl,
                                    const snd_ctl_elem_id_t * id,
                                     unsigned int count, long int
imin,
                                          long int imax, long int
istep);
extern void snd_ctl_elem_id_clear(snd_ctl_elem_id_t * obj);
extern void snd_ctl_elem_id_copy(snd_ctl_elem_id_t * dst,
                                 const snd_ctl_elem_id_t * src);
extern void snd_ctl_elem_id_free(snd_ctl_elem_id_t * obj);
```

```
extern     unsigned     int     snd_ctl_elem_id_get_device(const
snd_ctl_elem_id_t *
                                                obj);
extern     unsigned     int     snd_ctl_elem_id_get_index(const
snd_ctl_elem_id_t *
                                               obj);
extern snd_ctl_elem_iface_t snd_ctl_elem_id_get_interface(const
                                                          snd_ctl
_elem_id_t
                                                         * obj);
extern      const      char      *snd_ctl_elem_id_get_name(const
snd_ctl_elem_id_t * obj);
extern     unsigned     int     snd_ctl_elem_id_get_numid(const
snd_ctl_elem_id_t *
                                               obj);
extern    unsigned    int    snd_ctl_elem_id_get_subdevice(const
snd_ctl_elem_id_t *
                                                  obj);
extern int snd_ctl_elem_id_malloc(snd_ctl_elem_id_t * *ptr);
extern void snd_ctl_elem_id_set_device(snd_ctl_elem_id_t * obj,
                                       unsigned int val);
extern void snd_ctl_elem_id_set_index(snd_ctl_elem_id_t * obj,
                                      unsigned int val);
extern  void  snd_ctl_elem_id_set_interface(snd_ctl_elem_id_t  *
obj,
                                            snd_ctl_elem_iface_t
val);
extern void snd_ctl_elem_id_set_name(snd_ctl_elem_id_t * obj,
                                     const char *val);
extern void snd_ctl_elem_id_set_numid(snd_ctl_elem_id_t * obj,
                                      unsigned int val);
extern  void  snd_ctl_elem_id_set_subdevice(snd_ctl_elem_id_t  *
obj,
                                            unsigned int val);
extern size_t snd_ctl_elem_id_sizeof(void);
extern  const  char  *snd_ctl_elem_iface_name(snd_ctl_elem_iface_t
iface);
extern int snd_ctl_elem_info(snd_ctl_t * ctl, snd_ctl_elem_info_t
* info);
extern void snd_ctl_elem_info_clear(snd_ctl_elem_info_t * obj);
extern void snd_ctl_elem_info_copy(snd_ctl_elem_info_t * dst,
                                   const snd_ctl_elem_info_t *
src);
extern void snd_ctl_elem_info_free(snd_ctl_elem_info_t * obj);
extern     unsigned     int     snd_ctl_elem_info_get_count(const
snd_ctl_elem_info_t *
                                                obj);
extern  void  snd_ctl_elem_info_get_id(const  snd_ctl_elem_info_t  *
obj,
                                       snd_ctl_elem_id_t * ptr);
extern const char *snd_ctl_elem_info_get_item_name(const
                                                   snd_ctl_elem_i
nfo_t *
                                                  obj);
extern     unsigned     int     snd_ctl_elem_info_get_items(const
snd_ctl_elem_info_t *
                                                obj);
extern       long       int       snd_ctl_elem_info_get_max(const
snd_ctl_elem_info_t * obj);
extern    long    long    int    snd_ctl_elem_info_get_max64(const
snd_ctl_elem_info_t
                                                 * obj);
extern       long       int       snd_ctl_elem_info_get_min(const
snd_ctl_elem_info_t * obj);
extern    long    long    int    snd_ctl_elem_info_get_min64(const
snd_ctl_elem_info_t
```

```
                                                    * obj);
extern      const      char      *snd_ctl_elem_info_get_name(const
snd_ctl_elem_info_t *
                                                  obj);
extern     unsigned     int     snd_ctl_elem_info_get_numid(const
snd_ctl_elem_info_t *
                                                   obj);
extern       long       int       snd_ctl_elem_info_get_step(const
snd_ctl_elem_info_t *
                                               obj);
extern    long    long    int    snd_ctl_elem_info_get_step64(const
snd_ctl_elem_info_t
                                                    * obj);
extern snd_ctl_elem_type_t snd_ctl_elem_info_get_type(const
                                                         snd_ctl_ele
m_info_t *
                                                        obj);
extern           int           snd_ctl_elem_info_is_inactive(const
snd_ctl_elem_info_t * obj);
extern  int  snd_ctl_elem_info_is_locked(const  snd_ctl_elem_info_t
* obj);
extern           int           snd_ctl_elem_info_is_readable(const
snd_ctl_elem_info_t * obj);
extern  int  snd_ctl_elem_info_is_user(const  snd_ctl_elem_info_t  *
obj);
extern           int           snd_ctl_elem_info_is_volatile(const
snd_ctl_elem_info_t * obj);
extern           int           snd_ctl_elem_info_is_writable(const
snd_ctl_elem_info_t * obj);
extern int snd_ctl_elem_info_malloc(snd_ctl_elem_info_t * *ptr);
extern void snd_ctl_elem_info_set_id(snd_ctl_elem_info_t * obj,
                                        const snd_ctl_elem_id_t *
ptr);
extern void snd_ctl_elem_info_set_item(snd_ctl_elem_info_t * obj,
                                       unsigned int val);
extern size_t snd_ctl_elem_info_sizeof(void);
extern int snd_ctl_elem_list(snd_ctl_t * ctl, snd_ctl_elem_list_t
* list);
extern  int  snd_ctl_elem_list_alloc_space(snd_ctl_elem_list_t  *
obj,
                                         unsigned int entries);
extern void snd_ctl_elem_list_clear(snd_ctl_elem_list_t * obj);
extern void snd_ctl_elem_list_copy(snd_ctl_elem_list_t * dst,
                                      const snd_ctl_elem_list_t *
src);
extern void snd_ctl_elem_list_free(snd_ctl_elem_list_t * obj);
extern  void  snd_ctl_elem_list_free_space(snd_ctl_elem_list_t  *
obj);
extern     unsigned     int     snd_ctl_elem_list_get_count(const
snd_ctl_elem_list_t *
                                                   obj);
extern  void  snd_ctl_elem_list_get_id(const  snd_ctl_elem_list_t  *
obj,
                                    unsigned int idx,
                                    snd_ctl_elem_id_t * ptr);
extern      const      char      *snd_ctl_elem_list_get_name(const
snd_ctl_elem_list_t *
                                                    obj, unsigned int
idx);
extern     unsigned     int      snd_ctl_elem_list_get_used(const
snd_ctl_elem_list_t *
                                                  obj);
extern int snd_ctl_elem_list_malloc(snd_ctl_elem_list_t * *ptr);
extern  void  snd_ctl_elem_list_set_offset(snd_ctl_elem_list_t  *
obj,
                                         unsigned int val);
```

```
extern size_t snd_ctl_elem_list_sizeof(void);
extern int snd_ctl_elem_read(snd_ctl_t * ctl,
                             snd_ctl_elem_value_t * value);
extern int snd_ctl_elem_remove(snd_ctl_t * ctl, snd_ctl_elem_id_t
* id);
extern   const   char   *snd_ctl_elem_type_name(snd_ctl_elem_type_t
type);
extern void snd_ctl_elem_value_clear(snd_ctl_elem_value_t * obj);
extern void snd_ctl_elem_value_copy(snd_ctl_elem_value_t * dst,
                                    const snd_ctl_elem_value_t *
src);
extern void snd_ctl_elem_value_free(snd_ctl_elem_value_t * obj);
extern          int          snd_ctl_elem_value_get_boolean(const
snd_ctl_elem_value_t * obj,
                                          unsigned int idx);
extern     unsigned     char     snd_ctl_elem_value_get_byte(const
snd_ctl_elem_value_t
                                                 * obj, unsigned
int idx);
extern     const     void     *snd_ctl_elem_value_get_bytes(const
snd_ctl_elem_value_t
                                                  * obj);
extern unsigned int snd_ctl_elem_value_get_enumerated(const
                                                      snd_ctl_ele
m_value_t
                                                      * obj,
                                                        unsigned
int idx);
extern  void  snd_ctl_elem_value_get_id(const  snd_ctl_elem_value_t
* obj,
                                        snd_ctl_elem_id_t * ptr);
extern          void          snd_ctl_elem_value_get_iec958(const
snd_ctl_elem_value_t * obj,
                                                    snd_aes_iec958_t *
ptr);
extern     long     int     snd_ctl_elem_value_get_integer(const
snd_ctl_elem_value_t *
                                                     obj, unsigned int
idx);
extern long long int snd_ctl_elem_value_get_integer64(const
                                                      snd_ctl_ele
m_value_t
                                                      * obj,
                                                        unsigned
int idx);
extern   int   snd_ctl_elem_value_malloc(snd_ctl_elem_value_t   *
*ptr);
extern void snd_ctl_elem_value_set_boolean(snd_ctl_elem_value_t *
obj,
                                           unsigned int idx, long
int val);
extern  void  snd_ctl_elem_value_set_byte(snd_ctl_elem_value_t  *
obj,
                                          unsigned int idx,
                                          unsigned char val);
extern                                                      void
snd_ctl_elem_value_set_enumerated(snd_ctl_elem_value_t * obj,
                                              unsigned int idx,
                                              unsigned int val);
extern void snd_ctl_elem_value_set_id(snd_ctl_elem_value_t * obj,
                                      const snd_ctl_elem_id_t *
ptr);
extern  void  snd_ctl_elem_value_set_iec958(snd_ctl_elem_value_t  *
obj,
                                           const snd_aes_iec958_t
* ptr);
```

```
extern void snd_ctl_elem_value_set_integer(snd_ctl_elem_value_t *
obj,
                                           unsigned int idx, long
int val);
extern void snd_ctl_elem_value_set_integer64(snd_ctl_elem_value_t
* obj,
                                             unsigned int idx,
                                             long long int val);
extern size_t snd_ctl_elem_value_sizeof(void);
extern int snd_ctl_elem_write(snd_ctl_t * ctl,
                              snd_ctl_elem_value_t * value);
extern void snd_ctl_event_clear(snd_ctl_event_t * obj);
extern void snd_ctl_event_copy(snd_ctl_event_t * dst,
                               const snd_ctl_event_t * src);
extern  void  snd_ctl_event_elem_get_id(const  snd_ctl_event_t  *
obj,
                                        snd_ctl_elem_id_t * ptr);
extern     unsigned     int     snd_ctl_event_elem_get_mask(const
snd_ctl_event_t *
                                                        obj);
extern void snd_ctl_event_free(snd_ctl_event_t * obj);
extern int snd_ctl_event_malloc(snd_ctl_event_t * *ptr);
extern size_t snd_ctl_event_sizeof(void);
extern int snd_ctl_hwdep_info(snd_ctl_t * ctl, snd_hwdep_info_t *
info);
extern   int   snd_ctl_hwdep_next_device(snd_ctl_t   *   ctl,   int
*device);
extern const char *snd_ctl_name(snd_ctl_t * ctl);
extern int snd_ctl_nonblock(snd_ctl_t * ctl, int nonblock);
extern int snd_ctl_open(snd_ctl_t * *ctl, const char *name, int
mode);
extern  int  snd_ctl_pcm_info(snd_ctl_t  *  ctl,  snd_pcm_info_t  *
info);
extern int snd_ctl_pcm_next_device(snd_ctl_t * ctl, int *device);
extern  int  snd_ctl_poll_descriptors(snd_ctl_t  *  ctl,  struct
pollfd *pfds,
                                      unsigned int space);
extern int snd_ctl_poll_descriptors_count(snd_ctl_t * ctl);
extern int snd_ctl_rawmidi_info(snd_ctl_t * ctl,
                                snd_rawmidi_info_t * info);
extern  int  snd_ctl_rawmidi_next_device(snd_ctl_t  *  ctl,  int
*device);
extern  int  snd_ctl_read(snd_ctl_t  *  ctl,  snd_ctl_event_t  *
event);
extern   int   snd_ctl_subscribe_events(snd_ctl_t   *   ctl,   int
subscribe);
extern int snd_hctl_close(snd_hctl_t * hctl);
extern       void       *snd_hctl_elem_get_callback_private(const
snd_hctl_elem_t *
                                                        obj);
extern void snd_hctl_elem_get_id(const snd_hctl_elem_t * obj,
                                 snd_ctl_elem_id_t * ptr);
extern int snd_hctl_elem_info(snd_hctl_elem_t * elem,
                              snd_ctl_elem_info_t * info);
extern   snd_hctl_elem_t   *snd_hctl_elem_next(snd_hctl_elem_t   *
elem);
extern   snd_hctl_elem_t   *snd_hctl_elem_prev(snd_hctl_elem_t   *
elem);
extern int snd_hctl_elem_read(snd_hctl_elem_t * elem,
                              snd_ctl_elem_value_t * value);
extern void snd_hctl_elem_set_callback(snd_hctl_elem_t * obj,
                                       snd_hctl_elem_callback_t
val);
extern  void  snd_hctl_elem_set_callback_private(snd_hctl_elem_t  *
obj,
                                                 void *val);
```

```
extern int snd_hctl_elem_write(snd_hctl_elem_t * elem,
                               snd_ctl_elem_value_t * value);
extern snd_hctl_elem_t *snd_hctl_find_elem(snd_hctl_t * hctl,
                                                        const
snd_ctl_elem_id_t * id);
extern snd_hctl_elem_t *snd_hctl_first_elem(snd_hctl_t * hctl);
extern int snd_hctl_free(snd_hctl_t * hctl);
extern void *snd_hctl_get_callback_private(snd_hctl_t * hctl);
extern unsigned int snd_hctl_get_count(snd_hctl_t * hctl);
extern int snd_hctl_handle_events(snd_hctl_t * hctl);
extern snd_hctl_elem_t *snd_hctl_last_elem(snd_hctl_t * hctl);
extern int snd_hctl_load(snd_hctl_t * hctl);
extern int snd_hctl_nonblock(snd_hctl_t * hctl, int nonblock);
extern  int  snd_hctl_open(snd_hctl_t  *  *hctl,  const  char  *name,
int mode);
extern void snd_hctl_set_callback(snd_hctl_t * hctl,
                                  snd_hctl_callback_t callback);
extern void snd_hctl_set_callback_private(snd_hctl_t * hctl, void
*data);
extern int snd_hctl_wait(snd_hctl_t * hctl, int timeout);
```

6.2.3 alsa/control_external.h

```
#define SND_CTL_EXT_VERSION      ((SND_CTL_EXT_VERSION_MAJOR<<16)
| (SND_CTL_EXT_VERSION_MINOR<<8) | (SND_CTL_EXT_VERSION_TINY))
#define SND_CTL_EXT_KEY_NOT_FOUND       (snd_ctl_ext_key_t)(-1)
#define SND_CTL_EXT_VERSION_MINOR       0
#define SND_CTL_EXT_VERSION_TINY        0
#define SND_CTL_EXT_VERSION_MAJOR       1

typedef struct snd_ctl_ext_callback {
    void (*close) (void);
    int (*elem_count) (void);
    int (*elem_list) (void);
     snd_ctl_ext_key_t(*find_elem) (void);
    void (*free_key) (void);
    int (*get_attribute) (void);
    int (*get_integer_info) (void);
    int (*get_integer64_info) (void);
    int (*get_enumerated_info) (void);
    int (*get_enumerated_name) (void);
    int (*read_integer) (void);
    int (*read_integer64) (void);
    int (*read_enumerated) (void);
    int (*read_bytes) (void);
    int (*read_iec958) (void);
    int (*write_integer) (void);
    int (*write_integer64) (void);
    int (*write_enumerated) (void);
    int (*write_bytes) (void);
    int (*write_iec958) (void);
    void (*subscribe_events) (void);
    int (*read_event) (void);
    int (*poll_descriptors_count) (void);
    int (*poll_descriptors) (void);
    int (*poll_revents) (void);
} snd_ctl_ext_callback_t;
typedef long unsigned int snd_ctl_ext_key_t;
typedef struct snd_ctl_ext {
    unsigned int version;
    int card_idx;
    char id[16];
    char driver[16];
    char name[32];
```

```
    char longname[80];
    char mixername[80];
    int poll_fd;
    const snd_ctl_ext_callback_t *callback;
    void *private_data;
    snd_ctl_t *handle;
    int nonblock;
    int subscribed;
} snd_ctl_ext_t;
```

6.2.4 alsa/error.h

```
#define SND_ERROR_INCOMPATIBLE_VERSION  (SND_ERROR_BEGIN+0)
#define SND_ERROR_ALISP_NIL     (SND_ERROR_BEGIN+1)
#define SND_ERROR_BEGIN 500000

typedef void (*snd_lib_error_handler_t) (const char *, int, const
char *,
                                         int, const char *, ...);
extern    int    snd_lib_error_set_handler(snd_lib_error_handler_t
handler);
extern const char *snd_strerror(int errnum);
```

6.2.5 alsa/global.h

```
typedef struct _snd_async_handler snd_async_handler_t;
typedef void (*snd_async_callback_t) (snd_async_handler_t *);
typedef struct timespec snd_htimestamp_t;
typedef struct timeval snd_timestamp_t;
extern const char *snd_asoundlib_version(void);
extern  int  snd_async_add_handler(snd_async_handler_t  *  *handler,
int fd,
                                   snd_async_callback_t callback,
                                   void *private_data);
extern int snd_async_del_handler(snd_async_handler_t * handler);
extern                                                        void
*snd_async_handler_get_callback_private(snd_async_handler_t *
                                                      handler);
```

6.2.6 alsa/hwdep.h

```
#define SND_HWDEP_OPEN_NONBLOCK (O_NONBLOCK)
#define SND_HWDEP_OPEN_READ     (O_RDONLY)
#define SND_HWDEP_OPEN_DUPLEX   (O_RDWR)
#define SND_HWDEP_OPEN_WRITE    (O_WRONLY)

typedef struct sndrv_hwdep_dsp_image snd_hwdep_dsp_image_t;
typedef struct sndrv_hwdep_dsp_status snd_hwdep_dsp_status_t;
typedef enum _snd_hwdep_iface {
    SND_HWDEP_IFACE_OPL2,
    SND_HWDEP_IFACE_OPL3 = 1,
    SND_HWDEP_IFACE_OPL4 = 2,
    SND_HWDEP_IFACE_SB16CSP = 3,
    SND_HWDEP_IFACE_EMU10K1 = 4,
    SND_HWDEP_IFACE_YSS225 = 5,
    SND_HWDEP_IFACE_ICS2115 = 6,
    SND_HWDEP_IFACE_SSCAPE = 7,
    SND_HWDEP_IFACE_VX = 8,
    SND_HWDEP_IFACE_MIXART = 9,
    SND_HWDEP_IFACE_USX2Y = 10,
    SND_HWDEP_IFACE_EMUX_WAVETABLE = 11,
```

```
    SND_HWDEP_IFACE_BLUETOOTH = 12,
    SND_HWDEP_IFACE_USX2Y_PCM = 13,
    SND_HWDEP_IFACE_PCXHR = 14,
    SND_HWDEP_IFACE_SB_RC = 15,
    SND_HWDEP_IFACE_LAST = 15
} snd_hwdep_iface_t;
typedef struct sndrv_hwdep_info snd_hwdep_info_t;
typedef struct _snd_hwdep snd_hwdep_t;
typedef enum _snd_hwdep_type {
    SND_HWDEP_TYPE_HW,
    SND_HWDEP_TYPE_SHM = 1,
    SND_HWDEP_TYPE_INET = 2
} snd_hwdep_type_t;
extern int snd_hwdep_close(snd_hwdep_t * hwdep);
extern void snd_hwdep_dsp_image_copy(snd_hwdep_dsp_image_t * dst,
                                     const snd_hwdep_dsp_image_t
* src);
extern   void   snd_hwdep_dsp_image_free(snd_hwdep_dsp_image_t   *
obj);
extern const void *snd_hwdep_dsp_image_get_image(const
                                                 snd_hwdep_dsp_im
age_t *
                                                 obj);
extern unsigned int snd_hwdep_dsp_image_get_index(const
                                                  snd_hwdep_dsp_i
mage_t *
                                                  obj);
extern         size_t         snd_hwdep_dsp_image_get_length(const
snd_hwdep_dsp_image_t *
                                              obj);
extern     const     char     *snd_hwdep_dsp_image_get_name(const
snd_hwdep_dsp_image_t
                                                    * obj);
extern   int   snd_hwdep_dsp_image_malloc(snd_hwdep_dsp_image_t   *
*ptr);
extern void snd_hwdep_dsp_image_set_image(snd_hwdep_dsp_image_t *
obj,
                                          void *buffer);
extern void snd_hwdep_dsp_image_set_index(snd_hwdep_dsp_image_t *
obj,
                                          unsigned int _index);
extern  void  snd_hwdep_dsp_image_set_length(snd_hwdep_dsp_image_t
* obj,
                                           size_t length);
extern  void  snd_hwdep_dsp_image_set_name(snd_hwdep_dsp_image_t  *
obj,
                                         const char *name);
extern size_t snd_hwdep_dsp_image_sizeof(void);
extern int snd_hwdep_dsp_load(snd_hwdep_t * hwdep,
                              snd_hwdep_dsp_image_t * block);
extern int snd_hwdep_dsp_status(snd_hwdep_t * hwdep,
                                snd_hwdep_dsp_status_t * status);
extern  void  snd_hwdep_dsp_status_copy(snd_hwdep_dsp_status_t  *
dst,
                                                                 const
snd_hwdep_dsp_status_t * src);
extern  void  snd_hwdep_dsp_status_free(snd_hwdep_dsp_status_t  *
obj);
extern unsigned int snd_hwdep_dsp_status_get_chip_ready(const
                                                        snd_hwdep
_dsp_status_t
                                                        * obj);
extern unsigned int snd_hwdep_dsp_status_get_dsp_loaded(const
                                                        snd_hwdep
_dsp_status_t
                                                        * obj);
```

```
extern     const     char     *snd_hwdep_dsp_status_get_id(const
snd_hwdep_dsp_status_t
                                                  * obj);
extern unsigned int snd_hwdep_dsp_status_get_num_dsps(const
                                                    snd_hwdep_d
sp_status_t
                                                   * obj);
extern unsigned int snd_hwdep_dsp_status_get_version(const
                                                  snd_hwdep_ds
p_status_t
                                                  * obj);
extern  int  snd_hwdep_dsp_status_malloc(snd_hwdep_dsp_status_t  *
*ptr);
extern size_t snd_hwdep_dsp_status_sizeof(void);
extern int snd_hwdep_info(snd_hwdep_t * hwdep, snd_hwdep_info_t *
info);
extern void snd_hwdep_info_copy(snd_hwdep_info_t * dst,
                                const snd_hwdep_info_t * src);
extern void snd_hwdep_info_free(snd_hwdep_info_t * obj);
extern int snd_hwdep_info_get_card(const snd_hwdep_info_t * obj);
extern      unsigned      int      snd_hwdep_info_get_device(const
snd_hwdep_info_t *
                                               obj);
extern const char *snd_hwdep_info_get_id(const snd_hwdep_info_t *
obj);
extern      snd_hwdep_iface_t      snd_hwdep_info_get_iface(const
snd_hwdep_info_t *
                                                    obj);
extern const char *snd_hwdep_info_get_name(const snd_hwdep_info_t
* obj);
extern int snd_hwdep_info_malloc(snd_hwdep_info_t * *ptr);
extern void snd_hwdep_info_set_device(snd_hwdep_info_t * obj,
                                      unsigned int val);
extern size_t snd_hwdep_info_sizeof(void);
extern  int  snd_hwdep_ioctl(snd_hwdep_t  *  hwdep,  unsigned  int
request,
                             void *arg);
extern int snd_hwdep_open(snd_hwdep_t * *hwdep, const char *name,
                          int mode);
extern int snd_hwdep_poll_descriptors(snd_hwdep_t * hwdep,
                                      struct pollfd *pfds,
                                      unsigned int space);
extern ssize_t snd_hwdep_read(snd_hwdep_t * hwdep, void *buffer,
                              size_t size);
extern  ssize_t  snd_hwdep_write(snd_hwdep_t  *  hwdep,  const  void
*buffer,
                                 size_t size);
```

6.2.7 alsa/iatomic.h

```
#define atomic_set(v,i) (((v)->counter) = (i))
#define atomic_read(v)  ((v)->counter)
#define ATOMIC_INIT(i)  { (i) }

typedef struct {
    unsigned int begin;
    unsigned int end;
} snd_atomic_write_t;
typedef struct {
    const volatile snd_atomic_write_t *write;
    unsigned int end;
} snd_atomic_read_t;
```

6.2.8 alsa/input.h

```
typedef struct _snd_input snd_input_t;
extern int snd_input_buffer_open(snd_input_t * *inputp, const
char *buffer,
                                 ssize_t size);
extern int snd_input_close(snd_input_t * input);
extern int snd_input_stdio_attach(snd_input_t * *inputp, FILE *
fp,
                                  int _close);
extern int snd_input_stdio_open(snd_input_t * *inputp, const char
*file,
                                const char *mode);
```

6.2.9 alsa/mixer.h

```
typedef struct _snd_mixer snd_mixer_t;
typedef struct _snd_mixer_elem snd_mixer_elem_t;
typedef enum _snd_mixer_elem_type {
    SND_MIXER_ELEM_SIMPLE,
    SND_MIXER_ELEM_LAST
} snd_mixer_elem_type_t;
typedef struct _snd_mixer_class snd_mixer_class_t;
typedef int (*snd_mixer_compare_t) (const snd_mixer_elem_t *,
                                    const snd_mixer_elem_t *);
typedef int (*snd_mixer_elem_callback_t) (snd_mixer_elem_t *,
                                          unsigned int);
typedef int (*snd_mixer_callback_t) (snd_mixer_t *, unsigned int,
                                     snd_mixer_elem_t *);
typedef int (*snd_mixer_event_t) (snd_mixer_class_t *, unsigned
int,
                                                  snd_hctl_elem_t *,
snd_mixer_elem_t *);
typedef enum _snd_mixer_selem_channel_id {
    SND_MIXER_SCHN_UNKNOWN = -1,
    SND_MIXER_SCHN_FRONT_LEFT,
    SND_MIXER_SCHN_FRONT_RIGHT = 1,
    SND_MIXER_SCHN_REAR_LEFT = 2,
    SND_MIXER_SCHN_REAR_RIGHT = 3,
    SND_MIXER_SCHN_FRONT_CENTER = 4,
    SND_MIXER_SCHN_WOOFER = 5,
    SND_MIXER_SCHN_SIDE_LEFT = 6,
    SND_MIXER_SCHN_SIDE_RIGHT = 7,
    SND_MIXER_SCHN_REAR_CENTER = 8,
    SND_MIXER_SCHN_LAST = 31,
    SND_MIXER_SCHN_MONO
} snd_mixer_selem_channel_id_t;
typedef struct _snd_mixer_selem_id snd_mixer_selem_id_t;
enum snd_mixer_selem_regopt_abstract {
    SND_MIXER_SABSTRACT_NONE,
    SND_MIXER_SABSTRACT_BASIC = 1
};
struct snd_mixer_selem_regopt {
    int ver;
    enum snd_mixer_selem_regopt_abstract abstract;
    const char *device;
    snd_pcm_t *playback_pcm;
    snd_pcm_t *capture_pcm;
};
extern int snd_mixer_attach(snd_mixer_t * mixer, const char
*name);
extern int snd_mixer_close(snd_mixer_t * mixer);
extern int snd_mixer_detach(snd_mixer_t * mixer, const char
```

```
*name);
extern       void       *snd_mixer_elem_get_callback_private(const
snd_mixer_elem_t *
                                                    obj);
extern     snd_mixer_elem_type_t     snd_mixer_elem_get_type(const
snd_mixer_elem_t
                                                          * obj);
extern  snd_mixer_elem_t  *snd_mixer_elem_next(snd_mixer_elem_t  *
elem);
extern  snd_mixer_elem_t  *snd_mixer_elem_prev(snd_mixer_elem_t  *
elem);
extern void snd_mixer_elem_set_callback(snd_mixer_elem_t * obj,
                                        snd_mixer_elem_callback_t
val);
extern  void  snd_mixer_elem_set_callback_private(snd_mixer_elem_t
* obj,
                                                   void *val);
extern   snd_mixer_elem_t   *snd_mixer_find_selem(snd_mixer_t   *
mixer,
                                                                const
snd_mixer_selem_id_t *
                                                 id);
extern   snd_mixer_elem_t   *snd_mixer_first_elem(snd_mixer_t   *
mixer);
extern void snd_mixer_free(snd_mixer_t * mixer);
extern  void  *snd_mixer_get_callback_private(const  snd_mixer_t  *
obj);
extern unsigned int snd_mixer_get_count(const snd_mixer_t * obj);
extern int snd_mixer_handle_events(snd_mixer_t * mixer);
extern   snd_mixer_elem_t    *snd_mixer_last_elem(snd_mixer_t    *
mixer);
extern int snd_mixer_load(snd_mixer_t * mixer);
extern int snd_mixer_open(snd_mixer_t * *mixer, int mode);
extern int snd_mixer_poll_descriptors(snd_mixer_t * mixer,
                                      struct pollfd *pfds,
                                      unsigned int space);
extern int snd_mixer_poll_descriptors_count(snd_mixer_t * mixer);
extern   int   snd_mixer_poll_descriptors_revents(snd_mixer_t   *
mixer,
                                                        struct pollfd
*pfds,
                                                 unsigned int nfds,
                                                   short unsigned int
*revents);
extern const char
        *snd_mixer_selem_channel_name(snd_mixer_selem_channel_id_t
channel);
extern  int  snd_mixer_selem_get_capture_group(snd_mixer_elem_t  *
elem);
extern  int  snd_mixer_selem_get_capture_switch(snd_mixer_elem_t  *
elem,
                                                snd_mixer_selem_cha
nnel_id_t
                                                        channel, int
*value);
extern  int  snd_mixer_selem_get_capture_volume(snd_mixer_elem_t  *
elem,
                                                snd_mixer_selem_cha
nnel_id_t
                                                  channel, long int
*value);
extern                                                          int
snd_mixer_selem_get_capture_volume_range(snd_mixer_elem_t *
                                                              elem, long
int *min,
                                                                long int
```

