

By: Philippe Le Mouël  
Rogue Wave Software Inc.  
philippe@roguewave.com

Doc. No.: X3J16/96-0099  
WG21/N0917  
Date: May 25 1996

## Iostreams Issues List Library Clause 27

---

### Revision History

Pre-Stockholm	X3J16/96-0099 WG21/N0917
Post-Santa-Cruz	X3J16/96-0079 WG21/N0897
Pre-Santa Cruz	X3J16/96-0009 WG21/N0827
Post-Tokyo	X3J16/95-0221 WG21/N0821
Pre-Tokyo	X3J16/95-0194 WG21/N0794
Pre-Monterey	X3J16/95-0089 WG21/N0689
Pre-Austin	X3J16/95-0034 WG21/N0634

### Summary of Issues

#### 27.4.2 ios\_traits

P. 5

<b>Active</b>	27-001	Making newline locale aware	
<b>Active</b>	27-002	is_whitespace is inconsistent	
<b>Active</b>	27-004	example of changing the behavior of is_whitespace is incorrect.	
<b>Active</b>	27-005	not_eof specification	
<b>Active</b>	27-007	ios_traits typedefs are 'char' oriented	
<b>Active</b>	27-008	ios_traits::length is missing <b>Returns:</b> clause	
<b>Active</b>	27-009	ios_traits::get_state should be specified	
<b>Active</b>	27-010	ios_traits::get_pos should be specified	
<b>Active</b>	27-011	Return type for ios_traits::copy is incorrect	
<b>Closed</b>	27-003	Mention of base struct string_char_traits	(Tokyo)
<b>Closed</b>	27-006	streamsize should be <i>SZ_T</i> not <i>INT_T</i>	(Tokyo)

#### 27.4.3 ios\_base

P. 10

<b>Active</b>	27-106	Init class should be an implementation detail	
<b>Active</b>	27-107	ios::failure doesn't have the same functionality	
<b>Active</b>	27-108	cin, cout ... construction and initialization (Box)	
<b>Closed</b>	27-101	ios_base manipulators (Santa-Cruz)	
<b>Closed</b>	27-102	ios_base::width semantics are incorrect (Santa-Cruz)	
<b>Closed</b>	27-103	proposal for adding ios_base::maxwidth (Santa-Cruz)	

- Closed** 27-104 ios\_base unitbuf and nunitbuf manipulators (Santa-Cruz)
- Closed** 27-105 ios\_base storage functions are not exception safe (Santa-Cruz)

#### 27.4.4 basic\_ios

P. 11

- Active** 27-203 operator bool( ) needs to be fixed
- Active** 27-205 imbue should not call rdbuf( )->pubimbue
- Active** 27-206 clear() should not unconditionally clear the error state
- Active** 27-207 availability of char\_traits in header ios (Box)
- Active** 27-208 move member functions from basic\_ios to ios\_base (Box)
- Active** 27-209 imbueing getloc()::codecvt into the argument stream buffer (Box)
  
- Closed** 27-201 remove throw specifications for clear and setstate (Tokyo)
- Closed** 27-202 tie not required to be associated with an input sequence (Tokyo)
- Closed** 27-204 replace int\_type by char\_type in int\_type fill( ) and int\_type fill( int\_type ) (Tokyo)

#### 27.5.2 basic\_streambuf

P. 15

- Active** 27-301 imbuing on streambufs. When, how often, etc...
- Active** 27-312 sync does not say what happens to the input sequence (Box)
  
- Closed** 27-302 sungetc has an incorrect return type (Tokyo)
- Closed** 27-303 not\_eof needs to be used where appropriate (Santa-Cruz)
- Closed** 27-304 uflow needs editing (Santa-Cruz)
- Closed** 27-305 basic\_streambuf::showmanyc Incorrect return clause (Santa-Cruz)
- Closed** 27-306 basic\_streambuf::uflow has incorrect default behavior (Santa-Cruz)
- Closed** 27-307 basic\_streambuf::uflow has nonsense returns clause (Santa-Cruz)
- Closed** 27-308 streambuf inlines (Santa-Cruz)
- Closed** 27-309 two return clauses for streambuf::underflow (Santa-Cruz)
- Closed** 27-310 streambuf::pbackfail has incorrect **Notes:** clause (Santa-Cruz)
- Closed** 27-311 caching results of calls to locale functions (Santa-Cruz)

#### 27.6.1 basic\_istream

P. 16

- Active** 27-404 istream functions need to check for NULL streambuf
- Active** 27-405 confusing English in formatted requirements
- Active** 27-406 operator>>(char\_type \*) failure
- Active** 27-407 operator>>(char\_type) failure
- Active** 27-408 ws manipulator
- Active** 27-409 unsigned short extractors cannot use unsigned long get function
- Active** 27-410 putback function has wrong description
- Active** 27-411 getline should not set failbit when reading no characters
- Active** 27-412 operator >>(basic\_streambuf \*sb), should not set badbit if sb is null
- Active** 27-413 sentence missing (Box)
- Active** 27-414 readsome, putback and unget need to check for good (Box)
- Active** 27-415 streampos need to be replaced (Box)
  
- Closed** 27-401 isfx what does it do? (Santa-Cruz)
- Closed** 27-402 ipfx example is incorrect (Santa-Cruz)
- Closed** 47-403 Clarification of exceptions thrown (Santa-Cruz)

## 27.6.2 basic\_ostream

P. 21

- Active** 27-501 op<<(char) needs to be consistent with the other formatted inserters
- Active** 27-503 ostream functions need to check for NULL streambuf
- Active** 27-505 incorrect conversion specifier for operator<<(unsigned long)
- Active** 27-506 wrong default behavior for padding
  
- Closed** 27-502 op<<(void \*) should it be const volatile void \* (Santa-Cruz)
- Closed** 27-504 exceptions in ostream (Santa-Cruz)

## 27.6.1-27.6.2 basic\_istream, basic\_ostream

P. 23

- Active** 27-601 op[<<|>>](ios\_base&) needed for manipulators
- Active** 27-602 positional typedefs in istream/ostream derived classes are not needed
- Active** 27-603 read/write should take a void \* instead of a char\_type \*
- Active** 27-604 Should we require ios::in to be set for istream's and ios::out to be set for ostream's?
- Active** 27-605 Should get/put use iterators?
- Active** 27-606 seekg and seekp should have their first parameter passed by value.
- Active** 27-607 locale getnum needed for void\* extractor (Box)

