

# **Preserving Backwards Compatibility**

## *A Proposal from The Open Group*

October 27th 1998

Andrew Josey (Email: [ajosey@opengroup.org](mailto:ajosey@opengroup.org))

### Abstract

PASC Resolution 9801-02 lays out the commitment to preserve backwards compatibility in amendments to POSIX.1-1990 and POSIX.2-1992. The issue of namespace is crucial to meeting this commitment. It is also crucial to ensuring no accidental divergences that would prevent IEEE POSIX standards from being integrated into future revisions of the Single UNIX Specification. This paper proposes a set of guidelines for namespace use in POSIX amendments and examples of how options can be added in a backwards compatible way.

The intended audience of this document is authors of amendments to POSIX.1-1990 and POSIX.2-1992.

Comments on this document should be sent to the author by electronic mail.

## ***Introduction***

PASC Resolution 9801-02 lays out the commitment to preserve backwards compatibility in amendments to POSIX.1-1990 and POSIX.2-1992. The issue of namespace is crucial to meeting the commitment. This paper proposes a set of guidelines and recommended practise to meet this resolution.

The intended audience of this document is authors of amendments to POSIX.1-1990 and POSIX.2-1992.

### **Guideline 1:**

The prefixes `POSIX_`, `posix_`, `_POSIX_`, and any upper- and lower-case letter variations shall be reserved for use in further amendments. Such uses will need to be coordinated (see guidelines 2 and 3 about use of existing reservations).

Any header may use the prefix `POSIX_`

Any header may use the prefix `posix_`

Any header may use the prefix `_POSIX_`

☞ Amendments introducing macro definitions should use the prefix `POSIX_` or `_POSIX_`, or in the cases of system limits `_MAX` or `_MIN`.

☞ Amendments introducing functions should use the prefix `posix_`

☞ Amendments introducing structure definitions should use the prefix `posix_` (or an existing reserved prefix, the rationale being to avoid interactions with user defined macros).

☞ Amendments introducing headers may use the prefix `posix_`

Identifiers used should be unique in the first 31 characters of the identifier.

Note that these rules do not call out all the possibilities permitted by ISO C but are a practical set of guidelines for compatibility. For example, users are not permitted to define macro names (or any other name) beginning with `_[A-Z_]`.

For example:

POSIX.1m adding the header `<ckpt.h>`, which could now be renamed to `<posix_ckpt.h>`, has no need to reserve the prefixes `ckpt_` and `CKPT_` since it should define such uses as `posix_ckpt_` and `POSIX_CKPT_`. Thus POSIX.1m would have no need to add any reserved symbols to section 2.7.2 of POSIX.1.

## **Guideline 2:**

Where a new feature is being added to an existing feature that has reserved namespace it is permissible to use that namespace, however such use needs to be coordinated. Also care needs to be taken not to break Guideline 3.

☞ Amendments extending the existing namespace are permissible but need to be coordinated.

Recommendation: A namespace reservation registration service be setup to coordinate the POSIX namespace.

The Open Group volunteers to offer a "Namespace Reservation Registry" to ensure that there are no unintentional conflicts within POSIX and also with the Single UNIX Specification.

For example:

If adding new `sysconf ( )` variables it is permissible to use the prefix `_SC_` , however such uses need to be conditional dependent on the implementation option, and coordinated.

Other examples follow:

- (a) Error Numbers beginning with the prefix `E[A-Z]*`
- (b) Additional limits macros using `_MAX` and `_MIN` suffixes
- (c) Datatypes , ending with the suffix `_t`
- (d) Additional functions using existing reserved prefixes, such as `pthread_` , `sem_` etc.
- (e) Additional structures using existing reserved prefixes.

*(ISSUE: Note that the UK objects to the general reservation of type-  
defs ending with `_t` )*

## **Where do we need to coordinate the Namespace?**

Initially in the headers that overlap between the C Standard, POSIX and the Single UNIX Specification. This includes the following headers:

C Standard Headers:

```
<assert.h> <ctype.h> <errno.h> <float.h> <limits.h>  
<locale.h> <math.h> <setjmp.h> <signal.h> <stdarg.h>  
<stddef.h> <stdio.h> <stdlib.h> <string.h> <time.h>
```

ISO C Amendment 1 adds the following headers:

```
<iso646.h> <wchar.h> <wctype.h>
```

POSIX.1 specifies 18 standard headers and subsumes the C standard headers, adding additional symbols to some of them.