```
*max);
extern int snd_mixer_selem_get_enum_item(snd_mixer_elem_t * elem,
                                         snd_mixer_selem_channel_
id_t
                                                 channel, unsigned int
*idxp);
extern  int  snd_mixer_selem_get_enum_item_name(snd_mixer_elem_t  *
elem,
                                                unsigned int idx,
                                                size_t maxlen, char
*str);
extern   int   snd_mixer_selem_get_enum_items(snd_mixer_elem_t   *
elem);
extern void snd_mixer_selem_get_id(snd_mixer_elem_t * element,
                                   snd_mixer_selem_id_t * id);
extern  unsigned  int  snd_mixer_selem_get_index(snd_mixer_elem_t  *
elem);
extern  const  char  *snd_mixer_selem_get_name(snd_mixer_elem_t  *
elem);
extern int snd_mixer_selem_get_playback_switch(snd_mixer_elem_t *
elem,
                                               snd_mixer_selem_ch
annel_id_t
                                                     channel, int
*value);
extern int snd_mixer_selem_get_playback_volume(snd_mixer_elem_t *
elem,
                                               snd_mixer_selem_ch
annel_id_t
                                                channel, long int
*value);
extern                                                          int
snd_mixer_selem_get_playback_volume_range(snd_mixer_elem_t *
                                                       elem, long
int *min,
                                                         long int
*max);
extern int snd_mixer_selem_has_capture_channel(snd_mixer_elem_t *
obj,
                                               snd_mixer_selem_ch
annel_id_t
                                              channel);
extern  int  snd_mixer_selem_has_capture_switch(snd_mixer_elem_t  *
elem);
extern                                                          int
snd_mixer_selem_has_capture_switch_exclusive(snd_mixer_elem_t *
                                                        elem);
extern                                                          int
snd_mixer_selem_has_capture_switch_joined(snd_mixer_elem_t *
                                                     elem);
extern  int  snd_mixer_selem_has_capture_volume(snd_mixer_elem_t  *
elem);
extern                                                          int
snd_mixer_selem_has_capture_volume_joined(snd_mixer_elem_t *
                                                     elem);
extern  int  snd_mixer_selem_has_common_switch(snd_mixer_elem_t  *
elem);
extern  int  snd_mixer_selem_has_common_volume(snd_mixer_elem_t  *
elem);
extern  int  snd_mixer_selem_has_playback_channel(snd_mixer_elem_t
* obj,
                                                  snd_mixer_selem_c
hannel_id_t
                                                 channel);
extern int snd_mixer_selem_has_playback_switch(snd_mixer_elem_t *
elem);
```

```
extern                                                          int
snd_mixer_selem_has_playback_switch_joined(snd_mixer_elem_t *
                                                              elem);
extern int snd_mixer_selem_has_playback_volume(snd_mixer_elem_t *
elem);
extern                                                          int
snd_mixer_selem_has_playback_volume_joined(snd_mixer_elem_t *
                                                              elem);
extern void snd_mixer_selem_id_copy(snd_mixer_selem_id_t * dst,
                                    const snd_mixer_selem_id_t *
src);
extern void snd_mixer_selem_id_free(snd_mixer_selem_id_t * obj);
extern     unsigned     int     snd_mixer_selem_id_get_index(const
snd_mixer_selem_id_t
                                                       * obj);
extern     const     char     *snd_mixer_selem_id_get_name(const
snd_mixer_selem_id_t *
                                                    obj);
extern   int   snd_mixer_selem_id_malloc(snd_mixer_selem_id_t   *
*ptr);
extern  void  snd_mixer_selem_id_set_index(snd_mixer_selem_id_t  *
obj,
                                           unsigned int val);
extern  void  snd_mixer_selem_id_set_name(snd_mixer_selem_id_t  *
obj,
                                          const char *val);
extern size_t snd_mixer_selem_id_sizeof(void);
extern int snd_mixer_selem_is_active(snd_mixer_elem_t * elem);
extern   int   snd_mixer_selem_is_capture_mono(snd_mixer_elem_t   *
elem);
extern   int   snd_mixer_selem_is_enum_capture(snd_mixer_elem_t   *
elem);
extern  int  snd_mixer_selem_is_enum_playback(snd_mixer_elem_t  *
elem);
extern   int   snd_mixer_selem_is_enumerated(snd_mixer_elem_t   *
elem);
extern  int  snd_mixer_selem_is_playback_mono(snd_mixer_elem_t  *
elem);
extern int snd_mixer_selem_register(snd_mixer_t * mixer,
                                    struct snd_mixer_selem_regopt
*options,
                                   snd_mixer_class_t * *classp);
extern  int  snd_mixer_selem_set_capture_switch(snd_mixer_elem_t  *
elem,
                                                snd_mixer_selem_cha
nnel_id_t
                                                       channel, int
value);
extern                                                          int
snd_mixer_selem_set_capture_switch_all(snd_mixer_elem_t * elem,
                                                         int value);
extern  int  snd_mixer_selem_set_capture_volume(snd_mixer_elem_t  *
elem,
                                                snd_mixer_selem_cha
nnel_id_t
                                                  channel, long int
value);
extern                                                          int
snd_mixer_selem_set_capture_volume_all(snd_mixer_elem_t * elem,
                                                            long int
value);
extern                                                          int
snd_mixer_selem_set_capture_volume_range(snd_mixer_elem_t *
                                                          elem, long
int min,
                                                            long int
```

```
max);
extern int snd_mixer_selem_set_enum_item(snd_mixer_elem_t * elem,
                                         snd_mixer_selem_channel_
id_t
                                           channel, unsigned int
idx);
extern int snd_mixer_selem_set_playback_switch(snd_mixer_elem_t *
elem,
                                               snd_mixer_selem_ch
annel_id_t
                                                     channel, int
value);
extern                                                         int
snd_mixer_selem_set_playback_switch_all(snd_mixer_elem_t * elem,
                                                    int value);
extern int snd_mixer_selem_set_playback_volume(snd_mixer_elem_t *
elem,
                                               snd_mixer_selem_ch
annel_id_t
                                                channel, long int
value);
extern                                                         int
snd_mixer_selem_set_playback_volume_all(snd_mixer_elem_t * elem,
                                                         long int
value);
extern                                                         int
snd_mixer_selem_set_playback_volume_range(snd_mixer_elem_t *
                                                       elem, long
int min,
                                                         long int
max);
extern void snd_mixer_set_callback(snd_mixer_t * obj,
                                   snd_mixer_callback_t val);
extern  void  snd_mixer_set_callback_private(snd_mixer_t  *  obj,
void *val);
extern int snd_mixer_wait(snd_mixer_t * mixer, int timeout);
```

6.2.10 alsa/mixer_abst.h

```
#define sm_selem(x)     ((sm_selem_t *)((x)->private_data))
#define sm_selem_ops(x) ((sm_selem_t *)((x)->private_data))->ops
#define SM_CAP_GVOLUME  (1<<1)
#define SM_CAP_CSWITCH_JOIN     (1<<10)
#define SM_CAP_CSWITCH_EXCL     (1<<11)
#define SM_CAP_PENUM    (1<<12)
#define SM_CAP_CENUM    (1<<13)
#define SM_CAP_GSWITCH  (1<<2)
#define SM_CAP_PVOLUME  (1<<3)
#define SM_CAP_PVOLUME_JOIN     (1<<4)
#define SM_CAP_PSWITCH  (1<<5)
#define SM_CAP_PSWITCH_JOIN     (1<<6)
#define SM_CAP_CVOLUME  (1<<7)
#define SM_CAP_CVOLUME_JOIN     (1<<8)
#define SM_CAP_CSWITCH  (1<<9)
#define SM_OPS_IS_ACTIVE        0
#define SM_OPS_IS_MONO  1
#define SM_OPS_IS_CHANNEL       2
#define SM_OPS_IS_ENUMERATED    3
#define SM_OPS_IS_ENUMCNT       4

typedef struct _sm_class_basic {
    char *device;
    snd_ctl_t *ctl;
    snd_hctl_t *hctl;
```

```
    snd_ctl_card_info_t *info;
} sm_class_basic_t;
```

6.2.11 alsa/output.h

```
typedef struct _snd_output snd_output_t;
extern int snd_output_buffer_open(snd_output_t * *outputp);
extern  size_t  snd_output_buffer_string(snd_output_t  *  output,
char **buf);
extern int snd_output_close(snd_output_t * output);
extern int snd_output_putc(snd_output_t * output, int c);
extern  int  snd_output_puts(snd_output_t  *  output,  const  char
*str);
extern int snd_output_stdio_attach(snd_output_t * *outputp, FILE
* fp,
                                   int _close);
extern  int  snd_output_stdio_open(snd_output_t  *  *outputp,  const
char *file,
                                 const char *mode);
```

6.2.12 alsa/pcm.h

```
#define SND_PCM_NONBLOCK        0x0001
#define SND_PCM_ASYNC   0x0002

typedef struct sndrv_mask snd_pcm_access_mask_t;
typedef enum _snd_pcm_access {
    SND_PCM_ACCESS_MMAP_INTERLEAVED,
    SND_PCM_ACCESS_MMAP_NONINTERLEAVED = 1,
    SND_PCM_ACCESS_MMAP_COMPLEX = 2,
    SND_PCM_ACCESS_RW_INTERLEAVED = 3,
    SND_PCM_ACCESS_RW_NONINTERLEAVED = 4,
    SND_PCM_ACCESS_LAST = 4
} snd_pcm_access_t;
typedef struct _snd_pcm_channel_area {
    void *addr;
    unsigned int first;
    unsigned int step;
} snd_pcm_channel_area_t;
typedef enum _snd_pcm_class {
    SND_PCM_CLASS_GENERIC,
    SND_PCM_CLASS_MULTI = 1,
    SND_PCM_CLASS_MODEM = 2,
    SND_PCM_CLASS_DIGITIZER = 3,
    SND_PCM_CLASS_LAST = 3
} snd_pcm_class_t;
typedef struct sndrv_mask snd_pcm_format_mask_t;
typedef enum _snd_pcm_format {
    SND_PCM_FORMAT_UNKNOWN = -1,
    SND_PCM_FORMAT_S8,
    SND_PCM_FORMAT_U8 = 1,
    SND_PCM_FORMAT_S16_LE = 2,
    SND_PCM_FORMAT_S16_BE = 3,
    SND_PCM_FORMAT_U16_LE = 4,
    SND_PCM_FORMAT_U16_BE = 5,
    SND_PCM_FORMAT_S24_LE = 6,
    SND_PCM_FORMAT_S24_BE = 7,
    SND_PCM_FORMAT_U24_LE = 8,
    SND_PCM_FORMAT_U24_BE = 9,
    SND_PCM_FORMAT_S32_LE = 10,
    SND_PCM_FORMAT_S32_BE = 11,
    SND_PCM_FORMAT_U32_LE = 12,
    SND_PCM_FORMAT_U32_BE = 13,
```

```
    SND_PCM_FORMAT_FLOAT_LE = 14,
    SND_PCM_FORMAT_FLOAT_BE = 15,
    SND_PCM_FORMAT_FLOAT64_LE = 16,
    SND_PCM_FORMAT_FLOAT64_BE = 17,
    SND_PCM_FORMAT_IEC958_SUBFRAME_LE = 18,
    SND_PCM_FORMAT_IEC958_SUBFRAME_BE = 19,
    SND_PCM_FORMAT_MU_LAW = 20,
    SND_PCM_FORMAT_A_LAW = 21,
    SND_PCM_FORMAT_IMA_ADPCM = 22,
    SND_PCM_FORMAT_MPEG = 23,
    SND_PCM_FORMAT_GSM = 24,
    SND_PCM_FORMAT_SPECIAL = 31,
    SND_PCM_FORMAT_S24_3LE = 32,
    SND_PCM_FORMAT_S24_3BE = 33,
    SND_PCM_FORMAT_U24_3LE = 34,
    SND_PCM_FORMAT_U24_3BE = 35,
    SND_PCM_FORMAT_S20_3LE = 36,
    SND_PCM_FORMAT_S20_3BE = 37,
    SND_PCM_FORMAT_U20_3LE = 38,
    SND_PCM_FORMAT_U20_3BE = 39,
    SND_PCM_FORMAT_S18_3LE = 40,
    SND_PCM_FORMAT_S18_3BE = 41,
    SND_PCM_FORMAT_U18_3LE = 42,
    SND_PCM_FORMAT_U18_3BE = 43,
    SND_PCM_FORMAT_LAST = 43,
    SND_PCM_FORMAT_S16 = 2,
    SND_PCM_FORMAT_U16 = 4,
    SND_PCM_FORMAT_S24 = 6,
    SND_PCM_FORMAT_U24 = 8,
    SND_PCM_FORMAT_S32 = 10,
    SND_PCM_FORMAT_U32 = 12,
    SND_PCM_FORMAT_FLOAT = 14,
    SND_PCM_FORMAT_FLOAT64 = 16,
    SND_PCM_FORMAT_IEC958_SUBFRAME = 18
} snd_pcm_format_t;
typedef struct _snd_pcm_hook snd_pcm_hook_t;
typedef int (*snd_pcm_hook_func_t) (snd_pcm_hook_t *);
typedef enum _snd_pcm_hook_type {
    SND_PCM_HOOK_TYPE_HW_PARAMS,
    SND_PCM_HOOK_TYPE_HW_FREE = 1,
    SND_PCM_HOOK_TYPE_CLOSE = 2,
    SND_PCM_HOOK_TYPE_LAST = 2
} snd_pcm_hook_type_t;
typedef struct sndrv_pcm_hw_params snd_pcm_hw_params_t;
typedef struct sndrv_pcm_info snd_pcm_info_t;
typedef struct _snd_pcm_scope_ops {
    int (*enable) (void);
    void (*disable) (void);
    void (*start) (void);
    void (*stop) (void);
    void (*update) (void);
    void (*reset) (void);
    void (*close) (void);
} snd_pcm_scope_ops_t;
typedef struct _snd_pcm_scope snd_pcm_scope_t;
typedef long int snd_pcm_sframes_t;
typedef enum _snd_pcm_start {
    SND_PCM_START_DATA,
    SND_PCM_START_EXPLICIT = 1,
    SND_PCM_START_LAST = 1
} snd_pcm_start_t;
typedef enum _snd_pcm_state {
    SND_PCM_STATE_OPEN,
    SND_PCM_STATE_SETUP = 1,
    SND_PCM_STATE_PREPARED = 2,
    SND_PCM_STATE_RUNNING = 3,
```

```
    SND_PCM_STATE_XRUN = 4,
    SND_PCM_STATE_DRAINING = 5,
    SND_PCM_STATE_PAUSED = 6,
    SND_PCM_STATE_SUSPENDED = 7,
    SND_PCM_STATE_DISCONNECTED = 8,
    SND_PCM_STATE_LAST = 8
} snd_pcm_state_t;
typedef struct sndrv_pcm_status snd_pcm_status_t;
typedef enum _snd_pcm_stream {
    SND_PCM_STREAM_PLAYBACK,
    SND_PCM_STREAM_CAPTURE = 1,
    SND_PCM_STREAM_LAST = 1
} snd_pcm_stream_t;
typedef enum _snd_pcm_subclass {
    SND_PCM_SUBCLASS_GENERIC_MIX,
    SND_PCM_SUBCLASS_MULTI_MIX = 1,
    SND_PCM_SUBCLASS_LAST = 1
} snd_pcm_subclass_t;
typedef struct sndrv_mask snd_pcm_subformat_mask_t;
typedef enum _snd_pcm_subformat {
    SND_PCM_SUBFORMAT_STD,
    SND_PCM_SUBFORMAT_LAST
} snd_pcm_subformat_t;
typedef struct sndrv_pcm_sw_params snd_pcm_sw_params_t;
typedef union _snd_pcm_sync_id {
    unsigned char id[16];
    short unsigned int id16[8];
    unsigned int id32[4];
} snd_pcm_sync_id_t;
typedef struct _snd_pcm snd_pcm_t;
typedef enum _snd_pcm_tstamp {
    SND_PCM_TSTAMP_NONE,
    SND_PCM_TSTAMP_MMAP = 1,
    SND_PCM_TSTAMP_LAST = 1
} snd_pcm_tstamp_t;
typedef enum _snd_pcm_type {
    SND_PCM_TYPE_HW,
    SND_PCM_TYPE_HOOKS = 1,
    SND_PCM_TYPE_MULTI = 2,
    SND_PCM_TYPE_FILE = 3,
    SND_PCM_TYPE_NULL = 4,
    SND_PCM_TYPE_SHM = 5,
    SND_PCM_TYPE_INET = 6,
    SND_PCM_TYPE_COPY = 7,
    SND_PCM_TYPE_LINEAR = 8,
    SND_PCM_TYPE_ALAW = 9,
    SND_PCM_TYPE_MULAW = 10,
    SND_PCM_TYPE_ADPCM = 11,
    SND_PCM_TYPE_RATE = 12,
    SND_PCM_TYPE_ROUTE = 13,
    SND_PCM_TYPE_PLUG = 14,
    SND_PCM_TYPE_SHARE = 15,
    SND_PCM_TYPE_METER = 16,
    SND_PCM_TYPE_MIX = 17,
    SND_PCM_TYPE_DROUTE = 18,
    SND_PCM_TYPE_LBSERVER = 19,
    SND_PCM_TYPE_LINEAR_FLOAT = 20,
    SND_PCM_TYPE_LADSPA = 21,
    SND_PCM_TYPE_DMIX = 22,
    SND_PCM_TYPE_JACK = 23,
    SND_PCM_TYPE_DSNOOP = 24,
    SND_PCM_TYPE_DSHARE = 25,
    SND_PCM_TYPE_IEC958 = 26,
    SND_PCM_TYPE_SOFTVOL = 27,
    SND_PCM_TYPE_IOPLUG = 28,
    SND_PCM_TYPE_EXTPLUG = 29,
```

```
    SND_PCM_TYPE_LAST = 29
} snd_pcm_type_t;
typedef long unsigned int snd_pcm_uframes_t;
typedef enum _snd_pcm_xrun {
    SND_PCM_XRUN_NONE,
    SND_PCM_XRUN_STOP = 1,
    SND_PCM_XRUN_LAST = 1
} snd_pcm_xrun_t;
typedef enum _snd_spcm_duplex_type {
    SND_SPCM_DUPLEX_LIBERAL,
    SND_SPCM_DUPLEX_PEDANTIC = 1
} snd_spcm_duplex_type_t;
typedef enum _snd_spcm_latency {
    SND_SPCM_LATENCY_STANDARD,
    SND_SPCM_LATENCY_MEDIUM = 1,
    SND_SPCM_LATENCY_REALTIME = 2
} snd_spcm_latency_t;
typedef enum _snd_spcm_xrun_type {
    SND_SPCM_XRUN_IGNORE,
    SND_SPCM_XRUN_STOP = 1
} snd_spcm_xrun_type_t;
extern    int    snd_async_add_pcm_handler(snd_async_handler_t    *
*handler,
                                    snd_pcm_t * pcm,
                                            snd_async_callback_t
callback,
                                    void *private_data);
extern snd_pcm_t *snd_async_handler_get_pcm(snd_async_handler_t *
handler);
extern    void    snd_pcm_access_mask_any(snd_pcm_access_mask_t    *
mask);
extern void snd_pcm_access_mask_copy(snd_pcm_access_mask_t * dst,
                                     const snd_pcm_access_mask_t
* src);
extern    void    snd_pcm_access_mask_free(snd_pcm_access_mask_t    *
obj);
extern   int   snd_pcm_access_mask_malloc(snd_pcm_access_mask_t   *
*ptr);
extern    void    snd_pcm_access_mask_none(snd_pcm_access_mask_t    *
mask);
extern void snd_pcm_access_mask_set(snd_pcm_access_mask_t * mask,
                                    snd_pcm_access_t val);
extern size_t snd_pcm_access_mask_sizeof(void);
extern int snd_pcm_access_mask_test(const snd_pcm_access_mask_t *
mask,
                                    snd_pcm_access_t val);
extern const char *snd_pcm_access_name(snd_pcm_access_t _access);
extern   int   snd_pcm_area_copy(const   snd_pcm_channel_area_t   *
dst_channel,
                                 snd_pcm_uframes_t dst_offset,
                                       const snd_pcm_channel_area_t *
src_channel,
                                 snd_pcm_uframes_t src_offset,
                                 unsigned int samples,
                                 snd_pcm_format_t format);
extern  int  snd_pcm_area_silence(const  snd_pcm_channel_area_t  *
dst_channel,
                                    snd_pcm_uframes_t dst_offset,
                                    unsigned int samples,
                                    snd_pcm_format_t format);
extern   int   snd_pcm_areas_copy(const   snd_pcm_channel_area_t   *
dst_channels,
                                  snd_pcm_uframes_t dst_offset,
                                       const snd_pcm_channel_area_t *
src_channels,
                                  snd_pcm_uframes_t src_offset,
```

```
                                unsigned int channels,
                                snd_pcm_uframes_t frames,
                                snd_pcm_format_t format);
extern int snd_pcm_areas_silence(const snd_pcm_channel_area_t *
                                   dst_channels,
                                   snd_pcm_uframes_t dst_offset,
                                   unsigned int channels,
                                   snd_pcm_uframes_t frames,
                                   snd_pcm_format_t format);
extern snd_pcm_sframes_t snd_pcm_avail_update(snd_pcm_t * pcm);
extern  snd_pcm_format_t  snd_pcm_build_linear_format(int  width,
int pwidth,
                                                       int unsignd,
                                                           int
big_endian);
extern snd_pcm_sframes_t snd_pcm_bytes_to_frames(snd_pcm_t * pcm,
                                                 ssize_t bytes);
extern long int snd_pcm_bytes_to_samples(snd_pcm_t * pcm, ssize_t
bytes);
extern int snd_pcm_close(snd_pcm_t * pcm);
extern  int  snd_pcm_delay(snd_pcm_t  *  pcm,  snd_pcm_sframes_t  *
delayp);
extern int snd_pcm_drain(snd_pcm_t * pcm);
extern int snd_pcm_drop(snd_pcm_t * pcm);
extern int snd_pcm_dump(snd_pcm_t * pcm, snd_output_t * out);
extern int snd_pcm_format_big_endian(snd_pcm_format_t format);
extern int snd_pcm_format_cpu_endian(snd_pcm_format_t format);
extern  const  char  *snd_pcm_format_description(snd_pcm_format_t
format);
extern int snd_pcm_format_float(snd_pcm_format_t format);
extern int snd_pcm_format_linear(snd_pcm_format_t format);
extern int snd_pcm_format_little_endian(snd_pcm_format_t format);
extern   void   snd_pcm_format_mask_any(snd_pcm_format_mask_t   *
mask);
extern void snd_pcm_format_mask_copy(snd_pcm_format_mask_t * dst,
                                     const snd_pcm_format_mask_t
* src);
extern   void   snd_pcm_format_mask_free(snd_pcm_format_mask_t   *
obj);
extern  int  snd_pcm_format_mask_malloc(snd_pcm_format_mask_t  *
*ptr);
extern   void   snd_pcm_format_mask_none(snd_pcm_format_mask_t   *
mask);
extern void snd_pcm_format_mask_set(snd_pcm_format_mask_t * mask,
                                    snd_pcm_format_t val);
extern size_t snd_pcm_format_mask_sizeof(void);
extern int snd_pcm_format_mask_test(const snd_pcm_format_mask_t *
mask,
                                    snd_pcm_format_t val);
extern const char *snd_pcm_format_name(snd_pcm_format_t format);
extern     int     snd_pcm_format_physical_width(snd_pcm_format_t
format);
extern  int  snd_pcm_format_set_silence(snd_pcm_format_t  format,
void *buf,
                                      unsigned int samples);
extern int snd_pcm_format_signed(snd_pcm_format_t format);
extern ssize_t snd_pcm_format_size(snd_pcm_format_t format,
                                   size_t samples);
extern int snd_pcm_format_unsigned(snd_pcm_format_t format);
extern snd_pcm_format_t snd_pcm_format_value(const char *name);
extern int snd_pcm_format_width(snd_pcm_format_t format);
extern snd_pcm_sframes_t snd_pcm_forward(snd_pcm_t * pcm,
                                         snd_pcm_uframes_t
frames);
extern ssize_t snd_pcm_frames_to_bytes(snd_pcm_t * pcm,
                                       snd_pcm_sframes_t frames);
```

```
extern int snd_pcm_hw_free(snd_pcm_t * pcm);
extern int snd_pcm_hw_params(snd_pcm_t * pcm,
                             snd_pcm_hw_params_t * params);
extern int snd_pcm_hw_params_any(snd_pcm_t * pcm,
                                 snd_pcm_hw_params_t * params);
extern int snd_pcm_hw_params_can_mmap_sample_resolution(const
                                                        snd_pcm_h
w_params_t
                                                                *
params);
extern  int  snd_pcm_hw_params_can_pause(const  snd_pcm_hw_params_t
* params);
extern int snd_pcm_hw_params_can_resume(const snd_pcm_hw_params_t
*
                                        params);
extern         int         snd_pcm_hw_params_can_sync_start(const
snd_pcm_hw_params_t *
                                               params);
extern void snd_pcm_hw_params_copy(snd_pcm_hw_params_t * dst,
                                   const snd_pcm_hw_params_t *
src);
extern int snd_pcm_hw_params_current(snd_pcm_t * pcm,
                                     snd_pcm_hw_params_t *
params);
extern int snd_pcm_hw_params_dump(snd_pcm_hw_params_t * params,
                                  snd_output_t * out);
extern void snd_pcm_hw_params_free(snd_pcm_hw_params_t * obj);
extern int snd_pcm_hw_params_get_access(const snd_pcm_hw_params_t
* params,
                                        snd_pcm_access_t *
_access);
extern  int  snd_pcm_hw_params_get_access_mask(snd_pcm_hw_params_t
* params,
                                               snd_pcm_access_mask_
t * mask);
extern         int         snd_pcm_hw_params_get_buffer_size(const
snd_pcm_hw_params_t *
                                              params,
                                                snd_pcm_uframes_t *
val);
extern       int       snd_pcm_hw_params_get_buffer_size_max(const
snd_pcm_hw_params_t
                                                    * params,
                                                     snd_pcm_uframes_
t * val);
extern       int       snd_pcm_hw_params_get_buffer_size_min(const
snd_pcm_hw_params_t
                                                    * params,
                                                     snd_pcm_uframes_
t * val);
extern         int         snd_pcm_hw_params_get_buffer_time(const
snd_pcm_hw_params_t *
                                              params, unsigned int
*val,
                                              int *dir);
extern       int       snd_pcm_hw_params_get_buffer_time_max(const
snd_pcm_hw_params_t
                                                    * params,
                                                        unsigned int
*val,
                                                    int *dir);
extern       int       snd_pcm_hw_params_get_buffer_time_min(const
snd_pcm_hw_params_t
                                                    * params,
                                                        unsigned int
*val,
```

```
                                                int *dir);
extern          int          snd_pcm_hw_params_get_channels(const
snd_pcm_hw_params_t *
                                               params, unsigned int
*val);
extern        int        snd_pcm_hw_params_get_channels_max(const
snd_pcm_hw_params_t *
                                                params, unsigned
int *val);
extern        int        snd_pcm_hw_params_get_channels_min(const
snd_pcm_hw_params_t *
                                                params, unsigned
int *val);
extern int snd_pcm_hw_params_get_format(const snd_pcm_hw_params_t
* params,
                                        snd_pcm_format_t * val);
extern void snd_pcm_hw_params_get_format_mask(snd_pcm_hw_params_t
* params,
                                              snd_pcm_format_mask
_t *
                                               mask);
extern         int         snd_pcm_hw_params_get_period_size(const
snd_pcm_hw_params_t *
                                              params,
                                                snd_pcm_uframes_t *
frames,
                                              int *dir);
extern       int       snd_pcm_hw_params_get_period_size_max(const
snd_pcm_hw_params_t
                                                * params,
                                                snd_pcm_uframes_
t *
                                                     frames, int
*dir);
extern       int       snd_pcm_hw_params_get_period_size_min(const
snd_pcm_hw_params_t
                                                * params,
                                                snd_pcm_uframes_
t *
                                                     frames, int
*dir);
extern         int         snd_pcm_hw_params_get_period_time(const
snd_pcm_hw_params_t *
                                               params, unsigned int
*val,
                                              int *dir);
extern       int       snd_pcm_hw_params_get_period_time_max(const
snd_pcm_hw_params_t
                                                * params,
                                                    unsigned int
*val,
                                                int *dir);
extern       int       snd_pcm_hw_params_get_period_time_min(const
snd_pcm_hw_params_t
                                                * params,
                                                    unsigned int
*val,
                                                int *dir);
extern           int           snd_pcm_hw_params_get_periods(const
snd_pcm_hw_params_t *
                                               params, unsigned int
*val,
                                            int *dir);
extern         int         snd_pcm_hw_params_get_periods_max(const
snd_pcm_hw_params_t *
                                               params, unsigned int
```

```
*val,
                                                int *dir);
extern         int         snd_pcm_hw_params_get_periods_min(const
snd_pcm_hw_params_t *
                                                params, unsigned int
*val,
                                                int *dir);
extern int snd_pcm_hw_params_get_rate(const snd_pcm_hw_params_t *
params,
                                              unsigned int *val, int
*dir);
extern          int          snd_pcm_hw_params_get_rate_max(const
snd_pcm_hw_params_t *
                                                params, unsigned int
*val,
                                             int *dir);
extern          int          snd_pcm_hw_params_get_rate_min(const
snd_pcm_hw_params_t *
                                                params, unsigned int
*val,
                                             int *dir);
extern         int         snd_pcm_hw_params_get_rate_numden(const
snd_pcm_hw_params_t *
                                                params,
                                                        unsigned int
*rate_num,
                                                        unsigned int
*rate_den);
extern int snd_pcm_hw_params_get_rate_resample(snd_pcm_t * pcm,
                                                  snd_pcm_hw_params_
t *
                                                    params, unsigned
int *val);
extern  int  snd_pcm_hw_params_get_sbits(const  snd_pcm_hw_params_t
* params);
extern  int  snd_pcm_hw_params_is_double(const  snd_pcm_hw_params_t
* params);
extern          int          snd_pcm_hw_params_is_half_duplex(const
snd_pcm_hw_params_t *
                                               params);
extern         int         snd_pcm_hw_params_is_joint_duplex(const
snd_pcm_hw_params_t *
                                                params);
extern int snd_pcm_hw_params_malloc(snd_pcm_hw_params_t * *ptr);
extern int snd_pcm_hw_params_set_access(snd_pcm_t * pcm,
                                               snd_pcm_hw_params_t *
params,
                                                    snd_pcm_access_t
_access);
extern int snd_pcm_hw_params_set_access_mask(snd_pcm_t * pcm,
                                                  snd_pcm_hw_params_t
* params,
                                                 snd_pcm_access_mask_
t * mask);
extern int snd_pcm_hw_params_set_buffer_size(snd_pcm_t * pcm,
                                                  snd_pcm_hw_params_t
* params,
                                                    snd_pcm_uframes_t
val);
extern   int   snd_pcm_hw_params_set_buffer_size_near(snd_pcm_t   *
pcm,
                                                     snd_pcm_hw_para
ms_t *
                                                    params,
                                                     snd_pcm_uframes
_t * val);
```