## 27.6.3 Standard manipulators

P. 28

- Active** 27-651 setfill description is wrong
- Active** 27-652 smanip is not a single type

## 27.7 string streams

P. 29

- Active** 27-701 str() needs to clarify return value on else clause
- Active** 27-702 string stream classes need to have string\_char\_traits and allocator parameters
- Active** 27-703 stringbuf postconditions
- Active** 27-704 stringbuf::stringbuf constructor
- Active** 27-705 Incorrect calls to setg and setp in Table 14
- Active** 27-706 Incorrect calls to setg and setp in table 16
- Active** 27-707 setbuf function is missing

## 27.8 fstreams

P. 34

- Active** 27-801 filebuf::underflow example is incorrect
- Active** 27-802 filebuf::is\_open is a bit confusing
- Active** 27-803 ofstream constructor missing trunc as openmode
- Active** 27-804 ofstream::open missing trunc in openmode
- Active** 27-805 filebuf::imbue semantics
- Active** 27-806 filebuf::seekoff **Effects:** clause needs work
- Active** 27-807 filebuf::underflow performance questions
- Active** 27-808 Editorial fixes in wording for fstreams
- Active** 27-809 description of function setbuf is missing
- Active** 27-810 openmode notation is not consistent in basic\_ifstream and basic\_ofstream
- Active** 27-811 description of function sync is missing
- Active** 27-812 filebuf::overflow example is incorrect
- Active** 27-813 basic\_filebuf::overflow does not specify its return value on success
- Active** 27-814 basic\_filebuf::imbue has no description
- Active** 27-815 description of function seekpos is missing

**Active** 27-816 (i)(o)fstream open functions should not use is\_open

## Miscellaneous

**P. 42**

- Active** 27-901 input/output of unsigned char, and signed char
- Active** 27-904 iosfwd declarations incomplete
- Active** 27-906 add a typedef to access the traits parameter in each stream class
- Active** 27-907 Use of “instance of” vs. “version of” in descriptions of class ios
- Active** 27-908 unnecessary ‘;’ (semicolons) in tables
- Active** 27-909 Editorial issues (typo’s)
- Active** 27-910 remove streampos in favor of pos\_type
- Active** 27-911 stdio synchronization
- Active** 27-912 removing **Notes:** from the text
- Active** 27-913 Incorporating **Notes:** into the text
- Active** 27-914 rethrowing exceptions
- Active** 27-915 The use of specialization
- Active** 27-916 missing descriptions of specializations
- Active** 27-917 Editorial changes
- Active** 27-918 Validity of OFF\_T to POS\_T conversion
- Active** 27-919 Question on Table 2 assertions
- Active** 27-920 destination of clog and wlog
- Active** 27-921 default locale argument to constructor
- Active** 27-922 Cleaning the iostreams header synopsis #include of other headers
  
- Closed** 27-902 default locale (Santa-Cruz)
- Closed** 27-903 ipfx/opfx/isfx/osfx not compatible with exceptions (Santa-Cruz)
- Closed** 27-905 iostream type classes are missing (Santa-Cruz)

## Annex D

**P. 56**

- Active** 27-1001 description of function setbuf is not sufficient
- Active** 27-1002 strstreambuf Editorial issues
- Active** 27-1003 istrstream Editorial issues
- Active** 27-1004 ostrstream Editorial issues

**Total remaining active issues: 91**

## ios\_traits issues

**Issue Number:** 27-001  
**Title:** changing traits::newline to be locale aware  
**Section:** 27.4.2.1 **ios\_traits value functions** [**lib.ios.traits.values**]  
**Status:** active  
**Description:**

The problem with traits::newline is that it does not know about the currently imbued locale.

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends changing the description of functions *get* and *getline* so there are two variants instead of one with a default argument. Nathan Myers will provide text for the proposal.

**Requestor:** Nathan Myers (ncm@cantrip.org),  
John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-002  
**Title:** traits::is\_whitespace() is inconsistent  
**Section:** 27.4.2.2 **ios\_traits test functions** [**lib.ios.traits.tests**]  
**Status:** active  
**Description:**

This function is inconsistent throughout the document. For example:

**27.4.2 Template struct ios\_traits** [**lib.ios.traits**]  
static bool is\_whitespace(int\_type, const ctype<char\_type>&);

**27.4.2.2 ios\_traits test functions** [**lib.ios.traits.tests**]  
bool is\_whitespace(int\_type c, const ctype<char\_type>& ct);

**Returns:** true if *c* represents a white space character. The default definition is as if it returns *ct.isspace(c)*.

The returns paragraph calls a method of ctype that does not exist. It should say:

**Returns:** true if *c* represents a white space character. The default definition is as if it returns *ct.is(ct.space, c)*.

**27.6.1.1.2 basic\_istream::ipfx** [**lib.istream.prefix**]  
**Notes:** ...uses the function  
bool traits::is\_whitespace(charT, const ctype<charT>&)

The same paragraph goes on to use ctype<...> in the example.

**27.6.1.1.2 Paragraph 4:** [**lib.istream.prefix**]  
static bool is\_whitespace(char, const ctype<charT>&)

### Possible Resolution:

The Santa Cruz meeting, deprecates the `is_whitespace` function, by accepting doc: 96-0036R1=N0854R1 (Unification of Traits Revision1). The pre-Stockholm Iostreams WG recommends closing the issue with no action.