<aio.h>	<pthread.h>	<sys/stat.h>	<tar.h>
<dirent.h>	<pwd.h>	<sys/times.h>	<termios.h>
<fcntl.h>	<sched.h>	<sys/types.h>	<unistd.h>
<grp.h>	<semaphore.h>	<sys/utsname.h>	<utime.h>
<mqueue.h>	<sys/mman.h>	<sys/wait.h>	

The modified C standard headers are the following:

<errno.h>	<setjmp.h>	<stdio.h>
<limits.h>	<signal.h>	<time.h>

-- The POSIX.2 C Language binding adds 4 headers:

<fnmatch.h> <glob.h> <regex.h> <wordexp.h>

The Single UNIX Specification subsumes POSIX.1-1996/POSIX.2-1992 and adds the following headers

<arpa/inet.h>	<strings.h>
<cpio.h>	<stropts.h>
<curses.h>	<sys/ipc.h>
<dlfcn.h>	<sys/msg.h>
<fmtmsg.h>	<sys/resource.h>
<ftw.h>	<sys/sem.h>
<iconv.h>	<sys/shm.h>
<inttypes.h>	<sys/socket.h>
<langinfo.h>	<sys/statvfs.h>
<libgen.h>	<sys/time.h>
<monetary.h>	<sys/timeb.h>
<ndbm.h>	<sys/uio.h>
<netdb.h>	<sys/un.h>
<netinet/in.h>	<syslog.h>
<nl_types.h>	<ucontext.h>
<poll.h>	<ulimit.h>
<re_comp.h>	<utmpx.h>
<regexp.h>	<varargs.h>
<search.h>	

### **Guideline 3:**

In preserving backwards compatibility new amendments shall be optional.  
This includes three very important concepts:

1. Amendments to existing headers will be optional. We need to make this clear in the normative text, and also to make this clear in the informative Annex C.
2. Amendments which introduce new functions should not require stubs. This means that the Description section for new functions has to be conditional. This is an intentional change to existing text in POSIX.1-1996 which includes stubs in some cases (for example, clause 11.2.6.2, page 249, line 287).
3. Amendments to existing text should be conditional on an implementation option, otherwise existing conforming systems will be made non-conforming when the amendment is approved. For the tables in **Section 2: Terminology and General Requirements** this will usually mean addition of a separate table listing optional values and the implementation options that they depend upon.

☞ New functionality added by an amendment shall be optional, and shall be clearly denoted.

## **Recommended Practise**

The amendments to date have not been successful in maintaining backwards compatibility. The examples below suggest a number of improved methods for making amendments to POSIX.1 to address the issue.

### **Clause 2.4 Error Numbers**

We need to make it clear that additional error numbers need only be provided if the appropriate option is supported. Without this a conforming implementation would be required to add the symbolic value for the error number to remain in conformance, even if it does not want to support the option.

It is recommended that the following addition be made to POSIX.1-1996, clause 2.4 after line 786:

*"Support for some error numbers is dependent on implementation options. Where an implementation option is not supported the symbolic name for that error number need not be found in the header <errno.h>."*

Error numbers introduced by amendments should be added into a section at the end of 2.4, for example:

*"If the XXXX option is supported, then the following symbolic names for error numbers are provided:*

*[EDISABLED] .....*

*Otherwise:*

*Either the implementation shall support the symbolic names as described above, or they shall not be provided."*

### **Clause 2.5 Primitive System Data Types**

We need to make it clear that additional primitive data types need be provided only if the appropriate option is supported. Without this a conforming implementation would be required to add these types to remain in conformance , even if it does not want to support the option.

It is recommended that the following addition be made to POSIX.1-1996 , clause 2.5 after line 962:

*"Support for some primitive data types is dependent on implementation options. Where an implementation option is not supported the primitive data types for that option need not be found in the header <sys/types.h>."*

Amendments which add primitive data types should be described in a new table added after POSIX.1-1996 line 996 with the title "**Optional Primitive System Data Types**". This table is similar to table 2-1 with an additional column titled "*Implementation Option*" to denote which option needs to be supported for an implementation to provide the type.

Defined Type	Description	Implementation Option
pthread_barrier_t	Used to identify a barrier	_POSIX_BARRIERS
etc....		

### **Clause 2.7.2 Table 2-2 - Reserved Header Symbols**

In general, amendments should not reserve additional symbols in table 2-2, since they should follow guideline 1 and use the reserved prefixes `posix_` and `POSIX_` .

### **Clause 2.7.2 `_POSIX_C_SOURCE`**

The value of `_POSIX_C_SOURCE` should remain unchanged, since changing its value will make existing implementations of POSIX.1 non conforming once the amendment is approved.

Flagging versions and presence of options has to be done using additional feature test macros not by modifying existing feature test macros if we are going to achieve backwards compatibility.

Each of the Compile-Time Symbolic Constants in Table 2-10 should be defined with the value `199ymmL` if that option is supported. That will allow an application not only to determine if the option is supported, but also determine if the implementation supports the version of that option corresponding to the version of the standard the user is referencing.

The following change is recommended to be made to POSIX.1-1996:

Expand the entries in table 2-10 "Compile-Time Symbolic Constants" L194-207 to also include Feature Test macros. Add the value `199ymmL` to each of the Compile-Time Symbolic Constants in Table 2-10.