```
extern int snd_pcm_hw_params_set_buffer_time(snd_pcm_t * pcm,
                                             snd_pcm_hw_params_t
* params,
                                               unsigned int val,
int dir);
extern  int  snd_pcm_hw_params_set_buffer_time_near(snd_pcm_t  *
pcm,
                                                snd_pcm_hw_para
ms_t *
                                               params,
                                                   unsigned int
*val,
                                               int *dir);
extern int snd_pcm_hw_params_set_channels(snd_pcm_t * pcm,
                                          snd_pcm_hw_params_t *
params,
                                        unsigned int val);
extern int snd_pcm_hw_params_set_channels_near(snd_pcm_t * pcm,
                                            snd_pcm_hw_params_
t *
                                              params, unsigned
int *val);
extern int snd_pcm_hw_params_set_format(snd_pcm_t * pcm,
                                        snd_pcm_hw_params_t *
params,
                                      snd_pcm_format_t val);
extern int snd_pcm_hw_params_set_format_mask(snd_pcm_t * pcm,
                                             snd_pcm_hw_params_t
* params,
                                            snd_pcm_format_mask_
t * mask);
extern int snd_pcm_hw_params_set_period_size(snd_pcm_t * pcm,
                                             snd_pcm_hw_params_t
* params,
                                               snd_pcm_uframes_t
val,
                                            int dir);
extern  int  snd_pcm_hw_params_set_period_size_near(snd_pcm_t  *
pcm,
                                                snd_pcm_hw_para
ms_t *
                                               params,
                                                snd_pcm_uframes
_t * val,
                                               int *dir);
extern int snd_pcm_hw_params_set_period_time(snd_pcm_t * pcm,
                                             snd_pcm_hw_params_t
* params,
                                               unsigned int val,
int dir);
extern  int  snd_pcm_hw_params_set_period_time_near(snd_pcm_t  *
pcm,
                                                snd_pcm_hw_para
ms_t *
                                               params,
                                                   unsigned int
*val,
                                               int *dir);
extern int snd_pcm_hw_params_set_periods(snd_pcm_t * pcm,
                                         snd_pcm_hw_params_t *
params,
                                         unsigned int val, int
dir);
extern int snd_pcm_hw_params_set_periods_integer(snd_pcm_t * pcm,
                                                 snd_pcm_hw_param
s_t *
```

```
                                                    params);
extern int snd_pcm_hw_params_set_periods_near(snd_pcm_t * pcm,
                                              snd_pcm_hw_params_t
* params,
                                               unsigned int *val,
int *dir);
extern int snd_pcm_hw_params_set_rate(snd_pcm_t * pcm,
                                      snd_pcm_hw_params_t *
params,
                                  unsigned int val, int dir);
extern int snd_pcm_hw_params_set_rate_near(snd_pcm_t * pcm,
                                      snd_pcm_hw_params_t *
params,
                                     unsigned int *val, int
*dir);
extern int snd_pcm_hw_params_set_rate_resample(snd_pcm_t * pcm,
                                               snd_pcm_hw_params_
t *
                                                params, unsigned
int val);
extern size_t snd_pcm_hw_params_sizeof(void);
extern int snd_pcm_hw_params_test_access(snd_pcm_t * pcm,
                                      snd_pcm_hw_params_t *
params,
                                                 snd_pcm_access_t
_access);
extern int snd_pcm_hw_params_test_buffer_size(snd_pcm_t * pcm,
                                              snd_pcm_hw_params_t
* params,
                                                snd_pcm_uframes_t
val);
extern int snd_pcm_hw_params_test_buffer_time(snd_pcm_t * pcm,
                                              snd_pcm_hw_params_t
* params,
                                                unsigned int val,
int dir);
extern int snd_pcm_hw_params_test_channels(snd_pcm_t * pcm,
                                      snd_pcm_hw_params_t *
params,
                                    unsigned int val);
extern int snd_pcm_hw_params_test_format(snd_pcm_t * pcm,
                                      snd_pcm_hw_params_t *
params,
                                   snd_pcm_format_t val);
extern int snd_pcm_hw_params_test_period_size(snd_pcm_t * pcm,
                                              snd_pcm_hw_params_t
* params,
                                                snd_pcm_uframes_t
val,
                                               int dir);
extern int snd_pcm_hw_params_test_period_time(snd_pcm_t * pcm,
                                              snd_pcm_hw_params_t
* params,
                                                unsigned int val,
int dir);
extern int snd_pcm_hw_params_test_periods(snd_pcm_t * pcm,
                                      snd_pcm_hw_params_t *
params,
                                      unsigned int val, int
dir);
extern int snd_pcm_hw_params_test_rate(snd_pcm_t * pcm,
                                      snd_pcm_hw_params_t *
params,
                                     unsigned int val, int
dir);
extern int snd_pcm_hwsync(snd_pcm_t * pcm);
```

```
extern int snd_pcm_info(snd_pcm_t * pcm, snd_pcm_info_t * info);
extern void snd_pcm_info_copy(snd_pcm_info_t * dst,
                              const snd_pcm_info_t * src);
extern void snd_pcm_info_free(snd_pcm_info_t * obj);
extern int snd_pcm_info_get_card(const snd_pcm_info_t * obj);
extern        snd_pcm_class_t        snd_pcm_info_get_class(const
snd_pcm_info_t * obj);
extern unsigned int snd_pcm_info_get_device(const snd_pcm_info_t
* obj);
extern  const  char  *snd_pcm_info_get_id(const  snd_pcm_info_t  *
obj);
extern const char *snd_pcm_info_get_name(const snd_pcm_info_t *
obj);
extern       snd_pcm_stream_t       snd_pcm_info_get_stream(const
snd_pcm_info_t *
                                                    obj);
extern     unsigned     int     snd_pcm_info_get_subdevice(const
snd_pcm_info_t * obj);
extern    const    char    *snd_pcm_info_get_subdevice_name(const
snd_pcm_info_t *
                                                      obj);
extern   unsigned   int   snd_pcm_info_get_subdevices_avail(const
snd_pcm_info_t
                                                       * obj);
extern   unsigned   int   snd_pcm_info_get_subdevices_count(const
snd_pcm_info_t
                                                       * obj);
extern int snd_pcm_info_malloc(snd_pcm_info_t * *ptr);
extern void snd_pcm_info_set_device(snd_pcm_info_t * obj,
                                    unsigned int val);
extern void snd_pcm_info_set_stream(snd_pcm_info_t * obj,
                                    snd_pcm_stream_t val);
extern void snd_pcm_info_set_subdevice(snd_pcm_info_t * obj,
                                       unsigned int val);
extern size_t snd_pcm_info_sizeof(void);
extern int snd_pcm_link(snd_pcm_t * pcm1, snd_pcm_t * pcm2);
extern int snd_pcm_mmap_begin(snd_pcm_t * pcm,
                                   const snd_pcm_channel_area_t *
*areas,
                              snd_pcm_uframes_t * offset,
                              snd_pcm_uframes_t * frames);
extern snd_pcm_sframes_t snd_pcm_mmap_commit(snd_pcm_t * pcm,
                                                    snd_pcm_uframes_t
offset,
                                                    snd_pcm_uframes_t
frames);
extern snd_pcm_sframes_t snd_pcm_mmap_readi(snd_pcm_t * pcm, void
*buffer,
                                                    snd_pcm_uframes_t
size);
extern snd_pcm_sframes_t snd_pcm_mmap_readn(snd_pcm_t * pcm, void
**bufs,
                                                    snd_pcm_uframes_t
size);
extern snd_pcm_sframes_t snd_pcm_mmap_writei(snd_pcm_t * pcm,
                                                const void *buffer,
                                                    snd_pcm_uframes_t
size);
extern  snd_pcm_sframes_t  snd_pcm_mmap_writen(snd_pcm_t  *  pcm,
void **bufs,
                                                    snd_pcm_uframes_t
size);
extern const char *snd_pcm_name(snd_pcm_t * pcm);
extern int snd_pcm_nonblock(snd_pcm_t * pcm, int nonblock);
extern int snd_pcm_open(snd_pcm_t * *pcm, const char *name,
                        snd_pcm_stream_t stream, int mode);
```

```
extern int snd_pcm_open_lconf(snd_pcm_t * *pcm, const char *name,
                              snd_pcm_stream_t stream, int mode,
                              snd_config_t * lconf);
extern int snd_pcm_pause(snd_pcm_t * pcm, int enable);
extern  int  snd_pcm_poll_descriptors(snd_pcm_t  *  pcm,  struct
pollfd *pfds,
                                      unsigned int space);
extern int snd_pcm_poll_descriptors_count(snd_pcm_t * pcm);
extern int snd_pcm_poll_descriptors_revents(snd_pcm_t * pcm,
                                            struct pollfd *pfds,
                                            unsigned int nfds,
                                               short unsigned int
*revents);
extern int snd_pcm_prepare(snd_pcm_t * pcm);
extern  snd_pcm_sframes_t  snd_pcm_readi(snd_pcm_t  *  pcm,  void
*buffer,
                                         snd_pcm_uframes_t size);
extern  snd_pcm_sframes_t  snd_pcm_readn(snd_pcm_t  *  pcm,  void
**bufs,
                                         snd_pcm_uframes_t size);
extern int snd_pcm_recover(snd_pcm_t * pcm, int err, int silent);
extern int snd_pcm_reset(snd_pcm_t * pcm);
extern int snd_pcm_resume(snd_pcm_t * pcm);
extern snd_pcm_sframes_t snd_pcm_rewind(snd_pcm_t * pcm,
                                                 snd_pcm_uframes_t
frames);
extern ssize_t snd_pcm_samples_to_bytes(snd_pcm_t * pcm, long int
samples);
extern int snd_pcm_start(snd_pcm_t * pcm);
extern snd_pcm_state_t snd_pcm_state(snd_pcm_t * pcm);
extern const char *snd_pcm_state_name(snd_pcm_state_t state);
extern  int  snd_pcm_status(snd_pcm_t  *  pcm,  snd_pcm_status_t  *
status);
extern void snd_pcm_status_copy(snd_pcm_status_t * dst,
                                const snd_pcm_status_t * src);
extern int snd_pcm_status_dump(snd_pcm_status_t * status,
                               snd_output_t * out);
extern void snd_pcm_status_free(snd_pcm_status_t * obj);
extern      snd_pcm_uframes_t      snd_pcm_status_get_avail(const
snd_pcm_status_t *
                                                    obj);
extern snd_pcm_uframes_t snd_pcm_status_get_avail_max(const
                                                         snd_pcm_sta
tus_t *
                                                        obj);
extern      snd_pcm_sframes_t      snd_pcm_status_get_delay(const
snd_pcm_status_t *
                                                    obj);
extern       snd_pcm_state_t       snd_pcm_status_get_state(const
snd_pcm_status_t *
                                                   obj);
extern        void        snd_pcm_status_get_trigger_tstamp(const
snd_pcm_status_t * obj,
                                                    snd_timestamp_t *
ptr);
extern  void  snd_pcm_status_get_tstamp(const  snd_pcm_status_t  *
obj,
                                        snd_timestamp_t * ptr);
extern int snd_pcm_status_malloc(snd_pcm_status_t * *ptr);
extern size_t snd_pcm_status_sizeof(void);
extern snd_pcm_stream_t snd_pcm_stream(snd_pcm_t * pcm);
extern const char *snd_pcm_stream_name(snd_pcm_stream_t stream);
extern int snd_pcm_sw_params(snd_pcm_t * pcm,
                             snd_pcm_sw_params_t * params);
extern void snd_pcm_sw_params_copy(snd_pcm_sw_params_t * dst,
                                   const snd_pcm_sw_params_t *
```

```
src);
extern int snd_pcm_sw_params_current(snd_pcm_t * pcm,
                                                snd_pcm_sw_params_t *
params);
extern int snd_pcm_sw_params_dump(snd_pcm_sw_params_t * params,
                                  snd_output_t * out);
extern void snd_pcm_sw_params_free(snd_pcm_sw_params_t * obj);
extern          int          snd_pcm_sw_params_get_avail_min(const
snd_pcm_sw_params_t *
                                               params,
                                                   snd_pcm_uframes_t *
val);
extern          int          snd_pcm_sw_params_get_boundary(const
snd_pcm_sw_params_t *
                                                                  params,
snd_pcm_uframes_t * val);
extern        int        snd_pcm_sw_params_get_silence_size(const
snd_pcm_sw_params_t *
                                              params,
                                               snd_pcm_uframes_t *
val);
extern int snd_pcm_sw_params_get_silence_threshold(const
                                                    snd_pcm_sw_par
ams_t *
                                                   params,
                                                    snd_pcm_uframe
s_t *
                                                   val);
extern       int       snd_pcm_sw_params_get_start_threshold(const
snd_pcm_sw_params_t
                                                  * paramsm,
                                                   snd_pcm_uframes_
t * val);
extern       int       snd_pcm_sw_params_get_stop_threshold(const
snd_pcm_sw_params_t *
                                                 params,
                                                  snd_pcm_uframes_t
* val);
extern         int         snd_pcm_sw_params_get_tstamp_mode(const
snd_pcm_sw_params_t *
                                               params,
                                                  snd_pcm_tstamp_t *
val);
extern int snd_pcm_sw_params_malloc(snd_pcm_sw_params_t * *ptr);
extern int snd_pcm_sw_params_set_avail_min(snd_pcm_t * pcm,
                                               snd_pcm_sw_params_t *
params,
                                                   snd_pcm_uframes_t
val);
extern int snd_pcm_sw_params_set_silence_size(snd_pcm_t * pcm,
                                                 snd_pcm_sw_params_t
* params,
                                                   snd_pcm_uframes_t
val);
extern  int  snd_pcm_sw_params_set_silence_threshold(snd_pcm_t  *
pcm,
                                                    snd_pcm_sw_par
ams_t *
                                                   params,
                                                    snd_pcm_uframe
s_t val);
extern int snd_pcm_sw_params_set_start_threshold(snd_pcm_t * pcm,
                                                   snd_pcm_sw_param
s_t *
                                                  params,
                                                   snd_pcm_uframes_
```

```
t val);
extern int snd_pcm_sw_params_set_stop_threshold(snd_pcm_t * pcm,
                                                snd_pcm_sw_params
_t *
                                                params,
                                                snd_pcm_uframes_t
val);
extern int snd_pcm_sw_params_set_tstamp_mode(snd_pcm_t * pcm,
                                             snd_pcm_sw_params_t
* params,
                                                snd_pcm_tstamp_t
val);
extern int snd_pcm_sw_params_set_xfer_align(snd_pcm_t * pcm,
                                            snd_pcm_sw_params_t *
params,
                                                snd_pcm_uframes_t
val);
extern size_t snd_pcm_sw_params_sizeof(void);
extern snd_pcm_type_t snd_pcm_type(snd_pcm_t * pcm);
extern const char *snd_pcm_type_name(snd_pcm_type_t type);
extern int snd_pcm_unlink(snd_pcm_t * pcm);
extern int snd_pcm_wait(snd_pcm_t * pcm, int timeout);
extern snd_pcm_sframes_t snd_pcm_writei(snd_pcm_t * pcm,
                                        const void *buffer,
                                        snd_pcm_uframes_t size);
extern  snd_pcm_sframes_t  snd_pcm_writen(snd_pcm_t  *  pcm,  void
**bufs,
                                        snd_pcm_uframes_t size);
```

6.2.13 alsa/pcm_extplug.h

```
#define                                    SND_PCM_EXTPLUG_VERSION
((SND_PCM_EXTPLUG_VERSION_MAJOR<<16)                              |
(SND_PCM_EXTPLUG_VERSION_MINOR<<8)                                |
(SND_PCM_EXTPLUG_VERSION_TINY))
#define SND_PCM_EXTPLUG_VERSION_MINOR   0
#define SND_PCM_EXTPLUG_VERSION_MAJOR   1
#define SND_PCM_EXTPLUG_VERSION_TINY    1

typedef struct snd_pcm_extplug_callback {
    snd_pcm_sframes_t(*transfer) (void);
    int (*close) (void);
    int (*hw_params) (void);
    int (*hw_free) (void);
    void (*dump) (void);
    int (*init) (void);
} snd_pcm_extplug_callback_t;
typedef struct snd_pcm_extplug {
    unsigned int version;
    const char *name;
    const snd_pcm_extplug_callback_t *callback;
    void *private_data;
    snd_pcm_t *pcm;
    snd_pcm_stream_t stream;
    snd_pcm_format_t format;
    snd_pcm_subformat_t subformat;
    unsigned int channels;
    unsigned int rate;
    snd_pcm_format_t slave_format;
    snd_pcm_subformat_t slave_subformat;
    unsigned int slave_channels;
} snd_pcm_extplug_t;
```

6.2.14 alsa/pcm_plugin.h

```
#define SND_PCM_PLUGIN_ROUTE_HALF       0.5
#define SND_PCM_PLUGIN_ROUTE_FLOAT      1
#define SND_PCM_PLUGIN_ROUTE_FULL       1.0
#define SND_PCM_PLUGIN_ROUTE_RESOLUTION 16
#define SND_PCM_PLUGIN_RATE_MAX 192000
#define SND_PCM_PLUGIN_RATE_MIN 4000

typedef float snd_pcm_route_ttable_entry_t;
```

6.2.15 alsa/rawmidi.h

```
#define SND_RAWMIDI_APPEND      0x0001
#define SND_RAWMIDI_NONBLOCK    0x0002
#define SND_RAWMIDI_SYNC        0x0004

typedef struct sndrv_rawmidi_info snd_rawmidi_info_t;
typedef struct sndrv_rawmidi_params snd_rawmidi_params_t;
typedef struct sndrv_rawmidi_status snd_rawmidi_status_t;
typedef enum _snd_rawmidi_stream {
    SND_RAWMIDI_STREAM_OUTPUT,
    SND_RAWMIDI_STREAM_INPUT = 1,
    SND_RAWMIDI_STREAM_LAST = 1
} snd_rawmidi_stream_t;
typedef struct _snd_rawmidi snd_rawmidi_t;
typedef enum _snd_rawmidi_type {
    SND_RAWMIDI_TYPE_HW,
    SND_RAWMIDI_TYPE_SHM = 1,
    SND_RAWMIDI_TYPE_INET = 2,
    SND_RAWMIDI_TYPE_VIRTUAL = 3
} snd_rawmidi_type_t;
extern int snd_rawmidi_close(snd_rawmidi_t * rmidi);
extern int snd_rawmidi_drain(snd_rawmidi_t * rmidi);
extern int snd_rawmidi_drop(snd_rawmidi_t * rmidi);
extern void snd_rawmidi_info_free(snd_rawmidi_info_t * obj);
extern       const       char       *snd_rawmidi_info_get_id(const
snd_rawmidi_info_t * obj);
extern      const      char      *snd_rawmidi_info_get_name(const
snd_rawmidi_info_t *
                                             obj);
extern const char *snd_rawmidi_info_get_subdevice_name(const
                                                        snd_rawmid
i_info_t *
                                                       obj);
extern unsigned int snd_rawmidi_info_get_subdevices_count(const
                                                          snd_raw
midi_info_t
                                                         * obj);
extern int snd_rawmidi_info_malloc(snd_rawmidi_info_t * *ptr);
extern void snd_rawmidi_info_set_device(snd_rawmidi_info_t * obj,
                                        unsigned int val);
extern void snd_rawmidi_info_set_stream(snd_rawmidi_info_t * obj,
                                          snd_rawmidi_stream_t
val);
extern  void  snd_rawmidi_info_set_subdevice(snd_rawmidi_info_t  *
obj,
                                             unsigned int val);
extern size_t snd_rawmidi_info_sizeof(void);
extern   int   snd_rawmidi_nonblock(snd_rawmidi_t   *   rmidi,   int
nonblock);
extern int snd_rawmidi_open(snd_rawmidi_t * *in_rmidi,
                            snd_rawmidi_t * *out_rmidi, const
```

```
char *name,
                              int mode);
extern int snd_rawmidi_poll_descriptors(snd_rawmidi_t * rmidi,
                                        struct pollfd *pfds,
                                        unsigned int space);
extern   int   snd_rawmidi_poll_descriptors_count(snd_rawmidi_t   *
rmidi);
extern  int  snd_rawmidi_poll_descriptors_revents(snd_rawmidi_t  *
rawmidi,
                                                   struct pollfd
*pfds,
                                                    unsigned int
nfds,
                                                  short unsigned
int
                                                 *revent);
extern  ssize_t  snd_rawmidi_read(snd_rawmidi_t  *  rmidi,  void
*buffer,
                                  size_t size);
extern  ssize_t  snd_rawmidi_write(snd_rawmidi_t  *  rmidi,  const
void *buffer,
                                   size_t size);
```

6.2.16 alsa/seq.h

```
#define snd_seq_ev_is_prior(ev)  \
                    (((ev)->flags  &  SND_SEQ_PRIORITY_MASK)  ==
SND_SEQ_PRIORITY_HIGH)
#define snd_seq_ev_length_type(ev)       \
        ((ev)->flags & SND_SEQ_EVENT_LENGTH_MASK)
#define snd_seq_ev_timemode_type(ev)     \
        ((ev)->flags & SND_SEQ_TIME_MODE_MASK)
#define snd_seq_ev_timestamp_type(ev)    \
        ((ev)->flags & SND_SEQ_TIME_STAMP_MASK)
#define snd_seq_ev_is_channel_type(ev)   \
                              (snd_seq_event_types[(ev)->type]    &
(_SND_SEQ_TYPE(SND_SEQ_EVFLG_NOTE) \
        | _SND_SEQ_TYPE(SND_SEQ_EVFLG_CONTROL)))
#define snd_seq_type_check(ev,x)         \
        (snd_seq_event_types[(ev)->type] & _SND_SEQ_TYPE(x))
#define snd_seq_ev_is_fixed(ev)  \
                                  (snd_seq_ev_length_type(ev)    ==
SND_SEQ_EVENT_LENGTH_FIXED)
#define snd_seq_ev_is_variable(ev)       \
                                  (snd_seq_ev_length_type(ev)    ==
SND_SEQ_EVENT_LENGTH_VARIABLE)
#define snd_seq_ev_is_varusr(ev)         \
                                  (snd_seq_ev_length_type(ev)    ==
SND_SEQ_EVENT_LENGTH_VARUSR)
#define snd_seq_ev_is_abstime(ev)        \
        (snd_seq_ev_timemode_type(ev) == SND_SEQ_TIME_MODE_ABS)
#define snd_seq_ev_is_reltime(ev)        \
        (snd_seq_ev_timemode_type(ev) == SND_SEQ_TIME_MODE_REL)
#define snd_seq_ev_is_real(ev)   \
                                 (snd_seq_ev_timestamp_type(ev)    ==
SND_SEQ_TIME_STAMP_REAL)
#define snd_seq_ev_is_tick(ev)   \
                                 (snd_seq_ev_timestamp_type(ev)    ==
SND_SEQ_TIME_STAMP_TICK)
#define snd_seq_ev_is_subscribe_type(ev)         \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_CONNECTION)
#define snd_seq_ev_is_control_type(ev)   \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_CONTROL)
#define snd_seq_ev_is_fixed_type(ev)     \
```

```
        snd_seq_type_check(ev, SND_SEQ_EVFLG_FIXED)
#define snd_seq_ev_is_instr_type(ev)     \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_INSTR)
#define snd_seq_ev_is_message_type(ev)   \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_MESSAGE)
#define snd_seq_ev_is_note_type(ev)      \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_NOTE)
#define snd_seq_ev_is_queue_type(ev)     \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_QUEUE)
#define snd_seq_ev_is_result_type(ev)    \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_RESULT)
#define snd_seq_ev_is_sample_type(ev)    \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_SAMPLE)
#define snd_seq_ev_is_user_type(ev)      \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_USERS)
#define snd_seq_ev_is_variable_type(ev)  \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_VARIABLE)
#define snd_seq_ev_is_varusr_type(ev)    \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_VARUSR)
#define    snd_seq_ev_is_reserved(ev)                          (!
snd_seq_event_types[(ev)->type])
#define  snd_seq_ev_is_direct(ev)               ((ev)->queue  ==
SND_SEQ_QUEUE_DIRECT)
#define _SND_SEQ_TYPE_OPT(x)    ((x)<<24)
#define _SND_SEQ_TYPE(x)        (1<<(x))
#define SND_SEQ_PORT_CAP_READ   (1<<0)
#define SND_SEQ_PORT_TYPE_SPECIFIC      (1<<0)
#define SND_SEQ_REMOVE_INPUT    (1<<0)
#define SND_SEQ_PORT_CAP_WRITE  (1<<1)
#define SND_SEQ_PORT_TYPE_MIDI_GENERIC  (1<<1)
#define SND_SEQ_REMOVE_OUTPUT   (1<<1)
#define SND_SEQ_PORT_TYPE_SYNTH (1<<10)
#define SND_SEQ_PORT_TYPE_DIRECT_SAMPLE (1<<11)
#define SND_SEQ_PORT_TYPE_SAMPLE        (1<<12)
#define SND_SEQ_PORT_TYPE_HARDWARE      (1<<16)
#define SND_SEQ_PORT_TYPE_SOFTWARE      (1<<17)
#define SND_SEQ_PORT_TYPE_SYNTHESIZER   (1<<18)
#define SND_SEQ_PORT_TYPE_PORT  (1<<19)
#define SND_SEQ_PORT_CAP_SYNC_READ      (1<<2)
#define SND_SEQ_PORT_TYPE_MIDI_GM       (1<<2)
#define SND_SEQ_REMOVE_DEST     (1<<2)
#define SND_SEQ_PORT_TYPE_APPLICATION   (1<<20)
#define SND_SEQ_PORT_CAP_SYNC_WRITE     (1<<3)
#define SND_SEQ_PORT_TYPE_MIDI_GS       (1<<3)
#define SND_SEQ_REMOVE_DEST_CHANNEL     (1<<3)
#define SND_SEQ_PORT_CAP_DUPLEX (1<<4)
#define SND_SEQ_PORT_TYPE_MIDI_XG       (1<<4)
#define SND_SEQ_REMOVE_TIME_BEFORE      (1<<4)
#define SND_SEQ_PORT_CAP_SUBS_READ      (1<<5)
#define SND_SEQ_PORT_TYPE_MIDI_MT32     (1<<5)
#define SND_SEQ_REMOVE_TIME_AFTER       (1<<5)
#define SND_SEQ_PORT_CAP_SUBS_WRITE     (1<<6)
#define SND_SEQ_PORT_TYPE_MIDI_GM2      (1<<6)
#define SND_SEQ_REMOVE_TIME_TICK        (1<<6)
#define SND_SEQ_PORT_CAP_NO_EXPORT      (1<<7)
#define SND_SEQ_REMOVE_EVENT_TYPE       (1<<7)
#define SND_SEQ_REMOVE_IGNORE_OFF       (1<<8)
#define SND_SEQ_REMOVE_TAG_MATCH        (1<<9)
#define   SND_SEQ_OPEN_DUPLEX               (SND_SEQ_OPEN_OUTPUT|
SND_SEQ_OPEN_INPUT)
#define SND_SEQ_CLIENT_SYSTEM   0
#define SND_SEQ_PORT_SYSTEM_TIMER       0
#define SND_SEQ_NONBLOCK        0x0001
#define SND_SEQ_OPEN_OUTPUT     1
#define SND_SEQ_PORT_SYSTEM_ANNOUNCE    1
#define SND_SEQ_OPEN_INPUT      2
```

```
#define SND_SEQ_ADDRESS_UNKNOWN 253
#define SND_SEQ_QUEUE_DIRECT    253
#define SND_SEQ_ADDRESS_SUBSCRIBERS     254
#define SND_SEQ_ADDRESS_BROADCAST       255

typedef struct sndrv_seq_client_info snd_seq_client_info_t;
typedef struct sndrv_seq_client_pool snd_seq_client_pool_t;
typedef enum snd_seq_client_type {
    SND_SEQ_USER_CLIENT = 1,
    SND_SEQ_KERNEL_CLIENT = 2
} snd_seq_client_type_t;
typedef struct sndrv_seq_port_info snd_seq_port_info_t;
typedef struct sndrv_seq_port_subscribe snd_seq_port_subscribe_t;
typedef enum {
    SND_SEQ_QUERY_SUBS_READ,
    SND_SEQ_QUERY_SUBS_WRITE = 1
} snd_seq_query_subs_type_t;
typedef struct sndrv_seq_query_subs snd_seq_query_subscribe_t;
typedef struct sndrv_seq_queue_info snd_seq_queue_info_t;
typedef struct sndrv_seq_queue_status snd_seq_queue_status_t;
typedef struct sndrv_seq_queue_tempo snd_seq_queue_tempo_t;
typedef struct sndrv_seq_queue_timer snd_seq_queue_timer_t;
typedef enum {
    SND_SEQ_TIMER_ALSA,
    SND_SEQ_TIMER_MIDI_CLOCK = 1,
    SND_SEQ_TIMER_MIDI_TICK = 2
} snd_seq_queue_timer_type_t;
typedef struct sndrv_seq_remove_events snd_seq_remove_events_t;
typedef struct sndrv_seq_system_info snd_seq_system_info_t;
typedef struct _snd_seq snd_seq_t;
typedef enum _snd_seq_type {
    SND_SEQ_TYPE_HW,
    SND_SEQ_TYPE_SHM = 1,
    SND_SEQ_TYPE_INET = 2
} snd_seq_type_t;
extern int snd_seq_alloc_named_queue(snd_seq_t * seq, const char
*name);
extern int snd_seq_alloc_queue(snd_seq_t * handle);
extern int snd_seq_client_id(snd_seq_t * handle);
extern void snd_seq_client_info_copy(snd_seq_client_info_t * dst,
                                     const snd_seq_client_info_t
* src);
extern   void   snd_seq_client_info_free(snd_seq_client_info_t   *
ptr);
extern          int          snd_seq_client_info_get_client(const
snd_seq_client_info_t *
                                       info);
extern                          const                          char
*snd_seq_client_info_get_name(snd_seq_client_info_t *
                                                  info);
extern         int         snd_seq_client_info_get_num_ports(const
snd_seq_client_info_t *
                                                 info);
extern snd_seq_client_type_t snd_seq_client_info_get_type(const
                                                                snd_seq
_client_info_t
                                                                     *
info);
extern   int   snd_seq_client_info_malloc(snd_seq_client_info_t   *
*ptr);
extern  void  snd_seq_client_info_set_client(snd_seq_client_info_t
* info,
                                             int client);
extern  void  snd_seq_client_info_set_name(snd_seq_client_info_t  *
info,
                                           const char *name);
```