**Requestor:** John Hinke (hinke@roguewave.com)  
Philippe Le Mouël (philippe@roguewave.com)

---

**Issue Number:** 27-004  
**Title:** example of changing the behavior of `is_whitespace` is incorrect.  
**Section:** 27.6.1.1.2 Paragraph 4 **basic\_istream prefix and suffix** [**lib.istream.prefix**]  
**Status:** active  
**Description:**

Change from:

```
struct my_char_traits : public ios_traits<char> {
    static bool is_whitespace(char c, const ctype<charT>& ct)
        { ...my own implementation... }
};
```

To:

```
struct my_char_traits : public ios_traits<char> {
    static bool is_whitespace(char c, const ctype<char>& ct)
        { ...my own implementation... }
};
```

### Possible Resolution:

The Santa Cruz meeting, deprecates the `is_whitespace` function, by accepting doc: 96-0036R1=N0854R1 (Unification of Traits Revision1). The pre-Stockholm Iostreams WG recommends closing the issue with no action.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-005  
**Title:** `not_eof` specification  
**Section:** 27.4.2.1 `ios_traits` value functions [**lib.ios.traits.values**]  
**Status:** active  
**Description:**

```
int_type not_eof(int_type c);
```

Editorial: “**Notes:**” should also mention it is used for `sbumpc` and `sgetc`.

Per Bothner writes:

“The **Returns:** is incompatible with the traditional masking function for `zapeof`. This is because `int_type(-2) == -2` while `zapeof(-2) == ((-2) & 0xFF)`. And nowhere else does it say anything that would allow the traditional implementation.”

“I don’t understand the presentation style well enough to suggest the proper fix. But somewhere it should say or imply that when charT is specialized with char, then not\_eof(c) is int\_type((unsigned char)(c)).”

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends changing the specification of function *not\_eof*. It has to yield a value which is not equal to *eof*. The value is unspecified. ( The current WP requires that *not\_eof(e)==e* if *e!=eof()*, and forces a change from existing practice. The recommendation allows existing implementation to remain unchanged.) Tom Plum will provide text for the proposal.

**Requestor:** Per Bothner (bothner@cygnus.com)

---

**Issue Number:** 27-007  
**Title:** ios\_traits typedefs are ‘char’ oriented.  
**Section:** 27  
**Status:** active  
**Description:**

We cannot specify int\_type, off\_type, pos\_type, and state\_type corresponding to some specialized charT type.

For example, if in order to think about ‘char’ specialization, we might define

```
template <class charT> struct ios_traits {  
    ....  
    typedef charT char_type;  
    typedef int int_type;  
    ....  
};
```

we would have to accept it as constant definition in all of the specialized traits, not only ios\_traits<char>, but ios\_traits<wchar\_t>, ios\_traits<ultrachar>. This would lead to the restriction upon implementations that all of the charT must be converted in 'int' range. The restriction is too heavy for future wide character types and user-defined character types.

**Possible Resolution:**

The Santa Cruz meeting, fixes the problem, by accepting doc: 96-0036R1=N0854R1 (Unification of Traits Revision1). The pre-Stockholm Iostreams WG recommends closing the issue with no action.

**Requestor:** Norihiro Kumagai (kuma@slab.tnr.sharp.co.jp)

---

**Issue Number:** 27-008  
**Title:** ios\_traits::length is missing **Returns:** clause  
**Section:** 27.4.2.1 **ios\_traits value functions** [lib.ios.traits.values]  
**Status:** active  
**Description:**

ios\_traits::length has an **Effects:** clause but no **Returns:** clause. The **Effects:** clause should be reworded as a **Returns:** clause.

**Possible Resolution:**

The Santa Cruz meeting, fixes the problem, by accepting doc: 96-0036R1=N0854R1 (Unification of Traits Revision1). The pre-Stockholm Iostreams WG recommends closing the issue with no action.

**Requestor:** Public Comment

---

**Issue Number:** 27-009  
**Title:** definition for `get_state`  
**Section:** 27.4.2.3 **ios\_traits conversion functions** [`lib.ios.traits.convert`]  
**Status:** **active**  
**Description:**

The definition of `ios_traits::get_state` is incomplete. Here is the complete description:

```
state_type get_state(pos_type pos);
```

**Returns:** A `state_type` value which represents the conversion state in the object `pos`.

**Possible Resolution:**

No consensus was reached on this issue by the pre-Stockholm Iostreams WG. Jerry Schwarz will bring a proposal along these lines:

Add new trait type: `char_traits::upos_type`  
No constraints on this type.  
Replace `get_pos` with: `pos_type make_pos(upos_type, state_type)`  
Add new function: `upos_type get_pos(pos_type)`

**Requestor:** Norihiro Kumagai (kuma @ slab.tnr.sharp.co.jp)

---

**Issue Number:** 27-010  
**Title:** definition for `get_pos`  
**Section:** 27.4.2.3 **ios\_traits conversion functions** [`lib.ios.traits.convert`]  
**Status:** **active**  
**Description:**

The definition of `ios_traits::get_pos` is incomplete. Here is the complete description:

```
pos_type get_pos(streampos pos, state_type s);
```

**Effects:** Constructs a `pos_type` value which represents the stream position corresponding to the pair of `pos` and `s`.

**Returns:** A `pos_type` value which consists of the values of `pos` and `s`.

**Possible Resolution:**

No consensus was reached on this issue by the pre-Stockholm Iostreams WG. Jerry Schwarz will bring a proposal along these lines:



Add new trait type: `char_traits::upos_type`  
No constraints on this type.  
Replace `get_pos` with: `pos_type make_pos(upos_type, state_type)`  
Add new function: `upos_type get_pos(pos_type)`

**Requestor:** Norihiro Kumagai (kuma @ slab.tnr.sharp.co.jp)

---

**Issue Number:** 27-011  
**Title:** Return type for `ios_traits::copy` is incorrect  
**Section:** 27.4.2.3 **ios\_traits conversion functions** [`lib.ios.traits.convert`]  
**Status:** **active**  
**Description:**

The return type for `ios_traits::copy` says to return `dst`. It should return `dest`.

**Possible Resolution:**

The Santa Cruz meeting, fixes the problem, by accepting doc: 96-0036R1=N0854R1 (Unification of Traits Revision1). The pre-Stockholm Iostreams WG recommends closing the issue with no action.

**Requestor:** John Hinke (hinke@roguewave.com)

---

## ios\_base issues

**Issue Number:** 27-106  
**Title:** Init class should be an implementation detail  
**Section:** 27.4.3.1.6 Class `ios_base::Init` [`lib.ios::Init`]  
**Status:** active  
**Description:**

I fail to see why the Init class is part of the normative Standard. It is an implementation detail and hence, belongs in the realm of the implementor, not in the Standard.