### ***Clause 2.7.3 Header and Function Prototypes***

We need to make it clear that presence of certain headers and functions is dependent on implementation options. Without this existing implementations of POSIX.1 will be deemed non-conformant once an amendment is approved.

The following change is recommended to be added after POSIX.1-1996 , clause 2.7.3, line 1246:

*"Presence of some prototypes or declarations is dependent on implementation options. Where an implementation option is not supported the prototype or declaration need not be found in the header."*

Example text from a ballot on 1003.1d is as follows:

If the Advisory Information option is supported:

```
<fcntl.h>      posix_fadvise(), posix_madvise(),  
                posix_fallocate(), posix_ffree()
```

If the Message Passing option and the Timeouts option is supported:

```
<mqueue.h>     mq_timedsend(), mq_timedreceive()
```

If the Thread CPU-Time Clocks option is supported:

```
<pthread.h>    pthread_attr_setcpuclockallow(),  
                pthread_attr_getcpuclockallow(),  
                pthread_getcpuclockid()  
  
<time.h>      clock_getcpuclockid()
```

If the Threads option and the Timeouts option is supported:

```
<pthread.h>    pthread_mutex_timedlock(),  
                pthread_mutexattr_gettimeoutallow(),  
                pthread_mutexattr_settimeoutallow()
```

### ***Section 2.8.2 Minimum Values***

An amendment should not add values unconditionally to section 2.8.2, since doing so will require a conforming system to add the values to remain in conformance, even if it does not support the implementation option.

The following change is recommended to POSIX.1-1996 Clause 2.8.2 after line 1371:

*"Support for some minimum values is dependent on implementation options. Where an implementation option is not supported the minimum value or values associated with that option need not be found in the header <limits.h>."*

New minimum values should be added in a separate table with the title "*Table 2-3a - Optional Minimum Values*". This table should have the same format as per table 2-3 with an additional column to denote the implementation option associated with the minimum value.

#### ***Section 2.8.4 Run-time Invariant Values***

An amendment should not add values unconditionally to section 2.8.4, since doing so will require a conforming system to add the values even if it does not support the implementation option.

The following change is recommended to POSIX.1-1996 Clause 2.8.4 after line 1444:

*"Support for the run-time invariant values in Table 2-5a is dependent on implementation options. Where an implementation option is not supported the run-time invariant value or values associated with that option need not be found in the header <limits.h>, and need not be supported by the sysconf() function. "*

New run-time invariant values should be added in a separate table with the title "*Table 2-5a - Optional Run-Time Invariant Values*". This table should have the same format as per table 2-5 with an additional column to denote the implementation option associated with the value.

#### ***Section 2.9.3 \_POSIX\_VERSION***

The value of `_POSIX_VERSION` should remain unchanged, since changing its value will make existing implementations of POSIX.1 non conforming once the amendment is approved.

See the discussion on `_POSIX_C_SOURCE` .

#### ***Section 3.2.1.2 Wait for Process Termination***

We need to make it clear that additional macros need only be provided if the appropriate option is supported.

For example, in POSIX.1d additional macros can be added as follows:

Add after POSIX.1-1996 section 3.2.1.2 line 405

*"If the Spawn option is supported, then the following macros are provided:*

*WIFSPAWNFAIL(stat\_val) ....*

*WSPAWNERRNO(stat\_val) ....*

*Otherwise:*

*Either the implementation shall support the macros as described above, or they shall not be provided."*

### ***Section 4.8.1 Get Configuration System Variables***

We need to make it clear that additional configuration system variables need only be provided if the appropriate option is supported. Without this a conforming implementation would be required to be modified to stay in conformance , even if it does not want to support the option.

It is recommend that new options should be in a separate table to 4-2, since 4-2 has the mandatory configuration system variables that we've all agreed to support to date.

Its recommended that the following change to POSIX.1-1996 , clause 4.8.1 line 426 be made:

*"Support for some configuration system variables is dependent on implementation options (see table 4-3). Where an implementation option is not supported the variable need not be supported.*  
"

### ***Function Descriptions***

New functions should be optional and do not require stubs. New function descriptions should not require an ENOSYS error number to denote that the function is not supported.

*If the XXXXX option is supported:*

*The function .....*

*Otherwise:*

*Either the implementation shall support the .... function as described above, or the function shall not be provided.*

Notes:

We had considered and rejected making the function synopsis as follows but since section 2 makes it clear that headers can be optional this was deemed not necessary.

```
#include <unistd.h> /* Which POSIX options are supported */
#ifdef _POSIX_CKPT /* Is Checkpoint supported? */
#include <posix_ckpt.h>
int posix_ckpt_setup (struct posix_ckpt_args *args[], size_t nargs);
#endif
```