```
extern size_t snd_seq_client_info_sizeof(void);
extern int snd_seq_close(snd_seq_t * handle);
extern int snd_seq_create_port(snd_seq_t * handle,
                               snd_seq_port_info_t * info);
extern int snd_seq_delete_port(snd_seq_t * handle, int port);
extern int snd_seq_drain_output(snd_seq_t * handle);
extern int snd_seq_drop_output(snd_seq_t * handle);
extern int snd_seq_drop_output_buffer(snd_seq_t * handle);
extern     int     snd_seq_event_input(snd_seq_t     *     handle,
snd_seq_event_t * *ev);
extern int snd_seq_event_input_pending(snd_seq_t * seq,
                                       int fetch_sequencer);
extern ssize_t snd_seq_event_length(snd_seq_event_t * ev);
extern     int     snd_seq_event_output(snd_seq_t     *     handle,
snd_seq_event_t * ev);
extern int snd_seq_event_output_direct(snd_seq_t * handle,
                                       snd_seq_event_t * ev);
extern const unsigned int snd_seq_event_types[];
extern int snd_seq_free_event(snd_seq_event_t * ev);
extern int snd_seq_free_queue(snd_seq_t * handle, int q);
extern  int  snd_seq_get_any_client_info(snd_seq_t  *  handle,  int
client,
                                         snd_seq_client_info_t *
info);
extern  int  snd_seq_get_any_port_info(snd_seq_t  *  handle,  int
client,
                                                    int  port,
snd_seq_port_info_t * info);
extern int snd_seq_get_client_info(snd_seq_t * handle,
                                   snd_seq_client_info_t * info);
extern size_t snd_seq_get_input_buffer_size(snd_seq_t * handle);
extern size_t snd_seq_get_output_buffer_size(snd_seq_t * handle);
extern int snd_seq_get_port_info(snd_seq_t * handle, int port,
                                 snd_seq_port_info_t * info);
extern int snd_seq_get_port_subscription(snd_seq_t * handle,
                                         snd_seq_port_subscribe_t
* sub);
extern int snd_seq_get_queue_status(snd_seq_t * handle, int q,
                                    snd_seq_queue_status_t *
status);
extern int snd_seq_get_queue_tempo(snd_seq_t * handle, int q,
                                     snd_seq_queue_tempo_t *
tempo);
extern int snd_seq_nonblock(snd_seq_t * handle, int nonblock);
extern  int  snd_seq_open(snd_seq_t  *  *handle,  const  char  *name,
int streams,
                          int mode);
extern int snd_seq_poll_descriptors(snd_seq_t * handle,
                                    struct pollfd *pfds,
                                     unsigned int space, short int
events);
extern int snd_seq_poll_descriptors_count(snd_seq_t * handle,
                                          short int events);
extern int snd_seq_poll_descriptors_revents(snd_seq_t * seq,
                                            struct pollfd *pfds,
                                            unsigned int nfds,
                                               short unsigned int
*revents);
extern void snd_seq_port_info_copy(snd_seq_port_info_t * dst,
                                   const snd_seq_port_info_t *
src);
extern void snd_seq_port_info_free(snd_seq_port_info_t * ptr);
extern const snd_seq_addr_t *snd_seq_port_info_get_addr(const
                                                         snd_seq_p
ort_info_t
                                                        * info);
```

```
extern unsigned int snd_seq_port_info_get_capability(const
                                                     snd_seq_port
_info_t *
                                                    info);
extern int snd_seq_port_info_get_client(const snd_seq_port_info_t
* info);
extern      const      char      *snd_seq_port_info_get_name(const
snd_seq_port_info_t *
                                                 info);
extern int snd_seq_port_info_get_port(const snd_seq_port_info_t *
info);
extern     unsigned     int     snd_seq_port_info_get_type(const
snd_seq_port_info_t *
                                                  info);
extern int snd_seq_port_info_malloc(snd_seq_port_info_t * *ptr);
extern  void  snd_seq_port_info_set_capability(snd_seq_port_info_t
* info,
                                                    unsigned int
capability);
extern  void  snd_seq_port_info_set_client(snd_seq_port_info_t  *
info,
                                        int client);
extern                                                    void
snd_seq_port_info_set_midi_channels(snd_seq_port_info_t * info,
                                                   int channels);
extern   void   snd_seq_port_info_set_name(snd_seq_port_info_t   *
info,
                                       const char *name);
extern   void   snd_seq_port_info_set_port(snd_seq_port_info_t   *
info,
                                       int port);
extern                                                    void
snd_seq_port_info_set_port_specified(snd_seq_port_info_t *
                                                    info, int val);
extern                                                    void
snd_seq_port_info_set_timestamp_queue(snd_seq_port_info_t *
                                                      info, int
queue);
extern                                                    void
snd_seq_port_info_set_timestamp_real(snd_seq_port_info_t *
                                                     info, int
realtime);
extern                                                    void
snd_seq_port_info_set_timestamping(snd_seq_port_info_t * info,
                                                   int enable);
extern   void   snd_seq_port_info_set_type(snd_seq_port_info_t   *
info,
                                       unsigned int type);
extern size_t snd_seq_port_info_sizeof(void);
extern  void  snd_seq_port_subscribe_copy(snd_seq_port_subscribe_t
* dst,
                                                          const
snd_seq_port_subscribe_t *
                                        src);
extern  void  snd_seq_port_subscribe_free(snd_seq_port_subscribe_t
* ptr);
extern                     const                    snd_seq_addr_t
*snd_seq_port_subscribe_get_dest(const
                                                           snd_
seq_port_subscribe_t
                                                             *
info);
extern int snd_seq_port_subscribe_get_exclusive(const
                                                    snd_seq_port_subs
cribe_t *
                                                   info);
```

```
extern         int         snd_seq_port_subscribe_get_queue(const
snd_seq_port_subscribe_t
                                                * info);
extern                     const                    snd_seq_addr_t
*snd_seq_port_subscribe_get_sender(const
                                                                    sn
d_seq_port_subscribe_t
                                                                     *
info);
extern int snd_seq_port_subscribe_get_time_real(const
                                                      snd_seq_port_subs
cribe_t *
                                                     info);
extern int snd_seq_port_subscribe_get_time_update(const
                                                       snd_seq_port_su
bscribe_t
                                                      * info);
extern  int  snd_seq_port_subscribe_malloc(snd_seq_port_subscribe_t
* *ptr);
extern                                                              void
snd_seq_port_subscribe_set_dest(snd_seq_port_subscribe_t *
                                                 info,
                                                   const snd_seq_addr_t
* addr);
extern                                                              void
snd_seq_port_subscribe_set_exclusive(snd_seq_port_subscribe_t *
                                                      info, int val);
extern                                                              void
snd_seq_port_subscribe_set_queue(snd_seq_port_subscribe_t *
                                                  info, int q);
extern                                                              void
snd_seq_port_subscribe_set_sender(snd_seq_port_subscribe_t *
                                                   info,
                                                                      const
snd_seq_addr_t * addr);
extern                                                               void
snd_seq_port_subscribe_set_time_real(snd_seq_port_subscribe_t *
                                                      info, int val);
extern                                                               void
snd_seq_port_subscribe_set_time_update(snd_seq_port_subscribe_t
                                                          * info, int
val);
extern size_t snd_seq_port_subscribe_sizeof(void);
extern int snd_seq_query_next_client(snd_seq_t * handle,
                                           snd_seq_client_info_t *
info);
extern int snd_seq_query_next_port(snd_seq_t * handle,
                                     snd_seq_port_info_t * info);
extern int snd_seq_query_port_subscribers(snd_seq_t * seq,
                                           snd_seq_query_subscribe
_t *
                                          subs);
extern                                                              void
snd_seq_query_subscribe_copy(snd_seq_query_subscribe_t * dst,
                                                                     const
snd_seq_query_subscribe_t *
                                         src);
extern                                                               void
snd_seq_query_subscribe_free(snd_seq_query_subscribe_t * ptr);
extern                     const                    snd_seq_addr_t
*snd_seq_query_subscribe_get_addr(const
                                                                     snd
_seq_query_subscribe_t
                                                                      *
info);
extern int snd_seq_query_subscribe_get_exclusive(const
```

```
                                                    snd_seq_query_su
bscribe_t
                                                   * info);
extern int snd_seq_query_subscribe_get_index(const
                                                    snd_seq_query_subscr
ibe_t *
                                                   info);
extern int snd_seq_query_subscribe_get_queue(const
                                                    snd_seq_query_subscr
ibe_t *
                                                   info);
extern                      const                    snd_seq_addr_t
*snd_seq_query_subscribe_get_root(const
                                                                        snd
_seq_query_subscribe_t
                                                                         *
info);
extern int snd_seq_query_subscribe_get_time_real(const
                                                    snd_seq_query_su
bscribe_t
                                                   * info);
extern int snd_seq_query_subscribe_get_time_update(const
                                                      snd_seq_query_
subscribe_t
                                                     * info);
extern                                                   int
snd_seq_query_subscribe_malloc(snd_seq_query_subscribe_t *
                                              *ptr);
extern                                                  void
snd_seq_query_subscribe_set_index(snd_seq_query_subscribe_t *
                                               info, int _index);
extern                                                  void
snd_seq_query_subscribe_set_root(snd_seq_query_subscribe_t *
                                                   info,
                                                    const snd_seq_addr_t
* addr);
extern                                                  void
snd_seq_query_subscribe_set_type(snd_seq_query_subscribe_t *
                                                   info,
                                                    snd_seq_query_subs_t
ype_t
                                                   type);
extern size_t snd_seq_query_subscribe_sizeof(void);
extern  void  snd_seq_queue_status_copy(snd_seq_queue_status_t  *
dst,
                                                                  const
snd_seq_queue_status_t * src);
extern  void  snd_seq_queue_status_free(snd_seq_queue_status_t  *
ptr);
extern                  const                   snd_seq_real_time_t
*snd_seq_queue_status_get_real_time(const

snd_seq_queue_status_t

*

info);
extern                                             snd_seq_tick_time_t
snd_seq_queue_status_get_tick_time(const
                                                                        snd
_seq_queue_status_t
                                                                         *
info);
extern  int  snd_seq_queue_status_malloc(snd_seq_queue_status_t  *
*ptr);
extern size_t snd_seq_queue_status_sizeof(void);
```

```
extern void snd_seq_queue_tempo_copy(snd_seq_queue_tempo_t * dst,
                                     const snd_seq_queue_tempo_t
* src);
extern   void   snd_seq_queue_tempo_free(snd_seq_queue_tempo_t   *
ptr);
extern            int            snd_seq_queue_tempo_get_ppq(const
snd_seq_queue_tempo_t * info);
extern unsigned int snd_seq_queue_tempo_get_tempo(const
                                                   snd_seq_queue_t
empo_t *
                                                  info);
extern   int   snd_seq_queue_tempo_malloc(snd_seq_queue_tempo_t   *
*ptr);
extern  void  snd_seq_queue_tempo_set_ppq(snd_seq_queue_tempo_t  *
info,
                                          int ppq);
extern void snd_seq_queue_tempo_set_tempo(snd_seq_queue_tempo_t *
info,
                                           unsigned int tempo);
extern size_t snd_seq_queue_tempo_sizeof(void);
extern int snd_seq_set_client_info(snd_seq_t * handle,
                                   snd_seq_client_info_t * info);
extern  int   snd_seq_set_input_buffer_size(snd_seq_t   *   handle,
size_t size);
extern  int  snd_seq_set_output_buffer_size(snd_seq_t  *  handle,
size_t size);
extern int snd_seq_set_port_info(snd_seq_t * handle, int port,
                                 snd_seq_port_info_t * info);
extern int snd_seq_set_queue_tempo(snd_seq_t * handle, int q,
                                    snd_seq_queue_tempo_t *
tempo);
extern int snd_seq_subscribe_port(snd_seq_t * handle,
                                  snd_seq_port_subscribe_t *
sub);
extern int snd_seq_system_info(snd_seq_t * handle,
                               snd_seq_system_info_t * info);
extern void snd_seq_system_info_copy(snd_seq_system_info_t * dst,
                                     const snd_seq_system_info_t
* src);
extern   void   snd_seq_system_info_free(snd_seq_system_info_t   *
ptr);
extern          int          snd_seq_system_info_get_clients(const
snd_seq_system_info_t *
                                     info);
extern           int           snd_seq_system_info_get_ports(const
snd_seq_system_info_t *
                                    info);
extern           int           snd_seq_system_info_get_queues(const
snd_seq_system_info_t *
                                    info);
extern   int   snd_seq_system_info_malloc(snd_seq_system_info_t   *
*ptr);
extern size_t snd_seq_system_info_sizeof(void);
extern int snd_seq_unsubscribe_port(snd_seq_t * handle,
                                    snd_seq_port_subscribe_t *
sub);
```

6.2.17 alsa/seq_event.h

```
#define SND_SEQ_TIME_STAMP_TICK (0<<0)
#define SND_SEQ_TIME_MODE_ABS   (0<<1)
#define SND_SEQ_EVENT_LENGTH_FIXED      (0<<2)
#define SND_SEQ_PRIORITY_NORMAL (0<<4)
#define SND_SEQ_TIME_STAMP_MASK (1<<0)
```

```
#define SND_SEQ_TIME_STAMP_REAL (1<<0)
#define SND_SEQ_TIME_MODE_MASK  (1<<1)
#define SND_SEQ_TIME_MODE_REL   (1<<1)
#define SND_SEQ_EVENT_LENGTH_VARIABLE   (1<<2)
#define SND_SEQ_PRIORITY_HIGH   (1<<4)
#define SND_SEQ_PRIORITY_MASK   (1<<4)
#define SND_SEQ_EVENT_LENGTH_VARUSR     (2<<2)
#define SND_SEQ_EVENT_LENGTH_MASK       (3<<2)

typedef struct snd_seq_addr {
    unsigned char client;
    unsigned char port;
} snd_seq_addr_t;
typedef struct snd_seq_connect {
    snd_seq_addr_t sender;
    snd_seq_addr_t dest;
} snd_seq_connect_t;
typedef struct snd_seq_ev_ctrl {
    unsigned char channel;
    unsigned char unused[3];
    unsigned int param;
    int value;
} snd_seq_ev_ctrl_t;
typedef struct snd_seq_ev_ext {
    unsigned int len;
    void *ptr;
} __attribute__ ((packed)) snd_seq_ev_ext_t;
typedef struct snd_seq_ev_note {
    unsigned char channel;
    unsigned char note;
    unsigned char velocity;
    unsigned char off_velocity;
    unsigned int duration;
} snd_seq_ev_note_t;
typedef struct snd_seq_ev_queue_control {
    unsigned char queue;
    unsigned char unused[3];
    union {
        int value;              /* affected value (e.g. tempo) */
        snd_seq_timestamp_t time;       /* time */
        unsigned int position;  /* sync position */
        snd_seq_queue_skew_t skew;      /* queue skew */
        unsigned int d32[2];    /* any data */
        unsigned char d8[8];    /* any data */
    } param;
} snd_seq_ev_queue_control_t;
typedef struct snd_seq_ev_raw32 {
    unsigned int d[3];
} snd_seq_ev_raw32_t;
typedef struct snd_seq_ev_raw8 {
    unsigned char d[12];
} snd_seq_ev_raw8_t;
typedef struct snd_seq_event {
    snd_seq_event_type_t type;
    unsigned char flags;
    unsigned char tag;
    unsigned char queue;
    snd_seq_timestamp_t time;
    snd_seq_addr_t source;
    snd_seq_addr_t dest;
    union {
        snd_seq_ev_note_t note; /* note information */
             snd_seq_ev_ctrl_t  control;          /*  MIDI  control
information */
        snd_seq_ev_raw8_t raw8; /* raw8 data */
        snd_seq_ev_raw32_t raw32;       /* raw32 data */
```

```
        snd_seq_ev_ext_t ext;   /* external data */
         snd_seq_ev_queue_control_t queue;       /* queue control
*/
        snd_seq_timestamp_t time;       /* timestamp */
        snd_seq_addr_t addr;    /* address */
        snd_seq_connect_t connect;      /* connect information */
         snd_seq_result_t result;        /* operation result code
*/
    } data;
} snd_seq_event_t;
typedef unsigned char snd_seq_event_type_t;
typedef struct snd_seq_queue_skew {
    unsigned int value;
    unsigned int base;
} snd_seq_queue_skew_t;
union snd_seq_timestamp {
    snd_seq_tick_time_t tick;
    struct snd_seq_real_time time;
};
typedef struct snd_seq_real_time {
    unsigned int tv_sec;
    unsigned int tv_nsec;
} snd_seq_real_time_t;
typedef struct snd_seq_result {
    int event;
    int result;
} snd_seq_result_t;
typedef unsigned int snd_seq_tick_time_t;
typedef union snd_seq_timestamp {
    snd_seq_tick_time_t tick;
    struct snd_seq_real_time time;
} snd_seq_timestamp_t;
```

6.2.18 alsa/seq_midi_event.h

```
typedef struct snd_midi_event snd_midi_event_t;
extern long int snd_midi_event_decode(snd_midi_event_t * dev,
                                      unsigned char *buf, long
int count,
                                       const snd_seq_event_t *
ev);
extern long int snd_midi_event_encode(snd_midi_event_t * dev,
                                      const unsigned char *buf,
                                      long int count,
                                      snd_seq_event_t * ev);
extern int snd_midi_event_encode_byte(snd_midi_event_t * dev, int
c,
                                      snd_seq_event_t * ev);
extern void snd_midi_event_free(snd_midi_event_t * dev);
extern void snd_midi_event_init(snd_midi_event_t * dev);
extern int snd_midi_event_new(size_t bufsize, snd_midi_event_t *
*rdev);
extern void snd_midi_event_reset_decode(snd_midi_event_t * dev);
extern void snd_midi_event_reset_encode(snd_midi_event_t * dev);
```

6.2.19 alsa/seqmid.h

```
#define snd_seq_ev_set_dest(ev,c,p)      \
        ((ev)->dest.client = (c), (ev)->dest.port = (p))
#define snd_seq_ev_set_broadcast(ev)     \
           ((ev)->dest.client = SND_SEQ_ADDRESS_BROADCAST, (ev)-
>dest.port = \
        SND_SEQ_ADDRESS_BROADCAST)
```

```
#define snd_seq_ev_set_subs(ev)  \
          ((ev)->dest.client = SND_SEQ_ADDRESS_SUBSCRIBERS, (ev)-
>dest.port = \
        SND_SEQ_ADDRESS_UNKNOWN)
#define snd_seq_ev_set_fixed(ev)         \
        ((ev)->flags &= ~SND_SEQ_EVENT_LENGTH_MASK, (ev)->flags |
= \
        SND_SEQ_EVENT_LENGTH_FIXED)
#define snd_seq_ev_set_chanpress(ev,ch,val)      \
                        ((ev)->type   =   SND_SEQ_EVENT_CHANPRESS,
snd_seq_ev_set_fixed(ev), \
                      (ev)->data.control.channel   =   (ch),   (ev)-
>data.control.value = (val))
#define snd_seq_ev_set_controller(ev,ch,cc,val)  \
                        ((ev)->type   =   SND_SEQ_EVENT_CONTROLLER,
snd_seq_ev_set_fixed(ev), \
                      (ev)->data.control.channel   =   (ch),   (ev)-
>data.control.param = (cc), \
        (ev)->data.control.value = (val))
#define snd_seq_ev_set_keypress(ev,ch,key,vel)   \
                         ((ev)->type   =   SND_SEQ_EVENT_KEYPRESS,
snd_seq_ev_set_fixed(ev), \
          (ev)->data.note.channel = (ch), (ev)->data.note.note =
(key), \
        (ev)->data.note.velocity = (vel))
#define snd_seq_ev_set_pgmchange(ev,ch,val)      \
                        ((ev)->type   =   SND_SEQ_EVENT_PGMCHANGE,
snd_seq_ev_set_fixed(ev), \
                      (ev)->data.control.channel   =   (ch),   (ev)-
>data.control.value = (val))
#define snd_seq_ev_set_pitchbend(ev,ch,val)      \
                        ((ev)->type   =   SND_SEQ_EVENT_PITCHBEND,
snd_seq_ev_set_fixed(ev), \
                      (ev)->data.control.channel   =   (ch),   (ev)-
>data.control.value = (val))
#define  snd_seq_ev_set_direct(ev)               ((ev)->queue  =
SND_SEQ_QUEUE_DIRECT)
#define snd_seq_ev_set_source(ev,p)     ((ev)->source.port = (p))
#define snd_seq_ev_set_tag(ev,t)        ((ev)->tag = (t))
#define    snd_seq_ev_clear(ev)                 memset(ev,    0,
sizeof(snd_seq_event_t))

extern int snd_seq_connect_from(snd_seq_t * seq, int my_port,
                                int src_client, int src_port);
extern int snd_seq_connect_to(snd_seq_t * seq, int my_port,
                              int dest_client, int dest_port);
extern  int  snd_seq_control_queue(snd_seq_t  *  seq,  int  q,  int
type,
                                   int value, snd_seq_event_t *
ev);
extern int snd_seq_create_simple_port(snd_seq_t * seq, const char
*name,
                                    unsigned int caps,
                                    unsigned int type);
extern int snd_seq_delete_simple_port(snd_seq_t * seq, int port);
extern int snd_seq_disconnect_from(snd_seq_t * seq, int my_port,
                                   int src_client, int src_port);
extern int snd_seq_disconnect_to(snd_seq_t * seq, int my_port,
                                 int dest_client, int dest_port);
extern  int  snd_seq_parse_address(snd_seq_t  *  seq,  snd_seq_addr_t
* addr,
                                   const char *str);
extern  int  snd_seq_set_client_name(snd_seq_t  *  seq,  const  char
*name);
extern int snd_seq_sync_output_queue(snd_seq_t * seq);
```

6.2.20 alsa/timer.h

```
#define SND_TIMER_OPEN_NONBLOCK (1<<0)
#define SND_TIMER_OPEN_TREAD    (1<<1)
#define SND_TIMER_GLOBAL_SYSTEM 0
#define SND_TIMER_GLOBAL_RTC    1
#define SND_TIMER_GLOBAL_HPET   2

typedef struct sndrv_timer_ginfo snd_timer_ginfo_t;
typedef struct sndrv_timer_gparams snd_timer_gparams_t;
typedef struct sndrv_timer_gstatus snd_timer_gstatus_t;
typedef struct sndrv_timer_id snd_timer_id_t;
typedef struct sndrv_timer_info snd_timer_info_t;
typedef struct sndrv_timer_params snd_timer_params_t;
typedef struct _snd_timer_query snd_timer_query_t;
typedef struct sndrv_timer_status snd_timer_status_t;
typedef struct _snd_timer snd_timer_t;
typedef enum _snd_timer_type {
    SND_TIMER_TYPE_HW,
    SND_TIMER_TYPE_SHM = 1,
    SND_TIMER_TYPE_INET = 2
} snd_timer_type_t;
extern int snd_timer_close(snd_timer_t * handle);
extern int snd_timer_continue(snd_timer_t * handle);
extern void snd_timer_id_copy(snd_timer_id_t * dst,
                              const snd_timer_id_t * src);
extern void snd_timer_id_free(snd_timer_id_t * obj);
extern int snd_timer_id_get_card(snd_timer_id_t * id);
extern int snd_timer_id_get_class(snd_timer_id_t * id);
extern int snd_timer_id_get_device(snd_timer_id_t * id);
extern int snd_timer_id_get_sclass(snd_timer_id_t * id);
extern int snd_timer_id_get_subdevice(snd_timer_id_t * id);
extern int snd_timer_id_malloc(snd_timer_id_t * *ptr);
extern void snd_timer_id_set_card(snd_timer_id_t * id, int card);
extern  void  snd_timer_id_set_class(snd_timer_id_t  *  id,  int
dev_class);
extern  void  snd_timer_id_set_device(snd_timer_id_t  *  id,  int
device);
extern  void  snd_timer_id_set_sclass(snd_timer_id_t  *  id,  int
dev_sclass);
extern  void  snd_timer_id_set_subdevice(snd_timer_id_t  *  id,  int
subdevice);
extern size_t snd_timer_id_sizeof(void);
extern  int  snd_timer_info(snd_timer_t  *  handle,  snd_timer_info_t
* timer);
extern void snd_timer_info_copy(snd_timer_info_t * dst,
                                const snd_timer_info_t * src);
extern void snd_timer_info_free(snd_timer_info_t * obj);
extern int snd_timer_info_get_card(snd_timer_info_t * info);
extern   const   char   *snd_timer_info_get_id(snd_timer_info_t   *
info);
extern  const  char  *snd_timer_info_get_name(snd_timer_info_t  *
info);
extern  long  int  snd_timer_info_get_resolution(snd_timer_info_t  *
info);
extern int snd_timer_info_malloc(snd_timer_info_t * *ptr);
extern size_t snd_timer_info_sizeof(void);
extern  int  snd_timer_open(snd_timer_t  *  *handle,  const  char
*name,
                            int mode);
extern int snd_timer_params(snd_timer_t * handle,
                            snd_timer_params_t * params);
extern  long  int  snd_timer_params_get_ticks(snd_timer_params_t  *
params);
```

```
extern int snd_timer_params_malloc(snd_timer_params_t * *ptr);
extern  int  snd_timer_params_set_auto_start(snd_timer_params_t  *
params,
                                             int auto_start);
extern   void   snd_timer_params_set_ticks(snd_timer_params_t   *
params,
                                           long int ticks);
extern int snd_timer_poll_descriptors(snd_timer_t * handle,
                                      struct pollfd *pfds,
                                      unsigned int space);
extern    int    snd_timer_poll_descriptors_count(snd_timer_t    *
handle);
extern ssize_t snd_timer_read(snd_timer_t * handle, void *buffer,
                              size_t size);
extern int snd_timer_start(snd_timer_t * handle);
extern int snd_timer_status(snd_timer_t * handle,
                            snd_timer_status_t * status);
extern int snd_timer_stop(snd_timer_t * handle);
```

III Network Security Services

7 Libraries

7.1 Interfaces for libnspr4

Table 7-1 defines the library name and shared object name for the libnspr4 library

Table 7-1 libnspr4 Definition

Library:	libnspr4
SONAME:	libnspr4.so

The behavior of the interfaces in this library is specified by the following specifications:

[NSPR] NSPR Reference

7.1.1 Netscape Portable Runtime

7.1.1.1 Interfaces for Netscape Portable Runtime

An LSB conforming implementation shall provide the generic functions for Netscape Portable Runtime specified in Table 7-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 7-2 libnspr4 - Netscape Portable Runtime Function Interfaces

PR_Accept [NSPR]	PR_Bind [NSPR]	PR_Cleanup [NSPR]
PR_Close [NSPR]	PR_Connect [NSPR]	PR_CreateIOLayerStub [NSPR]
PR_EnumerateAddrInfo [NSPR]	PR_FreeAddrInfo [NSPR]	PR_GetAddrInfoByName [NSPR]
PR_GetDefaultIOMethods [NSPR]	PR_GetError [NSPR]	PR_GetLayersIdentity [NSPR]
PR_GetSocketOption [NSPR]	PR_GetUniqueIdentity [NSPR]	PR_ImportTCPSocket [NSPR]
PR_Interrupt [NSPR]	PR_Listen [NSPR]	PR_MillisecondsToInterval [NSPR]
PR_NetAddrToString [NSPR]	PR_Now [NSPR]	PR_OpenTCPSocket [NSPR]
PR_OpenUDPSocket [NSPR]	PR_Poll [NSPR]	PR_PopIOLayer [NSPR]
PR_PushIOLayer [NSPR]	PR_Read [NSPR]	PR_Recv [NSPR]
PR_RecvFrom [NSPR]	PR_SecondsToInterval [NSPR]	PR_Send [NSPR]
PR_SendTo [NSPR]	PR_SetError [NSPR]	PR_SetSocketOption [NSPR]
PR_Shutdown [NSPR]	PR_StringToNetAddr [NSPR]	PR_Write [NSPR]

7.2 Data Definitions for libnspr4

This section defines global identifiers and their values that are associated with interfaces contained in libnspr4. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

7.2.1 nspr4/nspr.h

```
#define nspr_h___
```

7.2.2 nspr4/plarena.h

```
#define plarena_h___

typedef struct PLArenaPool {
    struct PLArena first;
    struct PLArena *current;
    PRUint32 arenasize;
    PRUword mask;
} PLArenaPool;
struct PLArena {
    struct PLArena *next;
    PRUword base;
    PRUword limit;
    PRUword avail;
};
```

7.2.3 nspr4/plhash.h

```
#define plhash_h___

typedef PRUint32 PLHashNumber;
typedef PRIntn(*PLHashComparator) (const void *, const void *);
typedef struct PLHashAllocOps {
    void *(*allocTable) (void *, PRSize);
    void (*freeTable) (void *, void *);
    struct PLHashEntry *(*allocEntry) (void *, const void *);
    void (*freeEntry) (void *, struct PLHashEntry *, PRUintn);
} PLHashAllocOps;
typedef PLHashNumber(*PLHashFunction) (const void *);
struct PLHashEntry {
    struct PLHashEntry *next;
    PLHashNumber keyHash;
    const void *key;
    void *value;
```



```
};
struct PLHashTable {
    struct PLHashEntry **buckets;
    PRUint32 nentries;
    PRUint32 shift;
    PLHashFunction keyHash;
    PLHashComparator keyCompare;
    PLHashComparator valueCompare;
    const PLHashAllocOps *allocOps;
    void *allocPriv;
};
```

7.2.4 nspr4/prclist.h

```
#define prclist_h___

typedef struct PRCListStr {
    PRCList *next;
    PRCList *prev;
} PRCList;
```

7.2.5 nspr4/prerror.h

```
#define prerror_h___

typedef PRInt32 PRErrorCode;
extern PRErrorCode PR_GetError(void);
extern void PR_SetError(PRErrorCode errorCode, PRInt32 oserr);
```

7.2.6 nspr4/prinit.h

```
#define prinit_h___

extern PRStatus PR_Cleanup(void);
```

7.2.7 nspr4/prinrval.h

```
#define prinrval_h

typedef PRUint32 PRIntervalTime;
extern PRIntervalTime PR_MillisecondsToInterval(PRUint32 milli);
extern PRIntervalTime PR_SecondsToInterval(PRUint32 seconds);
```

7.2.8 nspr4/prio.h

```
#define prio_h___

typedef enum PRDescType {
    PR_DESC_FILE = 1,
    PR_DESC_SOCKET_TCP = 2,
    PR_DESC_SOCKET_UDP = 3,
    PR_DESC_LAYERED = 4,
    PR_DESC_PIPE = 5
} PRDescType;
typedef struct PRIPv6Addr {
    union {
        PRUint8 _S6_u8[15];
        PRUint16 _S6_u16[7];
```