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends no change to the WP. (Jerry Schwarz might add an example showing why Init is needed.)

**Requestor:** Public Comment

---

**Issue Number:** 27-107  
**Title:** `ios::failure` doesn't have the same functionality  
**Section:** 27.4.3 Class `ios_base` [`lib.ios.base`]  
**Status:** active  
**Description:**

Long ago when I originally proposed `ios::failure` I put the stream into it (as a reference). It now doesn't have that functionality. I don't know if it was removed deliberately or just got dropped inadvertently. I think it should be there.

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends no change to the WP. (Stream lifetime might end before lifetime of the reference.)

**Requestor:** Jerry Schwarz (jss@a.crl.com)

---

**Issue Number:** 27-108  
**Title:** `cin`, `cout` ... construction and initialization (Box)  
**Section:** 27.3 Standard iostream objects [`lib.istream.objects`]  
**Status:** active  
**Description:**

The standard iostreams objects (`cin`, `cout`, `cerr`, `clog`, `wcin`, `wcout`, `wcerr`, `wclog`) need to be constructed and associations established before dynamic initialization of file scope variables is begun.

**Possible Resolution:**

**Requestor:** pre-Stockholm Iostreams WG

## basic\_ios issues

**Issue Number:** 27-203  
**Title:** operator bool() needs to be fixed  
**Section:** 27.4.4.3 **basic\_ios iostate flags functions** [lib.iostate.flags]  
**Status:** active  
**Description:**

Defining ios\_base (or, as it appears in my copy of the WP, basic\_ios) with a member operator bool() seemed like a good idea at the time, but perhaps the change should be withdrawn.

The reason is: while a conversion to void\* is mostly harmless because few functions accept a void\* argument, and void\* doesn't silently convert to anything else, with an operator bool, the following absurdities are well-defined:

```
1 + cin
sin(cin)
vector<int> v(cin);
```

and (worse) ambiguities like

```
void f(istreambuf_iterator<char>);
void f(double);

f(cin); // ambiguous
```

have been introduced. In other words, this change broke reasonable code. The problem is that bool is an arithmetic type, and is ill-behaved.

### **Possible Resolution:**

Replace the member ios\_base::operator bool() with member ios\_base::operator const void\*(), specified to return 0 if fail() is true, and non 0 if it is false.

This restores the code we broke, and also prevents frustrating ambiguities in new code.

The pre-Stockholm Iostreams WG recommends to wait for the final resolution of bool properties before taking any action.

**Requestor:** Nathan Myers (ncm@cantrip.org)

---

**Issue Number:** 27-205  
**Title:** imbue should not call rdbuf()->pubimbue  
**Section:** 27.4.4.2 **Member functions** [lib.basic.ios.members]  
**Status:** active  
**Description:**

basic\_ios::imbue(const locale&) should call rdbuf()->pubimbue(loc) only if rdbuf() is not a null pointer. Even better, it should not call rdbuf()->pubimbue(loc) at all. Changing the locale that

controls stream conversions is best separated from changing the locale that affects numeric formatting, etc. Anyone who knows how to imbue a proper pair of codecvt facets in a streambuf won't mind having to make an explicit call.

### Possible Resolution:

The first part of the issue has already been resolved, the description of function *locale imbue(const locale& loc)* says:

**Effects:** Calls `ios_base::imbue(loc)` (27.4.3.3) and if `rdbuf()` != 0 then `rdbuf()->pubimbue(loc)` (27.5.2.2.1).

Concerning the second part you have two possibilities:

If we leave the `basic_ios::imbue` function unchanged, when users call the `imbue` function from `basic_istream`, `basic_ostream` or the classes derived from them, they are actually changing both the locale of the stream object, and the locale of the stream buffer object attached to the stream object. This is not a real problem, because the stream object is only affected by the `ctype`, `num_get`, `num_put`, and `numpunct` facets, while the stream buffer object is affected by the `codecvt` facet. Therefore, even if you want to have several stream pointing to the same stream buffer objects (with a different locale object for each of them), you can easily do it by having all the different stream locale objects having the same `codecvt` facet. You could also do it by imbuing the stream buffer at last. The advantage of this scheme is that in simple cases you need to imbue just once in the stream object without having to wonder about the stream buffer object attached to it. The drawback is that you need to be more careful when you imbue in a stream object, and make sure that the locale object you are imbuing contains the correct `codecvt` facet; otherwise, you need to imbue the stream buffer object attached to the stream with another locale.

The other possibility is to remove the call to `rdbuf()->pubimbue( loc )`, in which case you just imbue the stream object itself. The problem is that in simple cases you need to imbue both the stream object and the stream buffer attached to it. The advantage is you only imbue the object that needs to have a change of locale.

The pre-Stockholm Iostreams WG deferred the issue to Stockholm.

<b>Requestor:</b>	Public Comment
<b>Issue Number:</b>	27-206
<b>Title:</b>	<code>clear()</code> should not unconditionally clear the error state
<b>Section:</b>	27.4.4.3 <b>basic_ios iostate flags functions</b> [ <code>lib.iostate.flags</code> ]
<b>Status:</b>	active
<b>Description:</b>	

Problem: `clear()` can be used to unconditionally clear the error state, even if `rdbuf()`, returns null. The stream then appears to be in a good state, which is wrong.

### Possible Resolution:

The function `clear()` should set `badbit` (independent of its argument) if `rdbuf()` returns null.

Note: This means `clear()` must not be moved to `ios_base`; it must remain in `basic_ios`. Philippe Le Mouël will provide text for the WP.

**Requestor:** pre-Stockholm Iostreams WG

---

**Issue Number:** 27-207  
**Title:** availability of char\_traits in header ios (Box)  
**Section:** 27.4.2 **Template struct ios\_traits** [**lib.ios.traits**]  
**Status:** active  
**Description:**

The *ios\_traits* was previously defined in header *ios*. Character traits have now been consolidated into *char\_traits* defined in the string header. Should they still be available after a `#include<ios>`.