```
        PRUint32 _S6_u32[3];
        PRUint64 _S6_u64[1];
    } _S6_un;
} PRIPv6Addr;
typedef enum PRTransmitFileFlags {
    PR_TRANSMITFILE_KEEP_OPEN,
    PR_TRANSMITFILE_CLOSE_SOCKET = 1
} PRTransmitFileFlags;
typedef struct PRLinger {
    PRBool polarity;
    PRIntervalTime linger;
} PRLinger;
typedef struct PRFilePrivate PRFilePrivate;
typedef struct PRFileDesc {
    const struct PRIOMethods *methods;
    PRFilePrivate *secret;
    PRFileDesc *lower;
    PRFileDesc *higher;
    void (*dtor) (PRFileDesc *);
    PRDescIdentity identity;
} PRFileDesc;
typedef union PRNetAddr {
    struct {
        PRUint16 family;
        char data[14];
    } raw;
    struct {
        PRUint16 family;
        PRUint16 port;
        PRUint32 ip;
        char pad[7];
    } inet;
    struct {
        PRUint16 family;
        PRUint16 port;
        PRUint32 flowinfo;
        PRIPv6Addr ip;
        PRUint32 scope_id;
    } ipv6;
    struct {
        PRUint16 family;
        char path[103];
    } local;
} PRNetAddr;
typedef struct PRMcastRequest {
    union PRNetAddr mcaddr;
    union PRNetAddr ifaddr;
} PRMcastRequest;
typedef struct PRIOVec {
    char *iov_base;
    int iov_len;
} PRIOVec;
typedef struct PRSocketOptionData {
    PRSockOption option;
    union {
        PRUintn ip_ttl;
        PRUintn mcast_ttl;
        PRUintn tos;
        PRBool non_blocking;
        PRBool reuse_addr;
        PRBool keep_alive;
        PRBool mcast_loopback;
        PRBool no_delay;
        PRBool broadcast;
        PRSize max_segment;
        PRSize recv_buffer_size;
```

```
        PRSize send_buffer_size;
        PRLinger linger;
        PRMcastRequest add_member;
        PRMcastRequest drop_member;
        union PRNetAddr mcast_if;
    } value;
} PRSocketOptionData;
typedef PRStatus(*PRFsyncFN) (PRFileDesc *);
typedef PRStatus(*PRListenFN) (PRFileDesc *, PRIntn);
typedef enum PRSeekWhence {
    PR_SEEK_SET,
    PR_SEEK_CUR = 1,
    PR_SEEK_END = 2
} PRSeekWhence;
typedef PRInt32(*PRAcceptreadFN) (PRFileDesc *, PRFileDesc * *,
                                  PRNetAddr * *, void *, PRInt32,
                                  PRIntervalTime);
typedef PRStatus(*PRCloseFN) (PRFileDesc *);
typedef PRInt32(*PRTransmitfileFN) (PRFileDesc *, PRFileDesc *,
                                    const void *, PRInt32,
                                             PRTransmitFileFlags,
PRIntervalTime);
typedef enum PRSockOption {
    PR_SockOpt_Nonblocking,
    PR_SockOpt_Linger = 1,
    PR_SockOpt_Reuseaddr = 2,
    PR_SockOpt_Keepalive = 3,
    PR_SockOpt_RecvBufferSize = 4,
    PR_SockOpt_SendBufferSize = 5,
    PR_SockOpt_IpTimeToLive = 6,
    PR_SockOpt_IpTypeOfService = 7,
    PR_SockOpt_AddMember = 8,
    PR_SockOpt_DropMember = 9,
    PR_SockOpt_McastInterface = 10,
    PR_SockOpt_McastTimeToLive = 11,
    PR_SockOpt_McastLoopback = 12,
    PR_SockOpt_NoDelay = 13,
    PR_SockOpt_MaxSegment = 14,
    PR_SockOpt_Broadcast = 15,
    PR_SockOpt_Last = 16
} PRSockOption;
typedef PRFileDesc *(*PRAcceptFN) (PRFileDesc *, PRNetAddr *,
                                   PRIntervalTime);
typedef PRStatus(*PRConnectcontinueFN) (PRFileDesc *, PRInt16);
typedef PRInt32(*PRReadFN) (PRFileDesc *, void *, PRInt32);
typedef struct PRFileInfo64 {
    PRFileType type;
    PROffset64 size;
    PRTime creationTime;
    PRTime modifyTime;
} PRFileInfo64;
typedef PRStatus(*PRGetsocketoptionFN) (PRFileDesc *,
                                        PRSocketOptionData *);
typedef  PRInt32(*PRSendtoFN)  (PRFileDesc  *,  const  void  *,
PRInt32, PRIntn,
                                const PRNetAddr *, PRIntervalTime);
typedef PRStatus(*PRGetsocknameFN) (PRFileDesc *, PRNetAddr *);
typedef PRInt32(*PRSendFN) (PRFileDesc *, const void *, PRInt32,
PRIntn,
                               PRIntervalTime);
typedef   PROffset32(*PRSeekFN)   (PRFileDesc   *,   PROffset32,
PRSeekWhence);
typedef PRInt64(*PRAvailable64FN) (PRFileDesc *);
typedef PRInt32(*PRAvailableFN) (PRFileDesc *);
typedef struct PRFileInfo {
    PRFileType type;
```

```
    PROffset32 size;
    PRTime creationTime;
    PRTime modifyTime;
} PRFileInfo;
typedef   PROffset64(*PRSeek64FN)   (PRFileDesc   *,   PROffset64,
PRSeekWhence);
typedef PRStatus(*PRSetsocketoptionFN) (PRFileDesc *,
                                        const PRSocketOptionData
*);
typedef  PRInt32(*PRRecvFN)  (PRFileDesc  *,  void  *,  PRInt32,
PRIntn,
                              PRIntervalTime);
typedef struct PRSendFileData {
    PRFileDesc *fd;
    PRUint32 file_offset;
    PRSize file_nbytes;
    const void *header;
    PRInt32 hlen;
    const void *trailer;
    PRInt32 tlen;
} PRSendFileData;
typedef PRIntn PRDescIdentity;
typedef PRStatus(*PRConnectFN) (PRFileDesc *, const PRNetAddr *,
                                PRIntervalTime);
typedef PRInt32(*PRSendfileFN) (PRFileDesc *, PRSendFileData *,
                                           PRTransmitFileFlags,
PRIntervalTime);
typedef  PRInt32(*PRRecvfromFN)  (PRFileDesc  *,  void  *,  PRInt32,
PRIntn,
                                  PRNetAddr *, PRIntervalTime);
typedef struct PRPollDesc {
    PRFileDesc *fd;
    PRInt16 in_flags;
    PRInt16 out_flags;
} PRPollDesc;
typedef   PRInt32(*PRWriteFN)   (PRFileDesc   *,   const   void   *,
PRInt32);
typedef PRStatus(*PRFileInfo64FN) (PRFileDesc *, PRFileInfo64 *);
typedef PRStatus(*PRShutdownFN) (PRFileDesc *, PRIntn);
typedef PRIntn(*PRReservedFN) (PRFileDesc *);
typedef PRStatus(*PRFileInfoFN) (PRFileDesc *, PRFileInfo *);
typedef  PRInt32(*PRWritevFN)  (PRFileDesc  *,  const  PRIOVec  *,
PRInt32,
                                PRIntervalTime);
typedef enum PRFileType {
    PR_FILE_FILE = 1,
    PR_FILE_DIRECTORY = 2,
    PR_FILE_OTHER = 3
} PRFileType;
typedef PRStatus(*PRBindFN) (PRFileDesc *, const PRNetAddr *);
typedef PRInt16(*PRPollFN) (PRFileDesc *, PRInt16, PRInt16 *);
struct PRIOMethods {
    PRDescType file_type;
    PRCloseFN close;
    PRReadFN read;
    PRWriteFN write;
    PRAvailableFN available;
    PRAvailable64FN available64;
    PRFsyncFN fsync;
    PRSeekFN seek;
    PRSeek64FN seek64;
    PRFileInfoFN fileInfo;
    PRFileInfo64FN fileInfo64;
    PRWritevFN writev;
    PRConnectFN connect;
    PRAcceptFN accept;
```



```
    PRBindFN bind;
    PRListenFN listen;
    PRShutdownFN shutdown;
    PRRecvFN recv;
    PRSendFN send;
    PRRecvfromFN recvfrom;
    PRSendtoFN sendto;
    PRPollFN poll;
    PRAcceptreadFN acceptread;
    PRTransmitfileFN transmitfile;
    PRGetsocknameFN getsockname;
    PRGetpeernameFN getpeername;
    PRReservedFN reserved_fn_6;
    PRReservedFN reserved_fn_5;
    PRGetsocketoptionFN getsocketoption;
    PRSetsocketoptionFN setsocketoption;
    PRSendfileFN sendfile;
    PRConnectcontinueFN connectcontinue;
    PRReservedFN reserved_fn_3;
    PRReservedFN reserved_fn_2;
    PRReservedFN reserved_fn_1;
    PRReservedFN reserved_fn_0;
};
typedef PRStatus(*PRGetpeernameFN) (PRFileDesc *, PRNetAddr *);
typedef enum PRShutdownHow {
    PR_SHUTDOWN_RCV,
    PR_SHUTDOWN_SEND = 1,
    PR_SHUTDOWN_BOTH = 2
} PRShutdownHow;
extern PRFileDesc *PR_Accept(PRFileDesc * fd, PRNetAddr * addr,
                             PRIntervalTime timeout);
extern PRStatus PR_Bind(PRFileDesc * fd, const PRNetAddr * addr);
extern PRStatus PR_Close(PRFileDesc * fd);
extern PRStatus PR_Connect(PRFileDesc * fd, const PRNetAddr *
addr,
                           PRIntervalTime timeout);
extern PRFileDesc *PR_CreateIOLayerStub(PRDescIdentity ident,
                                        const struct PRIOMethods
*methods);
extern const struct PRIOMethods *PR_GetDefaultIOMethods(void);
extern PRDescIdentity PR_GetLayersIdentity(PRFileDesc * fd);
extern PRStatus PR_GetSocketOption(PRFileDesc * fd,
                                   PRSocketOptionData * data);
extern     PRDescIdentity     PR_GetUniqueIdentity(const     char
*layer_name);
extern PRStatus PR_Listen(PRFileDesc * fd, PRIntn backlog);
extern PRFileDesc *PR_OpenTCPSocket(PRIntn af);
extern PRFileDesc *PR_OpenUDPSocket(PRIntn af);
extern PRInt32 PR_Poll(PRPollDesc * pds, PRIntn npds,
                       PRIntervalTime timeout);
extern    PRFileDesc    *PR_PopIOLayer(PRFileDesc    *    fd_stack,
PRDescIdentity id);
extern    PRStatus    PR_PushIOLayer(PRFileDesc    *    fd_stack,
PRDescIdentity id,
                                 PRFileDesc * layer);
extern  PRInt32  PR_Read(PRFileDesc  *  fd,  void  *buf,  PRInt32
amount);
extern  PRInt32  PR_Recv(PRFileDesc  *  fd,  void  *buf,  PRInt32
amount,
                         PRIntn flags, PRIntervalTime timeout);
extern PRInt32 PR_RecvFrom(PRFileDesc * fd, void *buf, PRInt32
amount,
                           PRIntn flags, PRNetAddr * addr,
                           PRIntervalTime timeout);
extern PRInt32 PR_Send(PRFileDesc * fd, const void *buf, PRInt32
amount,
```

```
                        PRIntn flags, PRIntervalTime timeout);
extern  PRInt32  PR_SendTo(PRFileDesc  *  fd,  const  void  *buf,
PRInt32 amount,
                          PRIntn flags, const PRNetAddr * addr,
                          PRIntervalTime timeout);
extern PRStatus PR_SetSocketOption(PRFileDesc * fd,
                                           const PRSocketOptionData *
data);
extern PRStatus PR_Shutdown(PRFileDesc * fd, PRShutdownHow how);
extern PRInt32 PR_Write(PRFileDesc * fd, const void *buf, PRInt32
amount);
```

7.2.9 nspr4/private/pprio.h

```
#define pprio_h___

typedef PRInt32 PROsfd;
extern PRFileDesc *PR_ImportTCPSocket(PROsfd osfd);
```

7.2.10 nspr4/prlock.h

```
#define prlock_h___

typedef struct PRLock PRLock;
```

7.2.11 nspr4/prmon.h

```
#define prmon_h___

typedef struct PRMonitor PRMonitor;
```

7.2.12 nspr4/prnetdb.h

```
#define prnetdb_h___

typedef struct PRHostEnt {
    char *h_name;
    char **h_aliases;
    PRInt32 h_addrtype;
    PRInt32 h_length;
    char **h_addr_list;
} PRHostEnt;
typedef struct PRAddrInfo PRAddrInfo;
extern void *PR_EnumerateAddrInfo(void *enumPtr,
                                  const PRAddrInfo * addrInfo,
                                       PRUint16 port, PRNetAddr *
result);
extern void PR_FreeAddrInfo(PRAddrInfo * addrInfo);
extern  PRAddrInfo  *PR_GetAddrInfoByName(const  char  *hostname,
PRUint16 af,
                                          PRIntn flags);
extern  PRStatus  PR_NetAddrToString(const  PRNetAddr  *  addr,  char
*string,
                                    PRUint32 size);
extern  PRStatus  PR_StringToNetAddr(const  char  *string,  PRNetAddr
* addr);
```


7.2.13 nspr4/prthread.h

```
#define prthread_h___

typedef struct PRThread PRThread;
extern PRStatus PR_Interrupt(PRThread * thread);
```

7.2.14 nspr4/prtime.h

```
#define prtime_h___

typedef PRInt64 PRTime;
extern PRTime PR_Now(void);
```

7.2.15 nspr4/prtypes.h

```
#define prtypes_h___

typedef int PRInt32;
typedef unsigned long int PRUword;
typedef int PRIntn;
typedef unsigned long int PRUint64;
typedef unsigned char PRUint8;
typedef short int PRInt16;
typedef long int PRInt64;
typedef PRIntn PRBool;
typedef unsigned short PRUint16;
typedef unsigned int PRUint32;
typedef size_t PRSize;
typedef unsigned int PRUintn;
typedef PRInt64 PROffset64;
typedef PRInt32 PROffset32;
typedef enum {
    PR_FAILURE = -1,
    PR_SUCCESS
} PRStatus;
```

7.3 Interfaces for libnss3

Table 7-3 defines the library name and shared object name for the libnss3 library

Table 7-3 libnss3 Definition

Library:	libnss3
SONAME:	libnss3.so

The behavior of the interfaces in this library is specified by the following specifications:

[NSS SSL] Mozilla's NSS SSL Reference

7.3.1 NSS Utility

7.3.1.1 Interfaces for NSS Utility

An LSB conforming implementation shall provide the generic functions for NSS Utility specified in Table 7-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 7-4 libnss3 - NSS Utility Function Interfaces

CERT_CheckCertValidTimes(NSS_3.2) [NSS SSL]	CERT_DestroyCertificate(NSS_3.2) [NSS SSL]	CERT_DupCertificate(NSS_3.2) [NSS SSL]
CERT_FreeNicknames(NSS_3.2) [NSS SSL]	CERT_GetCertNicknames(NSS_3.2) [NSS SSL]	CERT_GetDefaultCertDB(NSS_3.2) [NSS SSL]
CERT_VerifyCertName(NSS_3.2) [NSS SSL]	CERT_VerifyCertNow(NSS_3.2) [NSS SSL]	NSS_Init(NSS_3.2) [NSS SSL]
NSS_InitReadWrite(NSS_3.2) [NSS SSL]	NSS_NoDB_Init(NSS_3.2) [NSS SSL]	NSS_Shutdown(NSS_3.2) [NSS SSL]
PK11_FindCertFromNickname(NSS_3.2) [NSS SSL]	PK11_FindKeyByAnyCert(NSS_3.2) [NSS SSL]	PK11_GetSlotName(NSS_3.2) [NSS SSL]
PK11_GetTokenName(NSS_3.2) [NSS SSL]	PK11_IsHW(NSS_3.2) [NSS SSL]	PK11_IsPresent(NSS_3.2) [NSS SSL]
PK11_IsReadOnly(NSS_3.2) [NSS SSL]	PK11_SetPasswordFunc(NSS_3.2) [NSS SSL]	SECKEY_DestroyPrivateKey(NSS_3.2) [NSS SSL]

7.4 Data Definitions for libnss3

This section defines global identifiers and their values that are associated with interfaces contained in libnss3. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

7.4.1 nss3/blapit.h

```
#define _BLAPIT_H_
#define PQG_PBITS_TO_INDEX(bits)         \
   (((bits) < 512 || (bits) > 1024 || (bits) % 64) ? -1 : \
 (int)((bits)-512)/64)
#define PQG_INDEX_TO_PBITS(j)    \
   (((unsigned)(j) > 8) ? -1 : (512 + 64 * (j)))
#define NSS_AES 0
#define NSS_DES 0
#define NSS_RC2 0
#define NSS_RC5 0
#define NSS_AES_CBC     1
#define NSS_DES_CBC     1
#define NSS_RC2_CBC     1
```



```
#define NSS_RC5_CBC     1
#define DSA_MAX_P_BITS  1024
#define DH_MIN_P_BITS   128
#define RSA_MIN_MODULUS_BITS    128
#define AES_BLOCK_SIZE  16
#define DSA_Q_BITS      160
#define NSS_DES_EDE3    2
#define DSA_SUBPRIME_LEN        20
#define NSS_FREEBL_DEFAULT_CHUNKSIZE    2048
#define DH_MAX_P_BITS   2236
#define NSS_DES_EDE3_CBC        3
#define DSA_SIGNATURE_LEN       40
#define DSA_MIN_P_BITS  512
#define AES_KEY_WRAP_BLOCK_SIZE 8
#define AES_KEY_WRAP_IV_BYTES   8
#define DES_KEY_LENGTH  8

typedef struct PQGParamsStr {
    PLArenaPool *arena;
    SECItem prime;
    SECItem subPrime;
    SECItem base;
} PQGParams;
typedef struct PQGVerifyStr {
    PLArenaPool *arena;
    unsigned int counter;
    SECItem seed;
    SECItem h;
} PQGVerify;
```

7.4.2 nss3/cert.h

```
#define _CERT_H_

extern                                          SECCertTimeValidity
CERT_CheckCertValidTimes(CERTCertificate * cert,
                                                        PRTime t,
                                                                PRBool
allowOverride);
extern void CERT_DestroyCertificate(CERTCertificate * cert);
extern CERTCertificate *CERT_DupCertificate(CERTCertificate * c);
extern void CERT_FreeNicknames(CERTCertNicknames * nicknames);
extern  CERTCertNicknames  *CERT_GetCertNicknames(CERTCertDBHandle
* handle,
                                                       int what, void
*wincx);
extern CERTCertDBHandle *CERT_GetDefaultCertDB(void);
extern SECStatus CERT_VerifyCertName(CERTCertificate * cert,
                                     const char *hostname);
extern SECStatus CERT_VerifyCertNow(CERTCertDBHandle * handle,
                                    CERTCertificate * cert,
                                    PRBool checkSig,
                                      SECCertUsage certUsage, void
*wincx);
```

7.4.3 nss3/certt.h

```
#define _CERTT_H_
#define NS_CERT_TYPE_CA  \
        ( NS_CERT_TYPE_SSL_CA | NS_CERT_TYPE_EMAIL_CA | \
                                NS_CERT_TYPE_OBJECT_SIGNING_CA    |
EXT_KEY_USAGE_STATUS_RESPONDER )
#define NS_CERT_TYPE_APP         \
```

```
        ( NS_CERT_TYPE_SSL_CLIENT | NS_CERT_TYPE_SSL_SERVER | \
        NS_CERT_TYPE_EMAIL | NS_CERT_TYPE_OBJECT_SIGNING )
#define SEC_GET_TRUST_FLAGS(trust,type)  \
        (((type)==trustSSL)?((trust)->sslFlags): \
        (((type)==trustEmail)?((trust)->emailFlags): \
                        (((type)==trustObjectSigning)?((trust)-
>objectSigningFlags):0)))
#define KU_ALL   \
                  (KU_DIGITAL_SIGNATURE  |  KU_NON_REPUDIATION  |
KU_KEY_ENCIPHERMENT | \
                   KU_DATA_ENCIPHERMENT   |   KU_KEY_AGREEMENT   |
KU_KEY_CERT_SIGN | \
        KU_CRL_SIGN)
#define  CERT_LIST_END(n,l)         (((void  *)n)  ==  ((void  *)&l-
>list))
#define  CERT_LIST_NEXT(n)               ((CERTCertListNode  *)n-
>links.next)
#define   CERT_LIST_HEAD(l)                    ((CERTCertListNode
*)PR_LIST_HEAD(&l->list))
#define certificateUsageSSLClient       (0x0001)
#define certificateUsageSSLServer       (0x0002)
#define certificateUsageSSLServerWithStepUp     (0x0004)
#define certificateUsageSSLCA   (0x0008)
#define certificateUsageEmailSigner     (0x0010)
#define certificateUsageEmailRecipient  (0x0020)
#define certificateUsageObjectSigner    (0x0040)
#define certificateUsageUserCertImport  (0x0080)
#define NS_CERT_TYPE_OBJECT_SIGNING_CA  (0x01)
#define certificateUsageVerifyCA        (0x0100)
#define KU_CRL_SIGN     (0x02)
#define NS_CERT_TYPE_EMAIL_CA   (0x02)
#define RF_CERTIFICATE_HOLD     (0x02)
#define certificateUsageProtectedObjectSigner   (0x0200)
#define KU_KEY_CERT_SIGN        (0x04)
#define NS_CERT_TYPE_SSL_CA     (0x04)
#define RF_CESSATION_OF_OPERATION       (0x04)
#define certificateUsageStatusResponder (0x0400)
#define KU_KEY_AGREEMENT        (0x08)
#define NS_CERT_TYPE_RESERVED   (0x08)
#define RF_SUPERSEDED   (0x08)
#define certificateUsageAnyCA   (0x0800)
#define KU_DATA_ENCIPHERMENT    (0x10)
#define NS_CERT_TYPE_OBJECT_SIGNING     (0x10)
#define RF_AFFILIATION_CHANGED  (0x10)
#define KU_KEY_ENCIPHERMENT     (0x20)
#define NS_CERT_TYPE_EMAIL      (0x20)
#define RF_CA_COMPROMISE        (0x20)
#define KU_NON_REPUDIATION      (0x40)
#define NS_CERT_TYPE_SSL_SERVER (0x40)
#define RF_KEY_COMPROMISE       (0x40)
#define EXT_KEY_USAGE_STATUS_RESPONDER  (0x4000)
#define KU_KEY_AGREEMENT_OR_ENCIPHERMENT        (0x4000)
#define KU_DIGITAL_SIGNATURE    (0x80)
#define NS_CERT_TYPE_SSL_CLIENT (0x80)
#define RF_UNUSED       (0x80)
#define EXT_KEY_USAGE_TIME_STAMP        (0x8000)
#define KU_NS_GOVT_APPROVED     (0x8000)
#define CERT_UNLIMITED_PATH_CONSTRAINT  -2
#define SEC_CERTIFICATE_REQUEST_VERSION 0
#define SEC_CERTIFICATE_VERSION_1       0
#define SEC_CRL_VERSION_1       0
#define SEC_CERTIFICATE_VERSION_2       1
#define SEC_CERT_CLASS_CA       1
#define SEC_CERT_NICKNAMES_ALL  1
#define SEC_CRL_VERSION_2       1
#define SEC_CERTIFICATE_VERSION_3       2
```



```
#define SEC_CERT_CLASS_SERVER   2
#define SEC_CERT_NICKNAMES_USER 2
#define CERT_MAX_CERT_CHAIN     20
#define SEC_CERT_CLASS_USER     3
#define SEC_CERT_NICKNAMES_SERVER       3
#define SEC_CERT_CLASS_EMAIL    4
#define SEC_CERT_NICKNAMES_CA   4
#define certificateUsageHighest certificateUsageAnyCA
#define CERT_LIST_EMPTY(l)       CERT_LIST_END(CERT_LIST_HEAD(l),
l)

typedef struct CERTAVAStr {
    SECItem type;
    SECItem value;
} CERTAVA;
typedef struct CERTAttributeStr {
    SECItem attrType;
    SECItem **attrValue;
} CERTAttribute;
typedef struct CERTAuthInfoAccessStr {
    SECItem method;
    SECItem derLocation;
    CERTGeneralName *location;
} CERTAuthInfoAccess;
typedef struct CERTAuthKeyIDStr {
    SECItem keyID;
    CERTGeneralName *authCertIssuer;
    SECItem authCertSerialNumber;
    SECItem **DERAuthCertIssuer;
} CERTAuthKeyID;
typedef struct CERTBasicConstraintsStr {
    PRBool isCA;
    int pathLenConstraint;
} CERTBasicConstraints;
typedef struct NSSTrustDomainStr CERTCertDBHandle;
typedef struct CERTCertExtensionStr {
    SECItem id;
    SECItem critical;
    SECItem value;
} CERTCertExtension;
typedef struct CERTCertListStr {
    PRCList list;
    PLArenaPool *arena;
} CERTCertList;
typedef struct CERTCertListNodeStr {
    PRCList links;
    CERTCertificate *cert;
    void *appData;
} CERTCertListNode;
typedef struct CERTCertNicknamesStr {
    PLArenaPool *arena;
    void *head;
    int numnicknames;
    char **nicknames;
    int what;
    int totallen;
} CERTCertNicknames;
typedef struct CERTCertTrustStr {
    unsigned int sslFlags;
    unsigned int emailFlags;
    unsigned int objectSigningFlags;
} CERTCertTrust;
typedef struct CERTSignedDataStr {
    SECItem data;
    SECAlgorithmID signatureAlgorithm;
    SECItem signature;
```

```
} CERTSignedData;
typedef struct CERTCertificateListStr {
    SECItem *certs;
    int len;
    PLArenaPool *arena;
} CERTCertificateList;
typedef struct CERTNameStr {
    PLArenaPool *arena;
    CERTRDN **rdns;
} CERTName;
typedef struct CERTCrlStr {
    PLArenaPool *arena;
    SECItem version;
    SECAlgorithmID signatureAlg;
    SECItem derName;
    CERTName name;
    SECItem lastUpdate;
    SECItem nextUpdate;
    CERTCrlEntry **entries;
    CERTCertExtension **extensions;
} CERTCrl;
typedef struct CERTCrlDistributionPointsStr {
    CRLDistributionPoint **distPoints;
} CERTCrlDistributionPoints;
typedef struct CERTCrlEntryStr {
    SECItem serialNumber;
    SECItem revocationDate;
    CERTCertExtension **extensions;
} CERTCrlEntry;
typedef struct CERTCrlHeadNodeStr {
    PLArenaPool *arena;
    CERTCertDBHandle *dbhandle;
    CERTCrlNode *first;
    CERTCrlNode *last;
} CERTCrlHeadNode;
typedef struct CERTCrlNodeStr {
    CERTCrlNode *next;
    int type;
    CERTSignedCrl *crl;
} CERTCrlNode;
typedef struct CERTDistNamesStr {
    PLArenaPool *arena;
    int nnames;
    SECItem *names;
    void *head;
} CERTDistNames;
typedef struct OtherNameStr {
    SECItem name;
    SECItem oid;
} OtherName;
typedef struct CERTGeneralNameListStr {
    PLArenaPool *arena;
    CERTGeneralName *name;
    int refCount;
    int len;
    PRLock *lock;
} CERTGeneralNameList;
typedef struct CERTIssuerAndSNStr {
    SECItem derIssuer;
    CERTName issuer;
    SECItem serialNumber;
} CERTIssuerAndSN;
typedef struct CERTSubjectPublicKeyInfoStr {
    PLArenaPool *arena;
    SECAlgorithmID algorithm;
    SECItem subjectPublicKey;
```

```
} CERTSubjectPublicKeyInfo;
typedef struct CERTGeneralNameStr {
    CERTGeneralNameType type;
    union {
        CERTName directoryName;
        OtherName OthName;
        SECItem other;
    } name;
    SECItem derDirectoryName;
    PRCList l;
} CERTGeneralName;
typedef struct CERTNameConstraintsStr {
    CERTNameConstraint *permited;
    CERTNameConstraint *excluded;
    SECItem **DERPermited;
    SECItem **DERExcluded;
} CERTNameConstraints;
typedef struct CERTOKDomainNameStr {
    CERTOKDomainName *next;
    char name[1];
} CERTOKDomainName;
typedef struct CERTPrivKeyUsagePeriodStr {
    SECItem notBefore;
    SECItem notAfter;
    PLArenaPool *arena;
} CERTPrivKeyUsagePeriod;
typedef struct CERTRDNStr {
    CERTAVA **avas;
} CERTRDN;
typedef struct CERTSignedCrlStr {
    PLArenaPool *arena;
    CERTCrl crl;
    void *reserved1;
    PRBool reserved2;
    PRBool isperm;
    PRBool istemp;
    int referenceCount;
    CERTCertDBHandle *dbhandle;
    CERTSignedData signatureWrap;
    char *url;
    SECItem *derCrl;
    PK11SlotInfo *slot;
    CK_OBJECT_HANDLE pkcs11ID;
    void *opaque;
} CERTSignedCrl;
typedef struct CERTValidityStr {
    PLArenaPool *arena;
    SECItem notBefore;
    SECItem notAfter;
} CERTValidity;
typedef struct CERTStatusConfigStr {
    CERTStatusChecker statusChecker;
    CERTStatusDestroy statusDestroy;
    void *statusContext;
} CERTStatusConfig;
typedef struct CERTSubjectListStr {
    PLArenaPool *arena;
    int ncerts;
    char *emailAddr;
    CERTSubjectNode *head;
    CERTSubjectNode *tail;
    void *entry;
} CERTSubjectList;
typedef struct CERTSubjectNodeStr {
    struct CERTSubjectNodeStr *next;
    struct CERTSubjectNodeStr *prev;
```

```
    SECItem certKey;
    SECItem keyID;
} CERTSubjectNode;
typedef struct CERTCertificateRequestStr {
    PLArenaPool *arena;
    SECItem version;
    CERTName subject;
    CERTSubjectPublicKeyInfo subjectPublicKeyInfo;
    CERTAttribute **attributes;
} CERTCertificateRequest;
typedef struct CERTCertificateStr {
    PLArenaPool *arena;
    char *subjectName;
    char *issuerName;
    CERTSignedData signatureWrap;
    SECItem derCert;
    SECItem derIssuer;
    SECItem derSubject;
    SECItem derPublicKey;
    SECItem certKey;
    SECItem version;
    SECItem serialNumber;
    SECAlgorithmID signature;
    CERTName issuer;
    CERTValidity validity;
    CERTName subject;
    CERTSubjectPublicKeyInfo subjectPublicKeyInfo;
    SECItem issuerID;
    SECItem subjectID;
    CERTCertExtension **extensions;
    char *emailAddr;
    CERTCertDBHandle *dbhandle;
    SECItem subjectKeyID;
    PRBool keyIDGenerated;
    unsigned int keyUsage;
    unsigned int rawKeyUsage;
    PRBool keyUsagePresent;
    PRUint32 nsCertType;
    PRBool keepSession;
    PRBool timeOK;
    CERTOKDomainName *domainOK;
    PRBool isperm;
    PRBool istemp;
    char *nickname;
    char *dbnickname;
    struct NSSCertificateStr *nssCertificate;
    CERTCertTrust *trust;
    int referenceCount;
    CERTSubjectList *subjectList;
    CERTAuthKeyID *authKeyID;
    PRBool isRoot;
    union {
        void *apointer;
        struct {
            unsigned int hasUnsupportedCriticalExt;
        } bits;
    } options;
    int series;
    PK11SlotInfo *slot;
    CK_OBJECT_HANDLE pkcs11ID;
    PRBool ownSlot;
} CERTCertificate;
typedef struct CERTVerifyLogStr {
    PLArenaPool *arena;
    unsigned int count;
    struct CERTVerifyLogNodeStr *head;
```

```
    struct CERTVerifyLogNodeStr *tail;
} CERTVerifyLog;
typedef struct CRLDistributionPointStr {
    DistributionPointTypes distPointType;
    union {
        CERTGeneralName *fullName;
        CERTRDN relativeName;
    } distPoint;
    SECItem reasons;
    CERTGeneralName *crlIssuer;
    SECItem derDistPoint;
    SECItem derRelativeName;
    SECItem **derCrlIssuer;
    SECItem **derFullName;
    SECItem bitsmap;
} CRLDistributionPoint;
typedef enum SECCertUsageEnum {
    certUsageSSLClient,
    certUsageSSLServer = 1,
    certUsageSSLServerWithStepUp = 2,
    certUsageSSLCA = 3,
    certUsageEmailSigner = 4,
    certUsageEmailRecipient = 5,
    certUsageObjectSigner = 6,
    certUsageUserCertImport = 7,
    certUsageVerifyCA = 8,
    certUsageProtectedObjectSigner = 9,
    certUsageStatusResponder = 10,
    certUsageAnyCA = 11
} SECCertUsage;
typedef PRInt64 SECCertificateUsage;
typedef enum SECCertTimeValidityEnum {
    secCertTimeValid,
    secCertTimeExpired = 1,
    secCertTimeNotValidYet = 2,
    secCertTimeUndetermined = 3
} SECCertTimeValidity;
typedef enum CERTCompareValidityStatusEnum {
    certValidityUndetermined,
    certValidityChooseB = 1,
    certValidityEqual = 2,
    certValidityChooseA = 3
} CERTCompareValidityStatus;
typedef enum CERTGeneralNameTypeEnum {
    certOtherName = 1,
    certRFC822Name = 2,
    certDNSName = 3,
    certX400Address = 4,
    certDirectoryName = 5,
    certEDIPartyName = 6,
    certURI = 7,
    certIPAddress = 8,
    certRegisterID = 9
} CERTGeneralNameType;
typedef struct CERTNameConstraintStr {
    CERTGeneralName name;
    SECItem DERName;
    SECItem min;
    SECItem max;
    PRCList l;
} CERTNameConstraint;
typedef enum DistributionPointTypesEnum {
    generalName = 1,
    relativeDistinguishedName = 2
} DistributionPointTypes;
struct CERTVerifyLogNodeStr {
```

```
    CERTCertificate *cert;
    long int error;
    unsigned int depth;
    void *arg;
    struct CERTVerifyLogNodeStr *next;
    struct CERTVerifyLogNodeStr *prev;
};
typedef SECStatus(*CERTStatusChecker) (CERTCertDBHandle *,
                                                    CERTCertificate *,
PRInt64, void *);
typedef SECStatus(*CERTStatusDestroy) (CERTStatusConfig *);
typedef struct {
    SECOidTag oid;
    SECItem qualifierID;
    SECItem qualifierValue;
} CERTPolicyQualifier;
typedef struct {
    SECOidTag oid;
    SECItem policyID;
    CERTPolicyQualifier **policyQualifiers;
} CERTPolicyInfo;
typedef struct {
    PLArenaPool *arena;
    CERTPolicyInfo **policyInfos;
} CERTCertificatePolicies;
typedef struct {
    SECItem organization;
    SECItem **noticeNumbers;
} CERTNoticeReference;
typedef struct {
    PLArenaPool *arena;
    CERTNoticeReference noticeReference;
    SECItem derNoticeReference;
    SECItem displayText;
} CERTUserNotice;
typedef struct {
    PLArenaPool *arena;
    SECItem **oids;
} CERTOidSequence;
```

7.4.4 nss3/cmsreclist.h

```
#define _CMSRECLIST_H

typedef struct NSSCMSRecipientStr {
    int riIndex;
    int subIndex;
    enum {
        RLIssuerSN,
        RLSubjKeyID = 1
    } kind;
    union {
        CERTIssuerAndSN *issuerAndSN;
        SECItem *subjectKeyID;
    } id;
    CERTCertificate *cert;
    SECKEYPrivateKey *privkey;
    PK11SlotInfo *slot;
} NSSCMSRecipient;
```

7.4.5 nss3/cryptoht.h

```
#define _CRYPTOHT_H_
```

```
typedef struct SGNContextStr SGNContext;
typedef struct VFYContextStr VFYContext;
```

7.4.6 nss3/hasht.h

```
#define _HASHT_H_
#define MD2_LENGTH      16
#define MD5_LENGTH      16
#define SHA1_LENGTH     20
#define SHA256_LENGTH   32
#define SHA384_LENGTH   48
#define SHA512_LENGTH   64
#define HASH_LENGTH_MAX SHA512_LENGTH

typedef struct SECHashObjectStr {
    unsigned int length;
    void *(*create) (void);
    void *(*clone) (void *);
    void (*destroy) (void *, PRBool);
    void (*begin) (void *);
    void (*update) (void *, const unsigned char *, unsigned int);
       void (*end) (void *, unsigned char *, unsigned int *,
unsigned int);
    unsigned int blocklength;
    HASH_HashType type;
} SECHashObject;
typedef struct HASHContextStr {
    const struct SECHashObjectStr *hashobj;
    void *hash_context;
} HASHContext;
typedef enum {
    HASH_AlgNULL,
    HASH_AlgMD2 = 1,
    HASH_AlgMD5 = 2,
    HASH_AlgSHA1 = 3,
    HASH_AlgSHA256 = 4,
    HASH_AlgSHA384 = 5,
    HASH_AlgSHA512 = 6,
    HASH_AlgTOTAL = 7
} HASH_HashType;
```

7.4.7 nss3/key.h

```
#define _KEY_H_
```

7.4.8 nss3/keyhi.h

```
#define _KEYHI_H_

extern void SECKEY_DestroyPrivateKey(SECKEYPrivateKey * key);
```

7.4.9 nss3/keyt.h

```
#define _KEYT_H_
```

7.4.10 nss3/keythi.h

```
#define _KEYTHI_H_

typedef enum {
    nullKey,
    rsaKey = 1,
    dsaKey = 2,
    fortezzaKey = 3,
    dhKey = 4,
    keaKey = 5,
    ecKey = 6
} KeyType;
typedef struct SECKEYRSAPublicKeyStr {
    PLArenaPool *arena;
    SECItem modulus;
    SECItem publicExponent;
} SECKEYRSAPublicKey;
typedef struct SECKEYPQGParamsStr {
    PLArenaPool *arena;
    SECItem prime;
    SECItem subPrime;
    SECItem base;
} SECKEYPQGParams;
typedef struct SECKEYDSAPublicKeyStr {
    SECKEYPQGParams params;
    SECItem publicValue;
} SECKEYDSAPublicKey;
typedef struct SECKEYDHParamsStr {
    PLArenaPool *arena;
    SECItem prime;
    SECItem base;
} SECKEYDHParams;
typedef struct SECKEYDHPublicKeyStr {
    PLArenaPool *arena;
    SECItem prime;
    SECItem base;
    SECItem publicValue;
} SECKEYDHPublicKey;
typedef SECItem SECKEYECParams;
typedef struct SECKEYECPublicKeyStr {
    SECKEYECParams DEREncodedParams;
    int size;
    SECItem publicValue;
} SECKEYECPublicKey;
typedef struct SECKEYFortezzaPublicKeyStr {
    int KEAversion;
    int DSSversion;
    unsigned char KMID[8];
    SECItem clearance;
    SECItem KEApriviledge;
    SECItem DSSpriviledge;
    SECItem KEAKey;
    SECItem DSSKey;
    SECKEYPQGParams params;
    SECKEYPQGParams keaParams;
} SECKEYFortezzaPublicKey;
typedef struct SECKEYKEAParamsStr {
    PLArenaPool *arena;
    SECItem hash;
} SECKEYKEAParams;
typedef struct SECKEYKEAPublicKeyStr {
    SECKEYKEAParams params;
    SECItem publicValue;
} SECKEYKEAPublicKey;
typedef struct SECKEYPublicKeyStr {
    PLArenaPool *arena;
    KeyType keyType;
```



```
    PK11SlotInfo *pkcs11Slot;
    CK_OBJECT_HANDLE pkcs11ID;
    union {
        SECKEYRSAPublicKey rsa;
        SECKEYDSAPublicKey dsa;
        SECKEYDHPublicKey dh;
        SECKEYKEAPublicKey kea;
        SECKEYFortezzaPublicKey fortezza;
        SECKEYECPublicKey ec;
    } u;
} SECKEYPublicKey;
typedef struct SECKEYPrivateKeyStr {
    PLArenaPool *arena;
    KeyType keyType;
    PK11SlotInfo *pkcs11Slot;
    CK_OBJECT_HANDLE pkcs11ID;
    PRBool pkcs11IsTemp;
    void *wincx;
    PRUint32 staticflags;
} SECKEYPrivateKey;
typedef struct {
    PRCList links;
    SECKEYPrivateKey *key;
} SECKEYPrivateKeyListNode;
typedef struct {
    PRCList list;
    PLArenaPool *arena;
} SECKEYPrivateKeyList;
typedef struct {
    PRCList list;
    PLArenaPool *arena;
} SECKEYPublicKeyList;
```

7.4.11 nss3/nss.h

```
#define __nss_h_
#define NSS_VERSION     "3.11.4"
#define NSS_INIT_READONLY       0x1
#define NSS_INIT_NOROOTINIT     0x10
#define NSS_INIT_NOPK11FINALIZE 0x100
#define NSS_INIT_NOCERTDB       0x2
#define NSS_INIT_OPTIMIZESPACE  0x20
#define NSS_INIT_RESERVED       0x200
#define NSS_INIT_NOMODDB        0x4
#define NSS_INIT_PK11THREADSAFE 0x40
#define NSS_INIT_FORCEOPEN      0x8
#define NSS_INIT_PK11RELOAD     0x80
#define NSS_VMINOR      11
#define NSS_VMAJOR      3
#define NSS_VPATCH      4
#define  NSS_INIT_COOPERATE             NSS_INIT_PK11THREADSAFE  |
NSS_INIT_PK11RELOAD | NSS_INIT_NOPK11FINALIZE | NSS_INIT_RESERVED
#define SECMOD_DB       "secmod.db"

extern SECStatus NSS_Init(const char *configdir);
extern SECStatus NSS_InitReadWrite(const char *configdir);
extern SECStatus NSS_NoDB_Init(const char *configdir);
extern SECStatus NSS_Shutdown(void);
```

7.4.12 nss3/nssb64.h

```
#define _NSSB64_H_
```

7.4.13 nss3/nssb64t.h

```
#define _NSSB64T_H_

typedef struct NSSBase64DecoderStr NSSBase64Decoder;
typedef struct NSSBase64EncoderStr NSSBase64Encoder;
```

7.4.14 nss3/nssilckt.h

```
#define _NSSILCKT_H_

typedef enum {
    nssILockArena,
    nssILockSession = 1,
    nssILockObject = 2,
    nssILockRefLock = 3,
    nssILockCert = 4,
    nssILockCertDB = 5,
    nssILockDBM = 6,
    nssILockCache = 7,
    nssILockSSL = 8,
    nssILockList = 9,
    nssILockSlot = 10,
    nssILockFreelist = 11,
    nssILockOID = 12,
    nssILockAttribute = 13,
    nssILockPK11cxt = 14,
    nssILockRWLock = 15,
    nssILockOther = 16,
    nssILockSelfServ = 17,
    nssILockKeyDB = 18,
    nssILockLast = 19
} nssILockType;
```

7.4.15 nss3/nssrwlkt.h

```
#define nssrwlkt_h___

typedef struct nssRWLockStr NSSRWLock;
```

7.4.16 nss3/ocspt.h

```
#define _OCSPT_H_

typedef struct CERTOCSPRequestStr CERTOCSPRequest;
typedef struct CERTOCSPResponseStr CERTOCSPResponse;
typedef struct CERTOCSPCertIDStr CERTOCSPCertID;
typedef struct CERTOCSPSingleResponseStr CERTOCSPSingleResponse;
```

7.4.17 nss3/pk11pub.h

```
#define _PK11PUB_H_

extern   CERTCertificate   *PK11_FindCertFromNickname(const   char
*nickname,
                                                      void *wincx);
extern  SECKEYPrivateKey  *PK11_FindKeyByAnyCert(CERTCertificate  *
cert,
```



```
                                                    void *wincx);
extern char *PK11_GetSlotName(PK11SlotInfo * slot);
extern char *PK11_GetTokenName(PK11SlotInfo * slot);
extern PRBool PK11_IsHW(PK11SlotInfo * slot);
extern PRBool PK11_IsPresent(PK11SlotInfo * slot);
extern PRBool PK11_IsReadOnly(PK11SlotInfo * slot);
extern void PK11_SetPasswordFunc(PK11PasswordFunc func);
```

7.4.18 nss3/pkcs11t.h

```
#define _PKCS11T_H_

typedef unsigned char CK_BYTE;
typedef CK_BYTE CK_CHAR;
typedef CK_BYTE CK_UTF8CHAR;
typedef unsigned long int CK_ULONG;
typedef CK_ULONG CK_FLAGS;
typedef void *CK_VOID_PTR;
typedef struct CK_VERSION {
    CK_BYTE major;
    CK_BYTE minor;
} CK_VERSION;
typedef struct CK_INFO {
    CK_VERSION cryptokiVersion;
    CK_UTF8CHAR manufacturerID[31];
    CK_FLAGS flags;
    CK_UTF8CHAR libraryDescription[31];
    CK_VERSION libraryVersion;
} CK_INFO;
typedef CK_ULONG CK_SLOT_ID;
typedef struct CK_SLOT_INFO {
    CK_UTF8CHAR slotDescription[63];
    CK_UTF8CHAR manufacturerID[31];
    CK_FLAGS flags;
    CK_VERSION hardwareVersion;
    CK_VERSION firmwareVersion;
} CK_SLOT_INFO;
typedef struct CK_TOKEN_INFO {
    CK_UTF8CHAR label[31];
    CK_UTF8CHAR manufacturerID[31];
    CK_UTF8CHAR model[15];
    CK_CHAR serialNumber[15];
    CK_FLAGS flags;
    CK_ULONG ulMaxSessionCount;
    CK_ULONG ulSessionCount;
    CK_ULONG ulMaxRwSessionCount;
    CK_ULONG ulRwSessionCount;
    CK_ULONG ulMaxPinLen;
    CK_ULONG ulMinPinLen;
    CK_ULONG ulTotalPublicMemory;
    CK_ULONG ulFreePublicMemory;
    CK_ULONG ulTotalPrivateMemory;
    CK_ULONG ulFreePrivateMemory;
    CK_VERSION hardwareVersion;
    CK_VERSION firmwareVersion;
    CK_CHAR utcTime[15];
} CK_TOKEN_INFO;
typedef CK_ULONG CK_SESSION_HANDLE;
typedef CK_ULONG CK_OBJECT_HANDLE;
typedef CK_ULONG CK_OBJECT_CLASS;
typedef CK_ULONG CK_KEY_TYPE;
typedef CK_ULONG CK_ATTRIBUTE_TYPE;
typedef struct CK_ATTRIBUTE {
    CK_ATTRIBUTE_TYPE type;
```

```
    CK_VOID_PTR pValue;
    CK_ULONG ulValueLen;
} CK_ATTRIBUTE;
typedef CK_ATTRIBUTE *CK_ATTRIBUTE_PTR;
typedef CK_ULONG CK_MECHANISM_TYPE;
typedef struct CK_MECHANISM {
    CK_MECHANISM_TYPE mechanism;
    CK_VOID_PTR pParameter;
    CK_ULONG ulParameterLen;
} CK_MECHANISM;
typedef CK_MECHANISM *CK_MECHANISM_PTR;
typedef CK_ULONG CK_RV;
```

7.4.19 nss3/pkcs7t.h

```
#define _PKCS7T_H_

typedef struct SEC_PKCS7RecipientInfoStr {
    SECItem version;
    CERTIssuerAndSN *issuerAndSN;
    SECAlgorithmID keyEncAlg;
    SECItem encKey;
    CERTCertificate *cert;
} SEC_PKCS7RecipientInfo;
```

7.4.20 nss3/secasn1t.h

```
#define _SECASN1T_H_

typedef struct sec_ASN1Template_struct {
    unsigned long int kind;
    unsigned long int offset;
    const void *sub;
    unsigned int size;
} SEC_ASN1Template;
typedef struct sec_DecoderContext_struct SEC_ASN1DecoderContext;
typedef struct sec_EncoderContext_struct SEC_ASN1EncoderContext;
typedef enum {
    SEC_ASN1_Identifier,
    SEC_ASN1_Length = 1,
    SEC_ASN1_Contents = 2,
    SEC_ASN1_EndOfContents = 3
} SEC_ASN1EncodingPart;
typedef void (*SEC_ASN1NotifyProc) (void *, PRBool, void *, int);
typedef void (*SEC_ASN1WriteProc) (void *, const char *, unsigned
long int,
                                   int, SEC_ASN1EncodingPart);
```

7.4.21 nss3/seccomon.h

```
#define _SECCOMMON_H_

typedef enum {
    siBuffer,
    siClearDataBuffer = 1,
    siCipherDataBuffer = 2,
    siDERCertBuffer = 3,
    siEncodedCertBuffer = 4,
    siDERNameBuffer = 5,
    siEncodedNameBuffer = 6,
    siAsciiNameString = 7,
```

```
    siAsciiString = 8,
    siDEROID = 9,
    siUnsignedInteger = 10,
    siUTCTime = 11,
    siGeneralizedTime = 12,
    siVisibleString = 13,
    siUTF8String = 14,
    siBMPString = 15
} SECItemType;
typedef struct SECItemStr {
    SECItemType type;
    unsigned char *data;
    unsigned int len;
} SECItem;
typedef enum _SECStatus {
    SECWouldBlock = -2,
    SECFailure = -1,
    SECSuccess
} SECStatus;
typedef enum _SECComparison {
    SECLessThan = -1,
    SECEqual,
    SECGreaterThan = 1
} SECComparison;
```

7.4.22 nss3/secdert.h

```
#define _SECDERT_H_

typedef struct DERTemplateStr {
    unsigned long int kind;
    unsigned int offset;
    DERTemplate *sub;
    unsigned long int arg;
} DERTemplate;
```

7.4.23 nss3/secdigt.h

```
#define _SECDIGT_H_

typedef struct SGNDigestInfoStr {
    PLArenaPool *arena;
    SECAlgorithmID digestAlgorithm;
    SECItem digest;
} SGNDigestInfo;
```

7.4.24 nss3/secmodt.h

```
#define _SECMODT_H_
#define SECMOD_MAKE_NSS_FLAGS(fips,slot)         \
        "Flags=internal,critical"fips" \
        slotparams=("#slot"={"SECMOD_SLOT_FLAGS"})"
#define  SECMOD_FIPS_NAME            "NSS  Internal  FIPS  PKCS  #11
Module"
#define SECMOD_INT_NAME "NSS Internal PKCS #11 Module"
#define                                          SECMOD_SLOT_FLAGS
"slotFlags=[RSA,DSA,DH,RC2,RC4,DES,RANDOM,SHA1,MD5,MD2,SSL,TLS,AE
S,SHA256,SHA512]"
#define SECMOD_EXTERNAL 0
#define CRL_IMPORT_DEFAULT_OPTIONS      0x00000000
#define CRL_IMPORT_BYPASS_CHECKS        0x00000001
```

```
#define PK11_ATTR_TOKEN 0x00000001L
#define SECMOD_RSA_FLAG 0x00000001L
#define PK11_ATTR_SESSION       0x00000002L
#define SECMOD_DSA_FLAG 0x00000002L
#define PK11_ATTR_PRIVATE       0x00000004L
#define SECMOD_RC2_FLAG 0x00000004L
#define PK11_ATTR_PUBLIC        0x00000008L
#define SECMOD_RC4_FLAG 0x00000008L
#define PK11_ATTR_MODIFIABLE    0x00000010L
#define SECMOD_DES_FLAG 0x00000010L
#define PK11_ATTR_UNMODIFIABLE  0x00000020L
#define SECMOD_DH_FLAG  0x00000020L
#define PK11_ATTR_SENSITIVE     0x00000040L
#define SECMOD_FORTEZZA_FLAG    0x00000040L
#define PK11_ATTR_INSENSITIVE   0x00000080L
#define SECMOD_RC5_FLAG 0x00000080L
#define PK11_ATTR_EXTRACTABLE   0x00000100L
#define SECMOD_SHA1_FLAG        0x00000100L
#define PK11_ATTR_UNEXTRACTABLE 0x00000200L
#define SECMOD_MD5_FLAG 0x00000200L
#define SECMOD_MD2_FLAG 0x00000400L
#define SECMOD_SSL_FLAG 0x00000800L
#define SECMOD_TLS_FLAG 0x00001000L
#define SECMOD_AES_FLAG 0x00002000L
#define SECMOD_SHA256_FLAG      0x00004000L
#define SECMOD_SHA512_FLAG      0x00008000L
#define SECMOD_END_WAIT 0x01
#define SECMOD_WAIT_SIMULATED_EVENT     0x02
#define SECMOD_WAIT_PKCS11_EVENT        0x04
#define SECMOD_RESERVED_FLAG    0X08000000L
#define SECMOD_FRIENDLY_FLAG    0x10000000L
#define PK11_OWN_PW_DEFAULTS    0x20000000L
#define PK11_DISABLE_FLAG       0x40000000L
#define SECMOD_RANDOM_FLAG      0x80000000L
#define CKM_FAKE_RANDOM 0x80000efeL
#define CKM_INVALID_MECHANISM   0xffffffffL
#define SECMOD_INTERNAL 1
#define SECMOD_FIPS     2
#define PK11_PW_AUTHENTICATED   "AUTH"
#define PK11_PW_RETRY   "RETRY"
#define SECMOD_INT_FLAGS        SECMOD_MAKE_NSS_FLAGS("",1)
#define SECMOD_FIPS_FLAGS       SECMOD_MAKE_NSS_FLAGS(",fips",3)
#define PK11_PW_TRY     "TRY"

typedef struct SECMODModuleStr {
    PLArenaPool *arena;
    PRBool internal;
    PRBool loaded;
    PRBool isFIPS;
    char *dllName;
    char *commonName;
    void *library;
    void *functionList;
    PRLock *refLock;
    int refCount;
    PK11SlotInfo **slots;
    int slotCount;
    PK11PreSlotInfo *slotInfo;
    int slotInfoCount;
    SECMODModuleID moduleID;
    PRBool isThreadSafe;
    unsigned long int ssl[1];
    char *libraryParams;
    void *moduleDBFunc;
    SECMODModule *parent;
    PRBool isCritical;
```



```
    PRBool isModuleDB;
    PRBool moduleDBOnly;
    int trustOrder;
    int cipherOrder;
    unsigned long int evControlMask;
    CK_VERSION cryptokiVersion;
} SECMODModule;
typedef struct SECMODModuleListStr {
    SECMODModuleList *next;
    SECMODModule *module;
} SECMODModuleList;
typedef NSSRWLock SECMODListLock;
typedef struct PK11SlotInfoStr PK11SlotInfo;
typedef struct PK11PreSlotInfoStr PK11PreSlotInfo;
typedef struct PK11SymKeyStr PK11SymKey;
typedef struct PK11ContextStr PK11Context;
typedef struct PK11SlotListStr PK11SlotList;
typedef struct PK11SlotListElementStr PK11SlotListElement;
typedef unsigned long int SECMODModuleID;
typedef struct PK11DefaultArrayEntryStr PK11DefaultArrayEntry;
typedef struct PK11GenericObjectStr PK11GenericObject;
typedef void (*PK11FreeDataFunc) (void *);
typedef enum {
    PK11CertListUnique,
    PK11CertListUser = 1,
    PK11CertListRootUnique = 2,
    PK11CertListCA = 3,
    PK11CertListCAUnique = 4,
    PK11CertListUserUnique = 5,
    PK11CertListAll = 6
} PK11CertListType;
typedef PRUint32 PK11AttrFlags;
typedef enum {
    PK11_OriginNULL,
    PK11_OriginDerive = 1,
    PK11_OriginGenerated = 2,
    PK11_OriginFortezzaHack = 3,
    PK11_OriginUnwrap = 4
} PK11Origin;
typedef enum {
    PK11_DIS_NONE,
    PK11_DIS_USER_SELECTED = 1,
    PK11_DIS_COULD_NOT_INIT_TOKEN = 2,
    PK11_DIS_TOKEN_VERIFY_FAILED = 3,
    PK11_DIS_TOKEN_NOT_PRESENT = 4
} PK11DisableReasons;
typedef enum {
    PK11_TypeGeneric,
    PK11_TypePrivKey = 1,
    PK11_TypePubKey = 2,
    PK11_TypeCert = 3,
    PK11_TypeSymKey = 4
} PK11ObjectType;
typedef char *(*PK11PasswordFunc) (PK11SlotInfo *, PRBool, void
*);
typedef struct SECKEYAttributeStr {
    SECItem attrType;
    SECItem **attrValue;
} SECKEYAttribute;
typedef struct SECKEYPrivateKeyInfoStr {
    PLArenaPool *arena;
    SECItem version;
    SECAlgorithmID algorithm;
    SECItem privateKey;
    SECKEYAttribute **attributes;
} SECKEYPrivateKeyInfo;
```

```
typedef struct SECKEYEncryptedPrivateKeyInfoStr {
    PLArenaPool *arena;
    SECAlgorithmID algorithm;
    SECItem encryptedData;
} SECKEYEncryptedPrivateKeyInfo;
typedef enum {
    PK11TokenNotRemovable,
    PK11TokenPresent = 1,
    PK11TokenChanged = 2,
    PK11TokenRemoved = 3
} PK11TokenStatus;
typedef enum {
    PK11TokenRemovedOrChangedEvent,
    PK11TokenPresentEvent = 1
} PK11TokenEvent;
```

7.4.25 nss3/secoidt.h

```
#define _SECOIDT_H_

typedef struct SECOidDataStr {
    SECItem oid;
    SECOidTag offset;
    const char *desc;
    unsigned long int mechanism;
    SECSupportExtenTag supportedExtension;
} SECOidData;
typedef struct SECAlgorithmIDStr {
    SECItem algorithm;
    SECItem parameters;
} SECAlgorithmID;
typedef enum {
    SEC_OID_UNKNOWN,
    SEC_OID_MD2 = 1,
    SEC_OID_MD4 = 2,
    SEC_OID_MD5 = 3,
    SEC_OID_SHA1 = 4,
    SEC_OID_RC2_CBC = 5,
    SEC_OID_RC4 = 6,
    SEC_OID_DES_EDE3_CBC = 7,
    SEC_OID_RC5_CBC_PAD = 8,
    SEC_OID_DES_ECB = 9,
    SEC_OID_DES_CBC = 10,
    SEC_OID_DES_OFB = 11,
    SEC_OID_DES_CFB = 12,
    SEC_OID_DES_MAC = 13,
    SEC_OID_DES_EDE = 14,
    SEC_OID_ISO_SHA_WITH_RSA_SIGNATURE = 15,
    SEC_OID_PKCS1_RSA_ENCRYPTION = 16,
    SEC_OID_PKCS1_MD2_WITH_RSA_ENCRYPTION = 17,
    SEC_OID_PKCS1_MD4_WITH_RSA_ENCRYPTION = 18,
    SEC_OID_PKCS1_MD5_WITH_RSA_ENCRYPTION = 19,
    SEC_OID_PKCS1_SHA1_WITH_RSA_ENCRYPTION = 20,
    SEC_OID_PKCS5_PBE_WITH_MD2_AND_DES_CBC = 21,
    SEC_OID_PKCS5_PBE_WITH_MD5_AND_DES_CBC = 22,
    SEC_OID_PKCS5_PBE_WITH_SHA1_AND_DES_CBC = 23,
    SEC_OID_PKCS7 = 24,
    SEC_OID_PKCS7_DATA = 25,
    SEC_OID_PKCS7_SIGNED_DATA = 26,
    SEC_OID_PKCS7_ENVELOPED_DATA = 27,
    SEC_OID_PKCS7_SIGNED_ENVELOPED_DATA = 28,
    SEC_OID_PKCS7_DIGESTED_DATA = 29,
    SEC_OID_PKCS7_ENCRYPTED_DATA = 30,
    SEC_OID_PKCS9_EMAIL_ADDRESS = 31,
```



```
SEC_OID_PKCS9_UNSTRUCTURED_NAME = 32,
SEC_OID_PKCS9_CONTENT_TYPE = 33,
SEC_OID_PKCS9_MESSAGE_DIGEST = 34,
SEC_OID_PKCS9_SIGNING_TIME = 35,
SEC_OID_PKCS9_COUNTER_SIGNATURE = 36,
SEC_OID_PKCS9_CHALLENGE_PASSWORD = 37,
SEC_OID_PKCS9_UNSTRUCTURED_ADDRESS = 38,
SEC_OID_PKCS9_EXTENDED_CERTIFICATE_ATTRIBUTES = 39,
SEC_OID_PKCS9_SMIME_CAPABILITIES = 40,
SEC_OID_AVA_COMMON_NAME = 41,
SEC_OID_AVA_COUNTRY_NAME = 42,
SEC_OID_AVA_LOCALITY = 43,
SEC_OID_AVA_STATE_OR_PROVINCE = 44,
SEC_OID_AVA_ORGANIZATION_NAME = 45,
SEC_OID_AVA_ORGANIZATIONAL_UNIT_NAME = 46,
SEC_OID_AVA_DN_QUALIFIER = 47,
SEC_OID_AVA_DC = 48,
SEC_OID_NS_TYPE_GIF = 49,
SEC_OID_NS_TYPE_JPEG = 50,
SEC_OID_NS_TYPE_URL = 51,
SEC_OID_NS_TYPE_HTML = 52,
SEC_OID_NS_TYPE_CERT_SEQUENCE = 53,
SEC_OID_MISSI_KEA_DSS_OLD = 54,
SEC_OID_MISSI_DSS_OLD = 55,
SEC_OID_MISSI_KEA_DSS = 56,
SEC_OID_MISSI_DSS = 57,
SEC_OID_MISSI_KEA = 58,
SEC_OID_MISSI_ALT_KEA = 59,
SEC_OID_NS_CERT_EXT_NETSCAPE_OK = 60,
SEC_OID_NS_CERT_EXT_ISSUER_LOGO = 61,
SEC_OID_NS_CERT_EXT_SUBJECT_LOGO = 62,
SEC_OID_NS_CERT_EXT_CERT_TYPE = 63,
SEC_OID_NS_CERT_EXT_BASE_URL = 64,
SEC_OID_NS_CERT_EXT_REVOCATION_URL = 65,
SEC_OID_NS_CERT_EXT_CA_REVOCATION_URL = 66,
SEC_OID_NS_CERT_EXT_CA_CRL_URL = 67,
SEC_OID_NS_CERT_EXT_CA_CERT_URL = 68,
SEC_OID_NS_CERT_EXT_CERT_RENEWAL_URL = 69,
SEC_OID_NS_CERT_EXT_CA_POLICY_URL = 70,
SEC_OID_NS_CERT_EXT_HOMEPAGE_URL = 71,
SEC_OID_NS_CERT_EXT_ENTITY_LOGO = 72,
SEC_OID_NS_CERT_EXT_USER_PICTURE = 73,
SEC_OID_NS_CERT_EXT_SSL_SERVER_NAME = 74,
SEC_OID_NS_CERT_EXT_COMMENT = 75,
SEC_OID_NS_CERT_EXT_LOST_PASSWORD_URL = 76,
SEC_OID_NS_CERT_EXT_CERT_RENEWAL_TIME = 77,
SEC_OID_NS_KEY_USAGE_GOVT_APPROVED = 78,
SEC_OID_X509_SUBJECT_DIRECTORY_ATTR = 79,
SEC_OID_X509_SUBJECT_KEY_ID = 80,
SEC_OID_X509_KEY_USAGE = 81,
SEC_OID_X509_PRIVATE_KEY_USAGE_PERIOD = 82,
SEC_OID_X509_SUBJECT_ALT_NAME = 83,
SEC_OID_X509_ISSUER_ALT_NAME = 84,
SEC_OID_X509_BASIC_CONSTRAINTS = 85,
SEC_OID_X509_NAME_CONSTRAINTS = 86,
SEC_OID_X509_CRL_DIST_POINTS = 87,
SEC_OID_X509_CERTIFICATE_POLICIES = 88,
SEC_OID_X509_POLICY_MAPPINGS = 89,
SEC_OID_X509_POLICY_CONSTRAINTS = 90,
SEC_OID_X509_AUTH_KEY_ID = 91,
SEC_OID_X509_EXT_KEY_USAGE = 92,
SEC_OID_X509_AUTH_INFO_ACCESS = 93,
SEC_OID_X509_CRL_NUMBER = 94,
SEC_OID_X509_REASON_CODE = 95,
SEC_OID_X509_INVALID_DATE = 96,
SEC_OID_X500_RSA_ENCRYPTION = 97,
```