**Possible Resolution:**

**Requestor:** pre-Stockholm Iostreams WG

---

**Issue Number:** 27-208  
**Title:** move member functions from basic\_ios to ios\_base (Box)  
**Section:** 27.4.3 **Class ios\_base** [**lib.ios.base**]  
**Status:** active  
**Description:**

Move the following declarations from basic\_ios to ios\_base:

```
// 27.4.4.3 iostate flags:  
operator bool() const  
bool operator! () const  
  
iostate rdststate() const  
void clear (iostate state = goodbit)  
void setstate(iostate state)  
bool good() const  
bool eof() const  
bool fail() const  
bool bad() const
```

```
ios_base& copyfmt (const ios_base& rhs)
```

Note that there will be a version of copyfmt() specified for basic\_ios.

Note: void clear (iostate state = goodbit) and void setstate(iostate state) should not be moved see issue 27-206.

**Possible Resolution:**

**Requestor:** pre-Stockholm Iostreams WG

---

**Issue Number:** 27-209  
**Title:** imbueing getloc()::codecvt into the argument stream buffer (Box)  
**Section:** 27.4.4.2 **Member functions** [**lib.basic.ios.members**]  
**Status:** active  
**Description:**

The first Box of section 27.4.4.2 says:

Note: need to modify so as to describe the occurrence of imbueing getloc()::codecvt into the argument stream buffer.

**Possible Resolution:**

**Requestor:** pre-Stockholm Iostreams WG

---

## basic\_streambuf issues

**Issue Number:** 27-301  
**Title:** imbuing on streambufs: when, how often, etc...  
**Section:** 27.5.2.2.1 **Locales** [[lib.streambuf.locales](#)]  
**Status:** **active**  
**Description:**

There needs to be something said as to when a new locale can be imbued into a streambuf or stream. Which operations are considered “atomic” in regards to locale changes.

**Possible Resolution:**

Nathan wrote:

“The effect of calling `imbue` during activation of any member of a class derived from `basic_ios<>`, or of any operator `<<` or `>>` in which the class is the left argument, is unspecified. In particular (e.g.) any `codeset` conversion occurring in the streambuf may become incompatible with the formats specified by the old locale and still used.

The effect of calling `streambuf::imbue` or `pub_imbue` during activation of any streambuf virtual member is also undefined.”

The pre-Stockholm Iostreams WG recommends that no standard function defined in Chapter 27 calls `imbue` directly, except for the `imbue` function. If any user code (e.g. if called via a virtual function) calls `imbue` during the execution of any of these functions, the effect is undefined. Nathan Myers will provide text for the proposal.

**Requestor:** Nathan Myers ([ncm@cantrip.org](mailto:ncm@cantrip.org))

---

**Issue Number:** 27-312  
**Title:** `sync` does not say what happens to the input sequence (Box)  
**Section:** 27.5.2.4.2 **Buffer management and positioning** [[lib.streambuf.virt.buffer](#)]  
**Status:** **active**  
**Description:**

The description of function `sync` does not say what happens to the input sequence (if there is one).

Is it possible to synchronize the input sequence in all cases? If not, can we liberalize this specification to accommodate those constraints ?

**Possible Resolution:**

**Requestor:** pre-Stockholm Iostreams WG

---

## basic\_istream issues

**Issue Number:** 27-404  
**Title:** istream functions need to check for NULL streambuf  
**Section:** 27.6.1.1 **Template class basic\_istream [lib.istream]**  
**Status:** active  
**Description:**

Functions in basic\_istream that call members of rdbuf() need to check for a NULL streambuf before calling the function. There are some functions that make sure rdbuf() is not a NULL pointer before calling any functions on the buffer, but some functions don't check for the NULL pointer. This needs to be consistent.

**Discussion:**

P.J. Plauger wrote: "Any attempt to store a null stream buffer pointer causes badbit to be set in the stored status. Hence, no input or output is ever attempted, using such a pointer, by formatted functions."

**Possible Resolution:**

As pointed out by P.J. Plauger, we should add a footnote to explain why there is no need to check for a NULL streambuf.

We should also add, in section 27.4.4.2 **Member functions [lib.basic.ios.members]**, the following to the description of basic\_streambuf<charT,traits>\*

```
rdbuf(basic_streambuf<charT,traits>* sb); :
```

**Postcondition:** sb == rdbuf() and if sb is a NULL pointer rdstate() == badbit.

Note: This issue has to be discussed with issue 27-503.

The pre-Stockholm Iostreams WG recommends the issue to be treated as editorial. It needs to be pointed out where the check is not necessary. In places where check is required, recommend add the check. Jerry Schwarz will provide text for the proposal.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-405  
**Title:** confusing English in formatted requirements  
**Section:** 27.6.1.2.1 **Common requirements [lib.istream.formatted.reqmts]**  
**Status:** active  
**Description:**

27.6.1.2.1 [lib.istream.formatted.reqmts]: Paragraph 5: "In case the converting result is a value of either an integral type ... or a float type ... performing to parse and convert the result depend on the imbued locale object." This is really French converted to English by translation software, right? :->}

**Possible Resolution:**



The pre-Stockholm Iostreams WG recommends closing the issue since the rewrite of the section fixed this problem.

**Requestor:** Public Comment

---

**Issue Number:** 27-406  
**Title:** operator>>(char\_type \*) failure  
**Section:** 27.6.1.2.2 **basic\_istream::operator>>** [**lib.istream::extractors**]  
**Status:** active  
**Description:**

27.6.1.2.2 [**lib.istream::extractors**]: Paragraph 2: "If the function stores no characters, it calls `setstate(failbit)`, which may throw `ios_base::failure` (27.4.4.3). In any case, it then stores a null character ...." How can it store anything if an exception is thrown? C++ does not use the resumption model for exception handling. Different language than "In any case" is needed here.

**Possible Resolution:**

Change paragraph 2 to:

"If the function stores no characters, it calls `setstate(failbit)`, which may throw `ios_base::failure` (27.4.4.3)."

Add paragraph 3:

"Before returning or throwing an exception the function stores a null character into the next successive location of the array and calls `width(0)`."

Jerry Schwarz fixed the problem as editorial with the approval of the pre-Stockholm Iostreams WG. Therefore the issue should be closed.