```
        SEC_OID_RFC1274_UID = 98,
        SEC_OID_RFC1274_MAIL = 99,
        SEC_OID_PKCS12 = 100,
        SEC_OID_PKCS12_MODE_IDS = 101,
        SEC_OID_PKCS12_ESPVK_IDS = 102,
        SEC_OID_PKCS12_BAG_IDS = 103,
        SEC_OID_PKCS12_CERT_BAG_IDS = 104,
        SEC_OID_PKCS12_OIDS = 105,
        SEC_OID_PKCS12_PBE_IDS = 106,
        SEC_OID_PKCS12_SIGNATURE_IDS = 107,
        SEC_OID_PKCS12_ENVELOPING_IDS = 108,
        SEC_OID_PKCS12_PKCS8_KEY_SHROUDING = 109,
        SEC_OID_PKCS12_KEY_BAG_ID = 110,
        SEC_OID_PKCS12_CERT_AND_CRL_BAG_ID = 111,
        SEC_OID_PKCS12_SECRET_BAG_ID = 112,
        SEC_OID_PKCS12_X509_CERT_CRL_BAG = 113,
        SEC_OID_PKCS12_SDSI_CERT_BAG = 114,
        SEC_OID_PKCS12_PBE_WITH_SHA1_AND_128_BIT_RC4 = 115,
        SEC_OID_PKCS12_PBE_WITH_SHA1_AND_40_BIT_RC4 = 116,
        SEC_OID_PKCS12_PBE_WITH_SHA1_AND_TRIPLE_DES_CBC = 117,
        SEC_OID_PKCS12_PBE_WITH_SHA1_AND_128_BIT_RC2_CBC = 118,
        SEC_OID_PKCS12_PBE_WITH_SHA1_AND_40_BIT_RC2_CBC = 119,
        SEC_OID_PKCS12_RSA_ENCRYPTION_WITH_128_BIT_RC4 = 120,
        SEC_OID_PKCS12_RSA_ENCRYPTION_WITH_40_BIT_RC4 = 121,
        SEC_OID_PKCS12_RSA_ENCRYPTION_WITH_TRIPLE_DES = 122,
        SEC_OID_PKCS12_RSA_SIGNATURE_WITH_SHA1_DIGEST = 123,
        SEC_OID_ANSIX9_DSA_SIGNATURE = 124,
        SEC_OID_ANSIX9_DSA_SIGNATURE_WITH_SHA1_DIGEST = 125,
        SEC_OID_BOGUS_DSA_SIGNATURE_WITH_SHA1_DIGEST = 126,
        SEC_OID_VERISIGN_USER_NOTICES = 127,
        SEC_OID_PKIX_CPS_POINTER_QUALIFIER = 128,
        SEC_OID_PKIX_USER_NOTICE_QUALIFIER = 129,
        SEC_OID_PKIX_OCSP = 130,
        SEC_OID_PKIX_OCSP_BASIC_RESPONSE = 131,
        SEC_OID_PKIX_OCSP_NONCE = 132,
        SEC_OID_PKIX_OCSP_CRL = 133,
        SEC_OID_PKIX_OCSP_RESPONSE = 134,
        SEC_OID_PKIX_OCSP_NO_CHECK = 135,
        SEC_OID_PKIX_OCSP_ARCHIVE_CUTOFF = 136,
        SEC_OID_PKIX_OCSP_SERVICE_LOCATOR = 137,
        SEC_OID_PKIX_REGCTRL_REGTOKEN = 138,
        SEC_OID_PKIX_REGCTRL_AUTHENTICATOR = 139,
        SEC_OID_PKIX_REGCTRL_PKIPUBINFO = 140,
        SEC_OID_PKIX_REGCTRL_PKI_ARCH_OPTIONS = 141,
        SEC_OID_PKIX_REGCTRL_OLD_CERT_ID = 142,
        SEC_OID_PKIX_REGCTRL_PROTOCOL_ENC_KEY = 143,
        SEC_OID_PKIX_REGINFO_UTF8_PAIRS = 144,
        SEC_OID_PKIX_REGINFO_CERT_REQUEST = 145,
        SEC_OID_EXT_KEY_USAGE_SERVER_AUTH = 146,
        SEC_OID_EXT_KEY_USAGE_CLIENT_AUTH = 147,
        SEC_OID_EXT_KEY_USAGE_CODE_SIGN = 148,
        SEC_OID_EXT_KEY_USAGE_EMAIL_PROTECT = 149,
        SEC_OID_EXT_KEY_USAGE_TIME_STAMP = 150,
        SEC_OID_OCSP_RESPONDER = 151,
        SEC_OID_NETSCAPE_SMIME_KEA = 152,
        SEC_OID_FORTEZZA_SKIPJACK = 153,
        SEC_OID_PKCS12_V2_PBE_WITH_SHA1_AND_128_BIT_RC4 = 154,
        SEC_OID_PKCS12_V2_PBE_WITH_SHA1_AND_40_BIT_RC4 = 155,
            SEC_OID_PKCS12_V2_PBE_WITH_SHA1_AND_3KEY_TRIPLE_DES_CBC  =
156,
            SEC_OID_PKCS12_V2_PBE_WITH_SHA1_AND_2KEY_TRIPLE_DES_CBC  =
157,
        SEC_OID_PKCS12_V2_PBE_WITH_SHA1_AND_128_BIT_RC2_CBC = 158,
        SEC_OID_PKCS12_V2_PBE_WITH_SHA1_AND_40_BIT_RC2_CBC = 159,
        SEC_OID_PKCS12_SAFE_CONTENTS_ID = 160,
        SEC_OID_PKCS12_PKCS8_SHROUDED_KEY_BAG_ID = 161,
```

```
SEC_OID_PKCS12_V1_KEY_BAG_ID = 162,
SEC_OID_PKCS12_V1_PKCS8_SHROUDED_KEY_BAG_ID = 163,
SEC_OID_PKCS12_V1_CERT_BAG_ID = 164,
SEC_OID_PKCS12_V1_CRL_BAG_ID = 165,
SEC_OID_PKCS12_V1_SECRET_BAG_ID = 166,
SEC_OID_PKCS12_V1_SAFE_CONTENTS_BAG_ID = 167,
SEC_OID_PKCS9_X509_CERT = 168,
SEC_OID_PKCS9_SDSI_CERT = 169,
SEC_OID_PKCS9_X509_CRL = 170,
SEC_OID_PKCS9_FRIENDLY_NAME = 171,
SEC_OID_PKCS9_LOCAL_KEY_ID = 172,
SEC_OID_BOGUS_KEY_USAGE = 173,
SEC_OID_X942_DIFFIE_HELMAN_KEY = 174,
SEC_OID_NETSCAPE_NICKNAME = 175,
SEC_OID_NETSCAPE_RECOVERY_REQUEST = 176,
SEC_OID_CERT_RENEWAL_LOCATOR = 177,
SEC_OID_NS_CERT_EXT_SCOPE_OF_USE = 178,
SEC_OID_CMS_EPHEMERAL_STATIC_DIFFIE_HELLMAN = 179,
SEC_OID_CMS_3DES_KEY_WRAP = 180,
SEC_OID_CMS_RC2_KEY_WRAP = 181,
SEC_OID_SMIME_ENCRYPTION_KEY_PREFERENCE = 182,
SEC_OID_AES_128_ECB = 183,
SEC_OID_AES_128_CBC = 184,
SEC_OID_AES_192_ECB = 185,
SEC_OID_AES_192_CBC = 186,
SEC_OID_AES_256_ECB = 187,
SEC_OID_AES_256_CBC = 188,
SEC_OID_SDN702_DSA_SIGNATURE = 189,
SEC_OID_MS_SMIME_ENCRYPTION_KEY_PREFERENCE = 190,
SEC_OID_SHA256 = 191,
SEC_OID_SHA384 = 192,
SEC_OID_SHA512 = 193,
SEC_OID_PKCS1_SHA256_WITH_RSA_ENCRYPTION = 194,
SEC_OID_PKCS1_SHA384_WITH_RSA_ENCRYPTION = 195,
SEC_OID_PKCS1_SHA512_WITH_RSA_ENCRYPTION = 196,
SEC_OID_AES_128_KEY_WRAP = 197,
SEC_OID_AES_192_KEY_WRAP = 198,
SEC_OID_AES_256_KEY_WRAP = 199,
SEC_OID_ANSIX962_EC_PUBLIC_KEY = 200,
SEC_OID_ANSIX962_ECDSA_SHA1_SIGNATURE = 201,
SEC_OID_ANSIX962_EC_PRIME192V1 = 202,
SEC_OID_ANSIX962_EC_PRIME192V2 = 203,
SEC_OID_ANSIX962_EC_PRIME192V3 = 204,
SEC_OID_ANSIX962_EC_PRIME239V1 = 205,
SEC_OID_ANSIX962_EC_PRIME239V2 = 206,
SEC_OID_ANSIX962_EC_PRIME239V3 = 207,
SEC_OID_ANSIX962_EC_PRIME256V1 = 208,
SEC_OID_SECG_EC_SECP112R1 = 209,
SEC_OID_SECG_EC_SECP112R2 = 210,
SEC_OID_SECG_EC_SECP128R1 = 211,
SEC_OID_SECG_EC_SECP128R2 = 212,
SEC_OID_SECG_EC_SECP160K1 = 213,
SEC_OID_SECG_EC_SECP160R1 = 214,
SEC_OID_SECG_EC_SECP160R2 = 215,
SEC_OID_SECG_EC_SECP192K1 = 216,
SEC_OID_SECG_EC_SECP224K1 = 217,
SEC_OID_SECG_EC_SECP224R1 = 218,
SEC_OID_SECG_EC_SECP256K1 = 219,
SEC_OID_SECG_EC_SECP384R1 = 220,
SEC_OID_SECG_EC_SECP521R1 = 221,
SEC_OID_ANSIX962_EC_C2PNB163V1 = 222,
SEC_OID_ANSIX962_EC_C2PNB163V2 = 223,
SEC_OID_ANSIX962_EC_C2PNB163V3 = 224,
SEC_OID_ANSIX962_EC_C2PNB176V1 = 225,
SEC_OID_ANSIX962_EC_C2TNB191V1 = 226,
SEC_OID_ANSIX962_EC_C2TNB191V2 = 227,
```

```
SEC_OID_ANSIX962_EC_C2TNB191V3 = 228,
SEC_OID_ANSIX962_EC_C2ONB191V4 = 229,
SEC_OID_ANSIX962_EC_C2ONB191V5 = 230,
SEC_OID_ANSIX962_EC_C2PNB208W1 = 231,
SEC_OID_ANSIX962_EC_C2TNB239V1 = 232,
SEC_OID_ANSIX962_EC_C2TNB239V2 = 233,
SEC_OID_ANSIX962_EC_C2TNB239V3 = 234,
SEC_OID_ANSIX962_EC_C2ONB239V4 = 235,
SEC_OID_ANSIX962_EC_C2ONB239V5 = 236,
SEC_OID_ANSIX962_EC_C2PNB272W1 = 237,
SEC_OID_ANSIX962_EC_C2PNB304W1 = 238,
SEC_OID_ANSIX962_EC_C2TNB359V1 = 239,
SEC_OID_ANSIX962_EC_C2PNB368W1 = 240,
SEC_OID_ANSIX962_EC_C2TNB431R1 = 241,
SEC_OID_SECG_EC_SECT113R1 = 242,
SEC_OID_SECG_EC_SECT113R2 = 243,
SEC_OID_SECG_EC_SECT131R1 = 244,
SEC_OID_SECG_EC_SECT131R2 = 245,
SEC_OID_SECG_EC_SECT163K1 = 246,
SEC_OID_SECG_EC_SECT163R1 = 247,
SEC_OID_SECG_EC_SECT163R2 = 248,
SEC_OID_SECG_EC_SECT193R1 = 249,
SEC_OID_SECG_EC_SECT193R2 = 250,
SEC_OID_SECG_EC_SECT233K1 = 251,
SEC_OID_SECG_EC_SECT233R1 = 252,
SEC_OID_SECG_EC_SECT239K1 = 253,
SEC_OID_SECG_EC_SECT283K1 = 254,
SEC_OID_SECG_EC_SECT283R1 = 255,
SEC_OID_SECG_EC_SECT409K1 = 256,
SEC_OID_SECG_EC_SECT409R1 = 257,
SEC_OID_SECG_EC_SECT571K1 = 258,
SEC_OID_SECG_EC_SECT571R1 = 259,
SEC_OID_NETSCAPE_AOLSCREENNAME = 260,
SEC_OID_AVA_SURNAME = 261,
SEC_OID_AVA_SERIAL_NUMBER = 262,
SEC_OID_AVA_STREET_ADDRESS = 263,
SEC_OID_AVA_TITLE = 264,
SEC_OID_AVA_POSTAL_ADDRESS = 265,
SEC_OID_AVA_POSTAL_CODE = 266,
SEC_OID_AVA_POST_OFFICE_BOX = 267,
SEC_OID_AVA_GIVEN_NAME = 268,
SEC_OID_AVA_INITIALS = 269,
SEC_OID_AVA_GENERATION_QUALIFIER = 270,
SEC_OID_AVA_HOUSE_IDENTIFIER = 271,
SEC_OID_AVA_PSEUDONYM = 272,
SEC_OID_PKIX_CA_ISSUERS = 273,
SEC_OID_PKCS9_EXTENSION_REQUEST = 274,
SEC_OID_ANSIX962_ECDSA_SIGNATURE_RECOMMENDED_DIGEST = 275,
SEC_OID_ANSIX962_ECDSA_SIGNATURE_SPECIFIED_DIGEST = 276,
SEC_OID_ANSIX962_ECDSA_SHA224_SIGNATURE = 277,
SEC_OID_ANSIX962_ECDSA_SHA256_SIGNATURE = 278,
SEC_OID_ANSIX962_ECDSA_SHA384_SIGNATURE = 279,
SEC_OID_ANSIX962_ECDSA_SHA512_SIGNATURE = 280,
SEC_OID_X509_HOLD_INSTRUCTION_CODE = 281,
SEC_OID_X509_DELTA_CRL_INDICATOR = 282,
SEC_OID_X509_ISSUING_DISTRIBUTION_POINT = 283,
SEC_OID_X509_CERT_ISSUER = 284,
SEC_OID_X509_FRESHEST_CRL = 285,
SEC_OID_X509_INHIBIT_ANY_POLICY = 286,
SEC_OID_X509_SUBJECT_INFO_ACCESS = 287,
SEC_OID_CAMELLIA_128_CBC = 288,
SEC_OID_CAMELLIA_192_CBC = 289,
SEC_OID_CAMELLIA_256_CBC = 290,
SEC_OID_PKCS5_PBKDF2 = 291,
SEC_OID_PKCS5_PBES2 = 292,
SEC_OID_PKCS5_PBMAC1 = 293,
```

```
    SEC_OID_HMAC_SHA1 = 294,
    SEC_OID_HMAC_SHA224 = 295,
    SEC_OID_HMAC_SHA256 = 296,
    SEC_OID_HMAC_SHA384 = 297,
    SEC_OID_HMAC_SHA512 = 298,
    SEC_OID_PKIX_TIMESTAMPING = 299,
    SEC_OID_PKIX_CA_REPOSITORY = 300,
    SEC_OID_ISO_SHA1_WITH_RSA_SIGNATURE = 301,
    SEC_OID_TOTAL = 302
} SECOidTag;
typedef enum {
    INVALID_CERT_EXTENSION,
    UNSUPPORTED_CERT_EXTENSION = 1,
    SUPPORTED_CERT_EXTENSION = 2
} SECSupportExtenTag;
```

7.4.26 nss3/secpkcs5.h

```
#define _SECPKCS5_H_

typedef enum {
    pbeBitGenIDNull,
    pbeBitGenCipherKey = 1,
    pbeBitGenCipherIV = 2,
    pbeBitGenIntegrityKey = 3
} PBEBitGenID;
typedef struct PBEBitGenContextStr PBEBitGenContext;
```

7.4.27 nss3/secport.h

```
#define _SECPORT_H_

typedef  PRBool(*PORTCharConversionWSwapFunc)  (PRBool,  unsigned
char *,
                                                unsigned int,
                                                unsigned char *,
                                                      unsigned int,
unsigned int *,
                                                PRBool);
typedef PRBool(*PORTCharConversionFunc) (PRBool, unsigned char *,
                                            unsigned int, unsigned
char *,
                                            unsigned int, unsigned
int *);
```

7.5 Interfaces for libssl3

Table 7-5 defines the library name and shared object name for the libssl3 library

Table 7-5 libssl3 Definition

Library:	libssl3
SONAME:	libssl3.so

The behavior of the interfaces in this library is specified by the following specifications:

[NSS SSL] Mozilla's NSS SSL Reference

7.5.1 NSS SSL

7.5.1.1 Interfaces for NSS SSL

An LSB conforming implementation shall provide the generic functions for NSS SSL specified in Table 7-6, with the full mandatory functionality as described in the referenced underlying specification.

Table 7-6 libssl3 - NSS SSL Function Interfaces

NSS_CmpCertChainWCANames(NSS_3.2) [NSS SSL]	NSS_FindCertKEAType(NSS_3.2) [NSS SSL]	NSS_GetClientAuthData(NSS_3.2) [NSS SSL]
SSL_AuthCertificate(NSS_3.2) [NSS SSL]	SSL_AuthCertificateHook(NSS_3.2) [NSS SSL]	SSL_BadCertHook(NSS_3.2) [NSS SSL]
SSL_CipherPolicyGet(NSS_3.2) [NSS SSL]	SSL_CipherPolicySet(NSS_3.2) [NSS SSL]	SSL_CipherPrefGet(NSS_3.2) [NSS SSL]
SSL_CipherPrefGetDefault(NSS_3.2) [NSS SSL]	SSL_CipherPrefSet(NSS_3.2) [NSS SSL]	SSL_CipherPrefSetDefault(NSS_3.2) [NSS SSL]
SSL_ClearSessionCache(NSS_3.2) [NSS SSL]	SSL_ConfigMPServerSIDCache(NSS_3.2) [NSS SSL]	SSL_ConfigSecureServer(NSS_3.2) [NSS SSL]
SSL_ConfigServerSessionIDCache(NSS_3.2) [NSS SSL]	SSL_DataPending(NSS_3.2) [NSS SSL]	SSL_ForceHandshake(NSS_3.2) [NSS SSL]
SSL_GetClientAuthDataHook(NSS_3.2) [NSS SSL]	SSL_GetSessionID(NSS_3.2) [NSS SSL]	SSL_HandshakeCallback(NSS_3.2) [NSS SSL]
SSL_ImportFD(NSS_3.2) [NSS SSL]	SSL_InheritMPServerSIDCache(NSS_3.2) [NSS SSL]	SSL_InvalidateSession(NSS_3.2) [NSS SSL]
SSL_OptionGet(NSS_3.2) [NSS SSL]	SSL_OptionGetDefault(NSS_3.2) [NSS SSL]	SSL_OptionSet(NSS_3.2) [NSS SSL]
SSL_OptionSetDefault(NSS_3.2) [NSS SSL]	SSL_PeerCertificate(NSS_3.2) [NSS SSL]	SSL_ReHandshake(NSS_3.2) [NSS SSL]
SSL_ResetHandshake(NSS_3.2) [NSS SSL]	SSL_RevealPinArg(NSS_3.2) [NSS SSL]	SSL_RevealURL(NSS_3.2) [NSS SSL]
SSL_SecurityStatus(NSS_3.2) [NSS SSL]	SSL_SetPKCS11PinArg(NSS_3.2) [NSS SSL]	SSL_SetSockPeerID(NSS_3.2) [NSS SSL]
SSL_SetURL(NSS_3.2) [NSS SSL]		

7.6 Data Definitions for libssl3

This section defines global identifiers and their values that are associated with interfaces contained in libssl3. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

7.6.1 nss3/ecl-exp.h

```
#define __ecl_exp_h_
#define ECCurve_SECG_CHAR2_163R2        ECCurve_NIST_B163
#define ECCurve_SECG_CHAR2_233R1        ECCurve_NIST_B233
#define ECCurve_WTLS_11 ECCurve_NIST_B233
#define ECCurve_SECG_CHAR2_283R1        ECCurve_NIST_B283
#define ECCurve_SECG_CHAR2_409R1        ECCurve_NIST_B409
#define ECCurve_SECG_CHAR2_571R1        ECCurve_NIST_B571
#define ECCurve_SECG_CHAR2_163K1        ECCurve_NIST_K163
#define ECCurve_WTLS_3  ECCurve_NIST_K163
#define ECCurve_SECG_CHAR2_233K1        ECCurve_NIST_K233
#define ECCurve_WTLS_10 ECCurve_NIST_K233
#define ECCurve_SECG_CHAR2_283K1        ECCurve_NIST_K283
#define ECCurve_SECG_CHAR2_409K1        ECCurve_NIST_K409
#define ECCurve_SECG_CHAR2_571K1        ECCurve_NIST_K571
#define ECCurve_SECG_PRIME_192R1        ECCurve_NIST_P192
#define ECCurve_X9_62_PRIME_192V1       ECCurve_NIST_P192
#define ECCurve_SECG_PRIME_224R1        ECCurve_NIST_P224
#define ECCurve_WTLS_12 ECCurve_NIST_P224
#define ECCurve_SECG_PRIME_256R1        ECCurve_NIST_P256
#define ECCurve_X9_62_PRIME_256V1       ECCurve_NIST_P256
#define ECCurve_SECG_PRIME_384R1        ECCurve_NIST_P384
#define ECCurve_SECG_PRIME_521R1        ECCurve_NIST_P521
#define ECCurve_WTLS_4  ECCurve_SECG_CHAR2_113R1
#define ECCurve_WTLS_6  ECCurve_SECG_PRIME_112R1
#define ECCurve_WTLS_7  ECCurve_SECG_PRIME_160R1
#define ECCurve_WTLS_5  ECCurve_X9_62_CHAR2_PNB163V1

enum ECField {
    ECField_GFp = 0,
    ECField_GF2m = 1
};
typedef struct ECCurveParamsStr {
    char *text;
    enum ECField field;
    unsigned int size;
    char *irr;
    char *curvea;
    char *curveb;
    char *genx;
    char *geny;
    char *order;
    int cofactor;
} ECCurveParams;
enum ECCurveName {
    ECCurve_noName = 0,
    ECCurve_NIST_P192 = 1,
    ECCurve_NIST_P224 = 2,
    ECCurve_NIST_P256 = 3,
    ECCurve_NIST_P384 = 4,
    ECCurve_NIST_P521 = 5,
    ECCurve_NIST_K163 = 6,
```

```
    ECCurve_NIST_B163 = 7,
    ECCurve_NIST_K233 = 8,
    ECCurve_NIST_B233 = 9,
    ECCurve_NIST_K283 = 10,
    ECCurve_NIST_B283 = 11,
    ECCurve_NIST_K409 = 12,
    ECCurve_NIST_B409 = 13,
    ECCurve_NIST_K571 = 14,
    ECCurve_NIST_B571 = 15,
    ECCurve_X9_62_PRIME_192V2 = 16,
    ECCurve_X9_62_PRIME_192V3 = 17,
    ECCurve_X9_62_PRIME_239V1 = 18,
    ECCurve_X9_62_PRIME_239V2 = 19,
    ECCurve_X9_62_PRIME_239V3 = 20,
    ECCurve_X9_62_CHAR2_PNB163V1 = 21,
    ECCurve_X9_62_CHAR2_PNB163V2 = 22,
    ECCurve_X9_62_CHAR2_PNB163V3 = 23,
    ECCurve_X9_62_CHAR2_PNB176V1 = 24,
    ECCurve_X9_62_CHAR2_TNB191V1 = 25,
    ECCurve_X9_62_CHAR2_TNB191V2 = 26,
    ECCurve_X9_62_CHAR2_TNB191V3 = 27,
    ECCurve_X9_62_CHAR2_PNB208W1 = 28,
    ECCurve_X9_62_CHAR2_TNB239V1 = 29,
    ECCurve_X9_62_CHAR2_TNB239V2 = 30,
    ECCurve_X9_62_CHAR2_TNB239V3 = 31,
    ECCurve_X9_62_CHAR2_PNB272W1 = 32,
    ECCurve_X9_62_CHAR2_PNB304W1 = 33,
    ECCurve_X9_62_CHAR2_TNB359V1 = 34,
    ECCurve_X9_62_CHAR2_PNB368W1 = 35,
    ECCurve_X9_62_CHAR2_TNB431R1 = 36,
    ECCurve_SECG_PRIME_112R1 = 37,
    ECCurve_SECG_PRIME_112R2 = 38,
    ECCurve_SECG_PRIME_128R1 = 39,
    ECCurve_SECG_PRIME_128R2 = 40,
    ECCurve_SECG_PRIME_160K1 = 41,
    ECCurve_SECG_PRIME_160R1 = 42,
    ECCurve_SECG_PRIME_160R2 = 43,
    ECCurve_SECG_PRIME_192K1 = 44,
    ECCurve_SECG_PRIME_224K1 = 45,
    ECCurve_SECG_PRIME_256K1 = 46,
    ECCurve_SECG_CHAR2_113R1 = 47,
    ECCurve_SECG_CHAR2_113R2 = 48,
    ECCurve_SECG_CHAR2_131R1 = 49,
    ECCurve_SECG_CHAR2_131R2 = 50,
    ECCurve_SECG_CHAR2_163R1 = 51,
    ECCurve_SECG_CHAR2_193R1 = 52,
    ECCurve_SECG_CHAR2_193R2 = 53,
    ECCurve_SECG_CHAR2_239K1 = 54,
    ECCurve_WTLS_1 = 55,
    ECCurve_WTLS_8 = 56,
    ECCurve_WTLS_9 = 57,
    ECCurve_pastLastCurve = 58
};
```

7.6.2 nss3/ssl.h

```
#define __ssl_h_
#define SSL_IS_SSL2_CIPHER(which)         (((which) & 0xfff0) ==
0xff00)
#define SSL_REQUIRE_NEVER       ((PRBool)0)
#define SSL_REQUIRE_ALWAYS      ((PRBool)1)
#define SSL_REQUIRE_FIRST_HANDSHAKE     ((PRBool)2)
#define SSL_REQUIRE_NO_ERROR    ((PRBool)3)
#define SSL_SECURITY_STATUS_NOOPT       -1
```

```
#define SSL_NOT_ALLOWED 0
#define SSL_SECURITY_STATUS_OFF 0
#define SSL_ALLOWED     1
#define SSL_SECURITY    1
#define SSL_SECURITY_STATUS_ON_HIGH     1
#define SSL_REQUIRE_CERTIFICATE 10
#define SSL_ENABLE_FDX  11
#define SSL_V2_COMPATIBLE_HELLO 12
#define SSL_ENABLE_TLS  13
#define SSL_ROLLBACK_DETECTION  14
#define SSL_NO_STEP_DOWN        15
#define SSL_BYPASS_PKCS11       16
#define SSL_NO_LOCKS    17
#define SSL_RESTRICTED  2
#define SSL_SECURITY_STATUS_ON_LOW      2
#define SSL_SOCKS       2
#define SSL_REQUEST_CERTIFICATE 3
#define SSL_HANDSHAKE_AS_CLIENT 5
#define SSL_HANDSHAKE_AS_SERVER 6
#define SSL_ENABLE_SSL2 7
#define SSL_ENABLE_SSL3 8
#define SSL_NO_CACHE    9
#define SSL_ENV_VAR_NAME        "SSL_INHERITANCE"

typedef  SECStatus(*SSLAuthCertificate)  (void  *,  PRFileDesc  *,
PRBool,
                                          PRBool);
typedef SECStatus(*SSLGetClientAuthData) (void *, PRFileDesc *,
                                           CERTDistNames *,
                                           CERTCertificate * *,
                                           SECKEYPrivateKey * *);
typedef SECStatus(*SSLBadCertHandler) (void *, PRFileDesc *);
typedef void (*SSLHandshakeCallback) (PRFileDesc *, void *);
extern SECStatus NSS_CmpCertChainWCANames(CERTCertificate * cert,
                                                    CERTDistNames *
caNames);
extern SSLKEAType NSS_FindCertKEAType(CERTCertificate * cert);
extern  SECStatus  NSS_GetClientAuthData(void  *arg,  PRFileDesc  *
socket,
                                            struct CERTDistNamesStr
*caNames,
                                         struct CERTCertificateStr
                                         **pRetCert,
                                         struct SECKEYPrivateKeyStr
                                         **pRetKey);
extern SECStatus SSL_AuthCertificate(void *arg, PRFileDesc * fd,
                                            PRBool checkSig, PRBool
isServer);
extern SECStatus SSL_AuthCertificateHook(PRFileDesc * fd,
                                              SSLAuthCertificate f,
void *arg);
extern     SECStatus     SSL_BadCertHook(PRFileDesc     *     fd,
SSLBadCertHandler f,
                                  void *arg);
extern  SECStatus  SSL_CipherPolicyGet(PRInt32  cipher,  PRInt32  *
policy);
extern  SECStatus  SSL_CipherPolicySet(PRInt32  cipher,  PRInt32
policy);
extern  SECStatus  SSL_CipherPrefGet(PRFileDesc  *  fd,  PRInt32
cipher,
                                     PRBool * enabled);
extern SECStatus SSL_CipherPrefGetDefault(PRInt32 cipher,
                                          PRBool * enabled);
extern  SECStatus  SSL_CipherPrefSet(PRFileDesc  *  fd,  PRInt32
cipher,
                                     PRBool enabled);
```

```
extern SECStatus SSL_CipherPrefSetDefault(PRInt32 cipher, PRBool
enabled);
extern void SSL_ClearSessionCache(void);
extern SECStatus SSL_ConfigMPServerSIDCache(int maxCacheEntries,
                                            PRUint32 timeout,
                                                 PRUint32
ssl3_timeout,
                                                 const char
*directory);
extern SECStatus SSL_ConfigSecureServer(PRFileDesc * fd,
                                        CERTCertificate * cert,
                                        SECKEYPrivateKey * key,
                                        SSLKEAType kea);
extern        SECStatus        SSL_ConfigServerSessionIDCache(int
maxCacheEntries,
                                                PRUint32 timeout,
                                                  PRUint32
ssl3_timeout,
                                                 const char
*directory);
extern int SSL_DataPending(PRFileDesc * fd);
extern SECStatus SSL_ForceHandshake(PRFileDesc * fd);
extern SECStatus SSL_GetClientAuthDataHook(PRFileDesc * fd,
                                           SSLGetClientAuthData
f,
                                           void *a);
extern SECItem *SSL_GetSessionID(PRFileDesc * fd);
extern SECStatus SSL_HandshakeCallback(PRFileDesc * fd,
                                       SSLHandshakeCallback cb,
                                       void *client_data);
extern PRFileDesc *SSL_ImportFD(PRFileDesc * model, PRFileDesc *
fd);
extern    SECStatus    SSL_InheritMPServerSIDCache(const    char
*envString);
extern SECStatus SSL_InvalidateSession(PRFileDesc * fd);
extern SECStatus SSL_OptionGet(PRFileDesc * fd, PRInt32 option,
                               PRBool * on);
extern  SECStatus  SSL_OptionGetDefault(PRInt32  option,  PRBool  *
on);
extern SECStatus SSL_OptionSet(PRFileDesc * fd, PRInt32 option,
PRBool on);
extern SECStatus SSL_OptionSetDefault(PRInt32 option, PRBool on);
extern CERTCertificate *SSL_PeerCertificate(PRFileDesc * fd);
extern   SECStatus   SSL_ReHandshake(PRFileDesc   *   fd,   PRBool
flushCache);
extern  SECStatus  SSL_ResetHandshake(PRFileDesc  *  fd,  PRBool
asServer);
extern void *SSL_RevealPinArg(PRFileDesc * socket);
extern char *SSL_RevealURL(PRFileDesc * socket);
extern SECStatus SSL_SecurityStatus(PRFileDesc * fd, int *on,
                                    char **cipher, int *keySize,
                                         int *secretKeySize, char
**issuer,
                                    char **subject);
extern SECStatus SSL_SetPKCS11PinArg(PRFileDesc * fd, void *a);
extern   SECStatus   SSL_SetSockPeerID(PRFileDesc   *   fd,   char
*peerID);
extern SECStatus SSL_SetURL(PRFileDesc * fd, const char *url);
```

7.6.3 nss3/sslerr.h

```
#define __SSL_ERR_H_
#define IS_SSL_ERROR(code)       \
                  (((code)  >=  SSL_ERROR_BASE)  &&  ((code)  <
```



```
SSL_ERROR_LIMIT))
#define SSL_ERROR_BASE  (-0x3000)
#define SSL_ERROR_LIMIT (SSL_ERROR_BASE + 1000)

typedef enum {
    SSL_ERROR_EXPORT_ONLY_SERVER = (SSL_ERROR_BASE + 0),
    SSL_ERROR_US_ONLY_SERVER = (SSL_ERROR_BASE + 1),
    SSL_ERROR_NO_CYPHER_OVERLAP = (SSL_ERROR_BASE + 2),
    SSL_ERROR_NO_CERTIFICATE = (SSL_ERROR_BASE + 3),
    SSL_ERROR_BAD_CERTIFICATE = (SSL_ERROR_BASE + 4),
    SSL_ERROR_BAD_CLIENT = (SSL_ERROR_BASE + 6),
    SSL_ERROR_BAD_SERVER = (SSL_ERROR_BASE + 7),
      SSL_ERROR_UNSUPPORTED_CERTIFICATE_TYPE  =  (SSL_ERROR_BASE  +
8),
    SSL_ERROR_UNSUPPORTED_VERSION = (SSL_ERROR_BASE + 9),
    SSL_ERROR_WRONG_CERTIFICATE = (SSL_ERROR_BASE + 11),
    SSL_ERROR_BAD_CERT_DOMAIN = (SSL_ERROR_BASE + 12),
    SSL_ERROR_POST_WARNING = (SSL_ERROR_BASE + 13),
    SSL_ERROR_SSL2_DISABLED = (SSL_ERROR_BASE + 14),
    SSL_ERROR_BAD_MAC_READ = (SSL_ERROR_BASE + 15),
    SSL_ERROR_BAD_MAC_ALERT = (SSL_ERROR_BASE + 16),
    SSL_ERROR_BAD_CERT_ALERT = (SSL_ERROR_BASE + 17),
    SSL_ERROR_REVOKED_CERT_ALERT = (SSL_ERROR_BASE + 18),
    SSL_ERROR_EXPIRED_CERT_ALERT = (SSL_ERROR_BASE + 19),
    SSL_ERROR_SSL_DISABLED = (SSL_ERROR_BASE + 20),
    SSL_ERROR_FORTEZZA_PQG = (SSL_ERROR_BASE + 21),
    SSL_ERROR_UNKNOWN_CIPHER_SUITE = (SSL_ERROR_BASE + 22),
    SSL_ERROR_NO_CIPHERS_SUPPORTED = (SSL_ERROR_BASE + 23),
    SSL_ERROR_BAD_BLOCK_PADDING = (SSL_ERROR_BASE + 24),
    SSL_ERROR_RX_RECORD_TOO_LONG = (SSL_ERROR_BASE + 25),
    SSL_ERROR_TX_RECORD_TOO_LONG = (SSL_ERROR_BASE + 26),
    SSL_ERROR_RX_MALFORMED_HELLO_REQUEST = (SSL_ERROR_BASE + 27),
    SSL_ERROR_RX_MALFORMED_CLIENT_HELLO = (SSL_ERROR_BASE + 28),
    SSL_ERROR_RX_MALFORMED_SERVER_HELLO = (SSL_ERROR_BASE + 29),
    SSL_ERROR_RX_MALFORMED_CERTIFICATE = (SSL_ERROR_BASE + 30),
      SSL_ERROR_RX_MALFORMED_SERVER_KEY_EXCH  =  (SSL_ERROR_BASE  +
31),
    SSL_ERROR_RX_MALFORMED_CERT_REQUEST = (SSL_ERROR_BASE + 32),
    SSL_ERROR_RX_MALFORMED_HELLO_DONE = (SSL_ERROR_BASE + 33),
    SSL_ERROR_RX_MALFORMED_CERT_VERIFY = (SSL_ERROR_BASE + 34),
      SSL_ERROR_RX_MALFORMED_CLIENT_KEY_EXCH  =  (SSL_ERROR_BASE  +
35),
    SSL_ERROR_RX_MALFORMED_FINISHED = (SSL_ERROR_BASE + 36),
    SSL_ERROR_RX_MALFORMED_CHANGE_CIPHER = (SSL_ERROR_BASE + 37),
    SSL_ERROR_RX_MALFORMED_ALERT = (SSL_ERROR_BASE + 38),
    SSL_ERROR_RX_MALFORMED_HANDSHAKE = (SSL_ERROR_BASE + 39),
     SSL_ERROR_RX_MALFORMED_APPLICATION_DATA  =  (SSL_ERROR_BASE  +
40),
      SSL_ERROR_RX_UNEXPECTED_HELLO_REQUEST  =  (SSL_ERROR_BASE  +
41),
    SSL_ERROR_RX_UNEXPECTED_CLIENT_HELLO = (SSL_ERROR_BASE + 42),
    SSL_ERROR_RX_UNEXPECTED_SERVER_HELLO = (SSL_ERROR_BASE + 43),
    SSL_ERROR_RX_UNEXPECTED_CERTIFICATE = (SSL_ERROR_BASE + 44),
     SSL_ERROR_RX_UNEXPECTED_SERVER_KEY_EXCH  =  (SSL_ERROR_BASE  +
45),
    SSL_ERROR_RX_UNEXPECTED_CERT_REQUEST = (SSL_ERROR_BASE + 46),
    SSL_ERROR_RX_UNEXPECTED_HELLO_DONE = (SSL_ERROR_BASE + 47),
    SSL_ERROR_RX_UNEXPECTED_CERT_VERIFY = (SSL_ERROR_BASE + 48),
     SSL_ERROR_RX_UNEXPECTED_CLIENT_KEY_EXCH  =  (SSL_ERROR_BASE  +
49),
    SSL_ERROR_RX_UNEXPECTED_FINISHED = (SSL_ERROR_BASE + 50),
      SSL_ERROR_RX_UNEXPECTED_CHANGE_CIPHER  =  (SSL_ERROR_BASE  +
51),
    SSL_ERROR_RX_UNEXPECTED_ALERT = (SSL_ERROR_BASE + 52),
    SSL_ERROR_RX_UNEXPECTED_HANDSHAKE = (SSL_ERROR_BASE + 53),
     SSL_ERROR_RX_UNEXPECTED_APPLICATION_DATA = (SSL_ERROR_BASE +
```

```
54),
    SSL_ERROR_RX_UNKNOWN_RECORD_TYPE = (SSL_ERROR_BASE + 55),
    SSL_ERROR_RX_UNKNOWN_HANDSHAKE = (SSL_ERROR_BASE + 56),
    SSL_ERROR_RX_UNKNOWN_ALERT = (SSL_ERROR_BASE + 57),
    SSL_ERROR_CLOSE_NOTIFY_ALERT = (SSL_ERROR_BASE + 58),
    SSL_ERROR_HANDSHAKE_UNEXPECTED_ALERT = (SSL_ERROR_BASE + 59),
       SSL_ERROR_DECOMPRESSION_FAILURE_ALERT  =  (SSL_ERROR_BASE  +
60),
    SSL_ERROR_HANDSHAKE_FAILURE_ALERT = (SSL_ERROR_BASE + 61),
    SSL_ERROR_ILLEGAL_PARAMETER_ALERT = (SSL_ERROR_BASE + 62),
    SSL_ERROR_UNSUPPORTED_CERT_ALERT = (SSL_ERROR_BASE + 63),
    SSL_ERROR_CERTIFICATE_UNKNOWN_ALERT = (SSL_ERROR_BASE + 64),
    SSL_ERROR_GENERATE_RANDOM_FAILURE = (SSL_ERROR_BASE + 65),
    SSL_ERROR_SIGN_HASHES_FAILURE = (SSL_ERROR_BASE + 66),
    SSL_ERROR_EXTRACT_PUBLIC_KEY_FAILURE = (SSL_ERROR_BASE + 67),
       SSL_ERROR_SERVER_KEY_EXCHANGE_FAILURE  =  (SSL_ERROR_BASE  +
68),
       SSL_ERROR_CLIENT_KEY_EXCHANGE_FAILURE  =  (SSL_ERROR_BASE  +
69),
    SSL_ERROR_ENCRYPTION_FAILURE = (SSL_ERROR_BASE + 70),
    SSL_ERROR_DECRYPTION_FAILURE = (SSL_ERROR_BASE + 71),
    SSL_ERROR_SOCKET_WRITE_FAILURE = (SSL_ERROR_BASE + 72),
    SSL_ERROR_MD5_DIGEST_FAILURE = (SSL_ERROR_BASE + 73),
    SSL_ERROR_SHA_DIGEST_FAILURE = (SSL_ERROR_BASE + 74),
    SSL_ERROR_MAC_COMPUTATION_FAILURE = (SSL_ERROR_BASE + 75),
    SSL_ERROR_SYM_KEY_CONTEXT_FAILURE = (SSL_ERROR_BASE + 76),
    SSL_ERROR_SYM_KEY_UNWRAP_FAILURE = (SSL_ERROR_BASE + 77),
       SSL_ERROR_PUB_KEY_SIZE_LIMIT_EXCEEDED  =  (SSL_ERROR_BASE  +
78),
    SSL_ERROR_IV_PARAM_FAILURE = (SSL_ERROR_BASE + 79),
    SSL_ERROR_INIT_CIPHER_SUITE_FAILURE = (SSL_ERROR_BASE + 80),
    SSL_ERROR_SESSION_KEY_GEN_FAILURE = (SSL_ERROR_BASE + 81),
    SSL_ERROR_NO_SERVER_KEY_FOR_ALG = (SSL_ERROR_BASE + 82),
    SSL_ERROR_TOKEN_INSERTION_REMOVAL = (SSL_ERROR_BASE + 83),
    SSL_ERROR_TOKEN_SLOT_NOT_FOUND = (SSL_ERROR_BASE + 84),
    SSL_ERROR_NO_COMPRESSION_OVERLAP = (SSL_ERROR_BASE + 85),
    SSL_ERROR_HANDSHAKE_NOT_COMPLETED = (SSL_ERROR_BASE + 86),
    SSL_ERROR_BAD_HANDSHAKE_HASH_VALUE = (SSL_ERROR_BASE + 87),
    SSL_ERROR_CERT_KEA_MISMATCH = (SSL_ERROR_BASE + 88),
    SSL_ERROR_NO_TRUSTED_SSL_CLIENT_CA = (SSL_ERROR_BASE + 89),
    SSL_ERROR_SESSION_NOT_FOUND = (SSL_ERROR_BASE + 90),
    SSL_ERROR_DECRYPTION_FAILED_ALERT = (SSL_ERROR_BASE + 91),
    SSL_ERROR_RECORD_OVERFLOW_ALERT = (SSL_ERROR_BASE + 92),
    SSL_ERROR_UNKNOWN_CA_ALERT = (SSL_ERROR_BASE + 93),
    SSL_ERROR_ACCESS_DENIED_ALERT = (SSL_ERROR_BASE + 94),
    SSL_ERROR_DECODE_ERROR_ALERT = (SSL_ERROR_BASE + 95),
    SSL_ERROR_DECRYPT_ERROR_ALERT = (SSL_ERROR_BASE + 96),
    SSL_ERROR_EXPORT_RESTRICTION_ALERT = (SSL_ERROR_BASE + 97),
    SSL_ERROR_PROTOCOL_VERSION_ALERT = (SSL_ERROR_BASE + 98),
       SSL_ERROR_INSUFFICIENT_SECURITY_ALERT  =  (SSL_ERROR_BASE  +
99),
    SSL_ERROR_INTERNAL_ERROR_ALERT = (SSL_ERROR_BASE + 100),
    SSL_ERROR_USER_CANCELED_ALERT = (SSL_ERROR_BASE + 101),
    SSL_ERROR_NO_RENEGOTIATION_ALERT = (SSL_ERROR_BASE + 102),
       SSL_ERROR_SERVER_CACHE_NOT_CONFIGURED  =  (SSL_ERROR_BASE  +
103),
       SSL_ERROR_UNSUPPORTED_EXTENSION_ALERT  =  (SSL_ERROR_BASE  +
104),
     SSL_ERROR_CERTIFICATE_UNOBTAINABLE_ALERT = (SSL_ERROR_BASE +
105),
    SSL_ERROR_UNRECOGNIZED_NAME_ALERT = (SSL_ERROR_BASE + 106),
     SSL_ERROR_BAD_CERT_STATUS_RESPONSE_ALERT = (SSL_ERROR_BASE +
107),
    SSL_ERROR_BAD_CERT_HASH_VALUE_ALERT = (SSL_ERROR_BASE + 108)
} SSLErrorCodes;
```


7.6.4 nss3/sslproto.h

```
#define __sslproto_h_
#define SSL_MT_ERROR    0
#define SSL_NULL_WITH_NULL_NULL 0x0000
#define SSL_PE_NO_CYPHERS       0x0001
#define SSL_RSA_WITH_NULL_MD5   0x0001
#define SSL_LIBRARY_VERSION_2   0x0002
#define SSL_PE_NO_CERTIFICATE   0x0002
#define SSL_RSA_WITH_NULL_SHA   0x0002
#define SSL_RSA_EXPORT_WITH_RC4_40_MD5  0x0003
#define SSL_PE_BAD_CERTIFICATE  0x0004
#define SSL_RSA_WITH_RC4_128_MD5        0x0004
#define SSL_RSA_WITH_RC4_128_SHA        0x0005
#define SSL_PE_UNSUPPORTED_CERTIFICATE_TYPE     0x0006
#define SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5      0x0006
#define SSL_RSA_WITH_IDEA_CBC_SHA       0x0007
#define SSL_RSA_EXPORT_WITH_DES40_CBC_SHA       0x0008
#define SSL_RSA_WITH_DES_CBC_SHA        0x0009
#define SSL_RSA_WITH_3DES_EDE_CBC_SHA   0x000a
#define SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA    0x000b
#define SSL_DH_DSS_WITH_DES_CBC_SHA     0x000c
#define SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA        0x000d
#define SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA    0x000e
#define SSL_DH_RSA_WITH_DES_CBC_SHA     0x000f
#define SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA        0x0010
#define SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA   0x0011
#define SSL_DHE_DSS_WITH_DES_CBC_SHA    0x0012
#define SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA       0x0013
#define SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA   0x0014
#define SSL_DHE_RSA_WITH_DES_CBC_SHA    0x0015
#define SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA       0x0016
#define SSL_DH_ANON_EXPORT_WITH_RC4_40_MD5      0x0017
#define SSL_DH_ANON_WITH_RC4_128_MD5    0x0018
#define SSL_DH_ANON_EXPORT_WITH_DES40_CBC_SHA   0x0019
#define SSL_DH_ANON_WITH_DES_CBC_SHA    0x001a
#define SSL_DH_ANON_WITH_3DES_EDE_CBC_SHA       0x001b
#define SSL_FORTEZZA_DMS_WITH_NULL_SHA  0x001c
#define SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA  0x001d
#define SSL_FORTEZZA_DMS_WITH_RC4_128_SHA       0x001e
#define TLS_RSA_WITH_AES_128_CBC_SHA    0x002F
#define TLS_DH_DSS_WITH_AES_128_CBC_SHA 0x0030
#define TLS_DH_RSA_WITH_AES_128_CBC_SHA 0x0031
#define TLS_DHE_DSS_WITH_AES_128_CBC_SHA        0x0032
#define TLS_DHE_RSA_WITH_AES_128_CBC_SHA        0x0033
#define TLS_DH_ANON_WITH_AES_128_CBC_SHA        0x0034
#define TLS_RSA_WITH_AES_256_CBC_SHA    0x0035
#define TLS_DH_DSS_WITH_AES_256_CBC_SHA 0x0036
#define TLS_DH_RSA_WITH_AES_256_CBC_SHA 0x0037
#define TLS_DHE_DSS_WITH_AES_256_CBC_SHA        0x0038
#define TLS_DHE_RSA_WITH_AES_256_CBC_SHA        0x0039
#define TLS_DH_ANON_WITH_AES_256_CBC_SHA        0x003A
#define TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA     0x0062
#define TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA 0x0063
#define TLS_RSA_EXPORT1024_WITH_RC4_56_SHA      0x0064
#define TLS_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA  0x0065
#define TLS_DHE_DSS_WITH_RC4_128_SHA    0x0066
#define SSL_AT_MD5_WITH_RSA_ENCRYPTION  0x01
#define SSL_CK_RC4_128_WITH_MD5 0x01
#define SSL_CT_X509_CERTIFICATE 0x01
#define SSL_CK_RC4_128_EXPORT40_WITH_MD5        0x02
#define SSL_CK_RC2_128_CBC_WITH_MD5     0x03
#define SSL_LIBRARY_VERSION_3_0 0x0300
#define SSL_LIBRARY_VERSION_3_1_TLS     0x0301
```

```
#define SSL_CK_RC2_128_CBC_EXPORT40_WITH_MD5    0x04
#define SSL_CK_IDEA_128_CBC_WITH_MD5    0x05
#define SSL_CK_DES_64_CBC_WITH_MD5      0x06
#define SSL_CK_DES_192_EDE3_CBC_WITH_MD5        0x07
#define TLS_ECDH_ECDSA_WITH_NULL_SHA    0xC001
#define TLS_ECDH_ECDSA_WITH_RC4_128_SHA 0xC002
#define TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA    0xC003
#define TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA     0xC004
#define TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA     0xC005
#define TLS_ECDHE_ECDSA_WITH_NULL_SHA   0xC006
#define TLS_ECDHE_ECDSA_WITH_RC4_128_SHA        0xC007
#define TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA   0xC008
#define TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA    0xC009
#define TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA    0xC00A
#define TLS_ECDH_RSA_WITH_NULL_SHA      0xC00B
#define TLS_ECDH_RSA_WITH_RC4_128_SHA   0xC00C
#define TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA      0xC00D
#define TLS_ECDH_RSA_WITH_AES_128_CBC_SHA       0xC00E
#define TLS_ECDH_RSA_WITH_AES_256_CBC_SHA       0xC00F
#define TLS_ECDHE_RSA_WITH_NULL_SHA     0xC010
#define TLS_ECDHE_RSA_WITH_RC4_128_SHA  0xC011
#define TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA     0xC012
#define TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA      0xC013
#define TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA      0xC014
#define TLS_ECDH_anon_WITH_NULL_SHA     0xC015
#define TLS_ECDH_anon_WITH_RC4_128_SHA  0xC016
#define TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA     0xC017
#define TLS_ECDH_anon_WITH_AES_128_CBC_SHA      0xC018
#define TLS_ECDH_anon_WITH_AES_256_CBC_SHA      0xC019
#define SSL_RSA_FIPS_WITH_DES_CBC_SHA   0xfefe
#define SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA      0xfeff
#define SSL_RSA_OLDFIPS_WITH_3DES_EDE_CBC_SHA   0xffe0
#define SSL_RSA_OLDFIPS_WITH_DES_CBC_SHA        0xffe1
#define SSL_HL_CLIENT_FINISHED_HBYTES   1
#define SSL_HL_SERVER_FINISHED_HBYTES   1
#define SSL_HL_SERVER_VERIFY_HBYTES     1
#define SSL_MT_CLIENT_HELLO     1
#define SSL_HL_CLIENT_MASTER_KEY_HBYTES 10
#define SSL_HL_SERVER_HELLO_HBYTES      11
#define SSL_HL_REQUEST_CERTIFICATE_HBYTES       2
#define SSL_MT_CLIENT_MASTER_KEY        2
#define SSL_HL_ERROR_HBYTES     3
#define SSL_MT_CLIENT_FINISHED  3
#define SSL_MT_SERVER_HELLO     4
#define SSL_MT_SERVER_VERIFY    5
#define SSL_HL_CLIENT_CERTIFICATE_HBYTES        6
#define SSL_MT_SERVER_FINISHED  6
#define SSL_MT_REQUEST_CERTIFICATE      7
#define SSL_MT_CLIENT_CERTIFICATE       8
#define SSL_HL_CLIENT_HELLO_HBYTES      9
```

7.6.5 nss3/sslt.h

```
#define __sslt_h_

typedef enum {
    ssl_kea_null,
    ssl_kea_rsa = 1,
    ssl_kea_dh = 2,
    ssl_kea_fortezza = 3,
    ssl_kea_ecdh = 4,
    ssl_kea_size = 5
} SSLKEAType;
typedef enum {
```

```
    ssl_sign_null,
    ssl_sign_rsa = 1,
    ssl_sign_dsa = 2,
    ssl_sign_ecdsa = 3
} SSLSignType;
typedef enum {
    ssl_auth_null,
    ssl_auth_rsa = 1,
    ssl_auth_dsa = 2,
    ssl_auth_kea = 3,
    ssl_auth_ecdsa = 4
} SSLAuthType;
typedef enum {
    ssl_calg_null,
    ssl_calg_rc4 = 1,
    ssl_calg_rc2 = 2,
    ssl_calg_des = 3,
    ssl_calg_3des = 4,
    ssl_calg_idea = 5,
    ssl_calg_fortezza = 6,
    ssl_calg_aes = 7,
    ssl_calg_camellia = 8
} SSLCipherAlgorithm;
typedef enum {
    ssl_mac_null,
    ssl_mac_md5 = 1,
    ssl_mac_sha = 2,
    ssl_hmac_md5 = 3,
    ssl_hmac_sha = 4
} SSLMACAlgorithm;
typedef struct SSLChannelInfoStr {
    PRUint32 length;
    PRUint16 protocolVersion;
    PRUint16 cipherSuite;
    PRUint32 authKeyBits;
    PRUint32 keaKeyBits;
    PRUint32 creationTime;
    PRUint32 lastAccessTime;
    PRUint32 expirationTime;
    PRUint32 sessionIDLength;
    PRUint8 sessionID[31];
} SSLChannelInfo;
typedef struct SSLCipherSuiteInfoStr {
    PRUint16 length;
    PRUint16 cipherSuite;
    const char *cipherSuiteName;
    const char *authAlgorithmName;
    SSLAuthType authAlgorithm;
    const char *keaTypeName;
    SSLKEAType keaType;
    const char *symCipherName;
    SSLCipherAlgorithm symCipher;
    PRUint16 symKeyBits;
    PRUint16 symKeySpace;
    PRUint16 effectiveKeyBits;
    const char *macAlgorithmName;
    SSLMACAlgorithm macAlgorithm;
    PRUint16 macBits;
    PRUintn isFIPS:1;
    PRUintn isExportable:1;
    PRUintn nonStandard:1;
    PRUintn reservedBits:29;
} SSLCipherSuiteInfo;
```

IV Java Interpreter

8 Java Interpreter

8.1 Introduction

The Java interpreter API is described in the Java Platform SE 6 API, with the following requirements for an LSB conforming runtime.

8.2 Java Interpreter Location

The Java interpreter binary, or a link to the binary, shall exist at `/usr/bin/java`.

8.3 Java Interpreter Version

The default installed Java version shall be Java 6 Platform Standard Edition or greater. Applications can depend on the Java 6 Platform SE interfaces.

8.4 Operators and Functions

Core Java operators, subroutines, and built-in functions shall be present and shall operate as defined in Java Platform SE 6 API.

8.5 Java Interpreter Command

This section contains a description of the **java** command.

java

Name

java — launch a Java application

Synopsis

```
java [-?] [-Dproperty=value] [-classpath] [-cp] [-help] [-jar file.jar] [-version] [arguments...]
```

Description

The **java** command shall launch a Java application by initializing a Java runtime environment, loading the class specified on the command line, and calling the main method of that class.

The method's declaration must be as follows.

```
public static void main(String args[])
```

By default, the first argument specified that is not an option shall be the fully-qualified name of the class to invoke. This behavior may be changed by the addition of the *-jar* option, in which case the first argument specified that is not an option shall be the name of a JAR file. Any arguments specified after the class name or JAR file name that are not command-line options are passed to the main function.

The Java runtime shall search for the classes used, including the startup class, in three locations: the bootstrap class path, the extensions that are currently installed, and the user class path.

Options

-?

This option is a synonym for *-help*.

-classpath classpath

This option shall specify a list of locations to search for class files, including directories, JAR files, and ZIP files. Locations in the class path shall be delimited by colons (:).

The *-classpath* option shall override the CLASSPATH environment variable. If the *-classpath* option is not specified and CLASSPATH is not set, then the user class path shall simply consist of the current directory (.).

The class path element * shall specify all files in the current directory with the extension .jar. or .JAR. Example: if the only JAR files in the directory foo are bar.jar and bas.JAR, then the class path element foo/* shall be equivalent to bar.jar:bas.JAR. The order of JAR files in this case shall be unspecified. The CLASSPATH environment variable functions in the same way.

Wildcards in the class path shall be expanded before the initialization of the Java virtual machine. However, Java programs can examine unexpanded wildcards if they query the environment. Example: System.getenv("CLASSPATH").

-cp classpath

This option is a synonym for *-classpath*.

-Dproperty=value

This option shall set the value of a system property.

-help

This option shall display usage information for the command, then exit.

-jar jarfile

This option shall specify a JAR file containing a program to execute. Its first argument must be the name of a JAR file with both class and resource files. The JAR file's manifest must contain a line of this form: `Main-Class: classname`. The class name *classname* must specify the class containing the main function for the application.

If this option is specified, all other user class path settings shall be ignored.

JAR files that can be specified with this option shall also be runnable without this option by setting their execute permissions.

-version

This option shall display version information for the command, then exit.

V Trial Use Module

9 Trial Use Module

9.1 Introduction

The Trial Use Module decribes components in Trial Use status. Trial Use Specifications are non-mandatory components of the Linux Standard Base.

9.2 Xdg-utils

Xdg-utils is a set of command line utilities that assist applications with a variety of desktop integration tasks. Some of the utilities focus on tasks commonly required during the installation of a desktop application. The remainder focus on integration with the desktop environment while the application is running.

These utilities operate as described at xdg-utils reference

9.2.1 Xdg-utils Commands

An LSB conforming implementation shall provide the commands and utilities as described in Table 9-1, with at least the behavior described as mandatory in the referenced underlying specification, with the following exceptions:

1. If any operand (except one which follows `--`) starts with a hyphen, the behavior is unspecified.

Rationale (Informative): Applications should place options before operands, or use `--`, as needed. This text is needed because, by default, GNU option parsing differs from POSIX, unless the environment variable POSIXLY_CORRECT is set. For example, **ls . -a** in GNU **ls** means to list the current directory, showing all files (that is, `"."` is an operand and `-a` is an option). In POSIX, `"."` and `-a` are both operands, and the command means to list the current directory, and also the file named `-a`. Suggesting that applications rely on the setting of the POSIXLY_CORRECT environment variable, or try to set it, seems worse than just asking the applications to invoke commands in ways which work with either the POSIX or GNU behaviors.

Table 9-1 Commands And Utilities

xdg-desktop-icon [1]	xdg-email [1]	xdg-mime [1]	xdg-screensaver [1]	
xdg-desktop-menu [1]	xdg-icon-resource [1]	xdg-open [1]		

Referenced Specification(s)

[1]. xdg-utils reference

Annex A GNU Free Documentation License (Informative)

This specification is published under the terms of the GNU Free Documentation License, Version 1.1, March 2000

A.1 PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.2 APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A.3 VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.4 COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.5 MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.6 COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the ti-

tle of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

A.7 COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.8 AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

A.9 TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

A.10 TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or

rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.11 FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.12 How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.