**Requestor:** Public Comment

---

**Issue Number:** 27-407  
**Title:** operator>>(char\_type) failure  
**Section:** 27.6.1.2.2 **basic\_istream::operator>>** [**lib.istream::extractors**]  
**Status:** active  
**Description:**

`basic_istream<charT,traits>& operator>>(char_type& c);`

**Effects:** Extracts a character, if one is available, and stores it in `c`. Otherwise, the function calls `setstate(failbit)`.

Not eofbit?

**Possible Resolution:**

In 27.6.1.2.1 **Common requirements** [**lib.istream.formatted.reqmts**] paragraph 8 says:

"If the scan fails for any reason, the formatted input function calls `setstate(failbit)`, which may throw `ios_base::failure` (27.4.4.3)."

This is one of the requirements for all the formatted input functions. Because of this the user can call the `ios_base` member function `fail()` or the operator `bool()` to check if the extraction failed. The user can therefore write code like this:

The pre-Stockholm Iostreams WG recommends closing the issue since the definition of “extract” (27.6.1.1) covers setting eof.

Note: footnotes could be added referring to that section. (Editorial)

**Requestor:** Public Comment

---

**Issue Number:** 27-408  
**Title:** ws manipulator  
**Section:** 27.6.1.4 **Standard basic\_istream manipulators** [**lib.istream.manip**]  
**Status:** **active**  
**Description:**

27.6.1.4 [lib.istream.manip]: "... saves a copy of is.fmtflags ...."  
Should this not read "... saves a copy of is.flags ...."?

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends a rewrite of the explanation. The intent is clear: Extracts any whitespace. Sets eof if eof is reached. Jerry Schwarz will provide text for the proposal.

**Requestor:** Public Comment

---

**Issue Number:** 27-409  
**Title:** unsigned short extractors cannot use unsigned long get function  
**Section:** 27.6.1.2.2 basic\_istream ::operator>> [**lib.istream::extractors**]  
**Status:** **active**  
**Description:**

Unsigned short (and unsigned int) extractors cannot use unsigned long get function in num\_get. It cannot distinguish certain valid inputs from errors.

**Possible Resolution:**

P.J. Plauger wrote: “num\_get should add a get function ( and underlying do get) for unsigned short and unsigned int extractions. Otherwise, input values in the range -1 through -USHRT\_MAX (or -UINT\_MAX) look erroneous, and cannot be distinguished from truly erroneous values.”

The pre-Stockholm Iostreams WG recommends closing the issue. These functions have been added to the WP, see 22.2.2.1 Template class num\_get.

**Requestor:** P.J. Plauger (plauger!pjp@uunet.uu.net)

---

**Issue Number:** 27-410  
**Title:** putback function has wrong description  
**Section:** 27.6.1.3 Unformatted input functions [**lib.istream.unformatted**]  
**Status:** **active**  
**Description:**

The description of the putback function is incorrect.

**Possible Resolution:**

The complete description of the function should be:

```
basic_istream<charT,traits>& putback(char_type c);
```

**Effects:** If `rdbuf()` is not null, calls `rdbuf()->sputback(c)`. If `rdbuf()` is null, or if `sputback(c)` returns `traits::eof()`, calls `setstate(badbit)` (which may throw `ios_base::failure` (27.4.4.3)).

**Returns:** `*this`.

Jerry Schwarz fixed the problem as editorial with the approval of the pre-Stockholm Iostreams WG. Therefore the issue should be closed.

---

**Requestor:** Philippe Le Mouël (philippe@roguewave.com)

**Issue Number:** 27-411

**Title:** getline should not set failbit when reading no characters

**Section:** 27.6.1.3 Unformatted input functions [**lib.istream.unformatted**]

**Status:** active

**Description:**

When the function `getline` is called and the stream has a line that contains no text, `ios_base::failbit` is set on the input stream (which may throw `ios_base::failure`). While consistent with the behavior of the similar function named `get`, the behavior is quite inconvenient. Furthermore, I tested this behavior on the AT&T Release 3.0 implementation of Iostreams and I did not encounter the problem described above. The same comment also apply to the `string`'s `getline` function described in section 21.1.1.10.8 Inserters and extractors [**lib.string.io**].

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends changing the WP, if it does not say: `getline` fails if no characters are extracted. An empty line is not a failure.

Jerry Schwarz will provide text for the proposal.

---

**Requestor:** Public Comment

**Issue Number:** 27-412

**Title:** `operator >>(basic_streambuf *sb)`, should not set `badbit` if `sb` is null

**Section:** 27.6.1.2.2 `basic_istream::operator >>` [**lib.istream::extractors**]

**Status:** active

**Description:**

`basic_istream::operator >>(basic_streambuf *sb)` now says, "If `sb` is null, calls `setstate(badbit)`." This requirement was added without committee approval. It is also inconsistent with the widespread convention that `badbit` should report loss of integrity of the stream proper (not some other stream). A null `sb` should set `failbit`.

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends setting *failbit* not *badbit*.

Jerry Schwarz will provide text for the proposal.

---

**Requestor:** Public Comment

**Issue Number:** 27-413  
**Title:** sentence missing (Box)  
**Section:** 27.6.1.1 **Template class basic\_istream** [lib.istream]  
**Status:** active  
**Description:**

X3J16/95-0149==WG21/N0749 deleted a sentence that prevented the formatted and unformatted input functions from calling other streambuf virtuals (such as seek).

**Possible Resolution:**

**Requestor:** pre-Stockholm Iostreams WG

---

**Issue Number:** 27-414  
**Title:** readsome, putback and unget need to check for good (Box)  
**Section:** 27.6.1.3 **Unformatted input functions** [lib.istream.unformatted]  
**Status:** active  
**Description:**

basic\_istream::readsome, basic\_istream::putback and basic\_istream::unget should not call virtual if good() returns false.

**Possible Resolution:**

**Requestor:** pre-Stockholm Iostreams WG

---

**Issue Number:** 27-415  
**Title:** streampos need to be replaced (Box)  
**Section:** 27.6.1.3 **Unformatted input functions** [lib.istream.unformatted]  
**Status:** active  
**Description:**

*tellg* and *tellp* refer to the type *streampos* that has been moved to Annex D as per Monterey resolution 35. That resolution did not say what to do with the functions that use *streampos* as an argument type, so they are left here. (see 27-910)

**Possible Resolution:**

**Requestor:** pre-Stockholm Iostreams WG

---

## basic\_ostream issues

**Issue Number:** 27-501  
**Title:** ostream<<(char) : formatting, padding, width  
**Section:** 27.6.2.4.2 **basic\_ostream::operator<<** [lib.ostream.inserters]  
**Status:** active  
**Description:**

For historical reasons, this function has usually ignored padding and formatting. In the WP, it does not mention anything about ignoring padding or formatting. This needs to be clarified.

Reasons for ignoring padding on op<<(char):

1. Historical reasons/compatibility

Reasons for full formatting on op<<(char):

1. put(char) currently does no formatting. But there is no way to insert a char with formatting.
2. Some implementations do formatting.

Since put can insert a character without formatting, there needs to be a way to insert a character with formatting. Currently this does not exist. It would be nice not to introduce an inconsistency with the other formatted inserters, but it would also be nice to provide compatibility. I think that consistency would be much better in this case than compatibility.

### **Possible Resolution:**

At the Tokyo meeting the straw vote gave the following result:

5 for past practice (no padding), 1 for consistency.

The pre-Stockholm Iostreams WG deferred the issue to Stockholm.

**Requestor:** John Hinke (hinke@roguewave.com),  
Bernd Eggink (admin@rrz.uni-hamburg.de)

---

**Issue Number:** 27-503  
**Title:** ostream functions need to check for NULL streambuf  
**Section:** 27.6.2.1 **Template class basic\_ostream** [lib.ostream]  
**Status:** active  
**Description:**

Functions in basic\_ostream that call members of rdbuf() need to check for a NULL streambuf before calling the function. There are some functions that make sure rdbuf() is not a NULL pointer before calling any functions on the buffer, but some functions don't check for the NULL pointer. This needs to be consistent.

### **Discussion:**

P.J. Plauger wrote: "Any attempt to store a null stream buffer pointer causes badbit to be set in the stored status. Hence, no input or output is ever attempted, using such a pointer, by formatted functions."

**Possible Resolution:**

As pointed out by P.J. Plauger we should add a footnote to explain why there is no need to check for a NULL streambuf.

We should also add in section 27.4.4.2 **Member functions** [**lib.basic.ios.members**] the following to the description of `basic_streambuf<charT,traits>*` `rdbuf(basic_streambuf<charT,traits>* sb);` :

**Postcondition:** `sb == rdbuf()` and if `sb` is a NULL pointer `rdstate() == badbit`.

Note: This issue has to be discussed with issue 27-404.

The pre-Stockholm Iostreams WG recommends the issue to be treated as editorial. It needs to be pointed out where the check is not necessary. In places where check is required, recommend add the check. Jerry Schwarz will provide text for the proposal.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-505  
**Title:** incorrect conversion specifier for operator<<(unsigned long)  
**Section:** 27.6.2.4.2 **basic\_ostream::operator<<** [**lib.ostream.inserters**]  
**Status:** active  
**Description:**

`basic_ostream<charT,traits>& operator<<(unsigned long n);`  
 Effects: Converts the unsigned long integer `n` with the integral conversion specified preceded by `l`. Should this be "... preceded by `ul`."?

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends closing the issue, since the rewrite of this section fixed the problem.

**Requestor:** Public Comment

---

**Issue Number:** 27-506  
**Title:** wrong default behavior for padding  
**Section:** 27.6.2.4.1 Common requirements Table 13 Fill padding  
**[lib.ostream.formatted.reqmts]**  
**Status:** active  
**Description:**

27.6.2.4.1 Table13 Fill padding changes the long-standing default behavior for padding output field. It has always been true that setting none of left, right, and internal called for left padding (pad after text). Now it calls for right padding (pad before text). Since this is the initial state of all ios objects, many simple C++ programs will change behavior.

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends closing the issue, since the rewrite and moving discussion of fill padding to chapter 22 (Localization) fixed the problem.

**Requestor:** P.J. Plauger (plauger!pjp@uunet.uu.net)

---

## basic\_istream/basic\_ostream issues

**Issue Number:** 27-601  
**Title:** istream::operator>>(ios\_base&), ostream::operator<<(ios\_base&)  
**Section:** 27.6.1.2.2 **basic\_istream::operator>>** [**lib.istream::extractors**],  
27.6.2.4.2 **basic\_ostream::operator<<** [**lib.ostream.inserters**]  
**Status:** active  
**Description:**

The ios\_base manipulators 27.4.5.1 [**lib.std.ios.manip**] will not work as written. They won't work because there is no conversion from ios\_base to basic\_ios.

They are currently declared as:  
ios\_base& boolalpha(ios\_base&);

I propose adding a new inserter/extractor for istream and ostream that does insertion/extraction for ios\_base.

### Possible Resolution:

John wrote:

“Add to basic\_istream:

```
basic_istream<charT, traits>& operator>>(ios_base& (*pf)(ios_base&));
```

**Effects:** Calls (\*pf)(\*this)

**Returns:** \*this.

Add to basic\_ostream:

```
basic_ostream<charT, traits>& operator<<(ios_base& (*pf)(ios_base&));
```

**Effects:** Calls (\*pf)(\*this)

**Returns:** \*this.

Also, several footnotes will need to be changed.”

We need to change footnote 9 in 27.4.5.3 **basefield manipulators** [**lib.basefield.manip**] to:

“The function signature dec(ios\_base& str) can be called by the function signature basic\_ostream<charT,traits>& basic\_ostream<charT,traits>::operator << ( ios\_base& (\*) (ios\_base& ) ) to permit expressions of the form cout << dec to change the format flags stored in cout.”

The pre-Stockholm Iostreams WG recommends accepting resolution in issues list. Philip Le Mouël will provide text for the proposal.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-602

**Title:** positional typedefs in istream/ostream derived classes  
**Section:** 27  
**Status:** active  
**Description:**

Remove the positional typedefs from the following classes. The positional typedefs are:

```
typedef traits::pos_type pos_type;  
typedef traits::off_type off_type;
```

They are not used in the following classes:

```
basic_istream  
basic_ostream  
basic_ifstream  
basic_ofstream
```

**Possible Resolution:**

John wrote:

“Remove them. They are still inherited from the base classes.”

The pre-Stockholm Iostreams WG recommends closing the issue with no change to the WP. It does not harm to leave in the typedefs, and probably they are required by language rules anyway.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-603  
**Title:** istream::read, ostream::write  
**Section:** 27.6.1.3 **Unformatted input functions [lib.istream.unformatted]**  
27.6.2.5 **Unformatted output functions [lib.ostream.unformatted]**  
**Status:** active  
**Description:**

```
basic_istream<charT,traits>& basic_istream<charT,traits>::read(char_type *,streamsize);  
basic_ostream<charT,traits>& basic_ostream<charT,traits>::write(const char_type *,streamsize);
```

These functions are typically used for binary data.

**Possible Resolution:**

John wrote:

“These functions should take a void \* instead of char\_type \*. If these functions are changed, then perhaps we should add another function that replaces this behavior. basic\_istream currently has a get function, which behaves like the read and write functions. It would make sense to add a corresponding put function in basic\_ostream that parallels the behavior of get.”

The pre-Stockholm Iostreams WG recommends closing the issue with no change to the WP. If *read/write* take void\*, what does it mean to write bytes to a wide stream? Here it is clear that you always write sequences of charT to a stream.



**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-604  
**Title:** Opening an istream without ios::in set? or an ostream without ios::out set?  
**Section:** 27.6.1.1 **Template class basic\_istream** [lib.input.streams],  
27.6.2.1 **Template class basic\_ostream** [lib.output.streams]  
**Status:** active  
**Description:**

Benedikt asks,

“Why can I open an istream without ios::in being set or an ostream without ios::out? I mean, I just did that by mistake with an ofstream and searched for quite a while to find out, why there were no actual writes to the newly created file.

“Or, even worse, why can I open an istream with ios::out (and no ios::in) being set and vice versa?”

“Shouldn't the iostreams check whether the given mode flags make any sense, and maybe even add ios::in if you missed to set this in an istream, or ios::out if you used an ostream?”

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends that ctors/open for iXstream, oXstream always assumes the “in” or “out” bit. Ctors/open for Xstream (bidirectional) do not assume either “in” or “out” bits. Steve Clamage will provide text for the proposal.

**Requestor:** Benedikt Erik Heinen (beh@tequila.oche.de)

---

**Issue Number:** 27-605  
**Title:** get/put type functions should be able to use iterators.  
**Section:** 27.6.1.3 **Unformatted input functions** [lib.istream.unformatted]  
27.6.2.5 **Unformatted output functions** [lib.ostream.unformatted]  
**Status:** active  
**Description:**

Several functions in istream and ostream take a pointer and a length and optionally a delimiter. It would be nice to add overloaded functions that take either InputIterators, or OutputIterators. These new functions would look like:

For basic\_istream:

```
template<class OutputIterator>
istream& get(OutputIterator begin, OutputIterator end, char_type delim);
```

The *begin* and *end* iterators define where the characters will be written. Characters will be read from the sequence until the *end* iterator is reached, or the next character is *delim*.

For basic\_ostream:

```
template<class InputIterator>
ostream& write(InputIterator begin, InputIterator end);
```

The *begin* and *end* iterators define the sequence of characters to be written.

These functions would be added to the current implementation. The current set of functions should not be removed. They are very commonly used. There are several functions which are candidates for these *begin* and *end* iterators. These functions are:

For `basic_istream`:

```
istream& get(char_type *, streamsize, char_type);
istream& getline(char_type *, streamsize, char_type);
istream& read(char_type *, streamsize);
```

For `basic_ostream`:

```
ostream& put(char_type *, streamsize);
ostream& write(void *, streamsize);
```

#### **Possible Resolution:**

The pre-Stockholm Iostreams WG recommends closing the issue with no change to the WP. This is a proposed extension.

**Requestor:** Nathan Myers (ncm@cantrip.org)

---

**Issue Number:** 27-606  
**Title:** seekg and seekp should have their first parameter passed by value.  
**Section:** 27.6.1.3 **Unformatted input functions [lib.istream.unformatted]**  
27.6.2.3 **basic\_ostream prefix and suffix functions [lib.ostream.prefix]**  
**Status:** active  
**Description:**

The following functions should have their first parameter passed by value as described in 27.6.1.1 **Template class basic\_istream [lib.istream]** and 27.6.2.1 **Template class basic\_ostream [lib.ostream]**.

```
basic_istream<charT,traits>& seekg(off_type& off, ios_base::seekdir dir);
basic_ostream<charT,traits>& seekp(pos_type& pos);
basic_ostream<charT,traits>& seekp(off_type& off, ios_base::seekdir dir);
```

The seekp functions should also be moved in section 27.6.2.5 **Unformatted output functions [lib.ostream.unformatted]**.

#### **Possible Resolution:**

Change them to:

```
basic_istream<charT,traits>& seekg(off_type off, ios_base::seekdir dir);
basic_ostream<charT,traits>& seekp(pos_type pos);
basic_ostream<charT,traits>& seekp(off_type off, ios_base::seekdir dir);
```

The pre-Stockholm Iostreams WG recommends treating the issue as editorial. Steve Clamage will provide text to make the declaration and definition agree to pass parameter by value.

**Requestor:** Philippe Le Mouël (philippe@roguewave.com)

---

**Issue Number:** 27-607

**Title:** locale getnum needed for void\* extractor (Box)

**Section:** 27.6.1.2.3 `basic_istream::operator>>` [`lib.istream::extractors`]

**Status:** active

**Description:**

The description of void\* extractor needs work. Maybe there should be a locale getnum for it. (Using %p conversion specifier).

**Possible Resolution:**

**Requestor:** pre-Stockholm Iostreams WG

---

## Standard manipulators issues

**Issue Number:** 27-651  
**Title:** setfill description is wrong  
**Section:** 27.6.3 Standard manipulators [**lib.std.manip**]  
**Status:** active  
**Description:**

P.J. Plauger wrote: “Setfill description is nonsense, since a fill character is now a charT, which cannot necessarily be represented as type int. Nor can it be applied to ios\_base, since the fill character now inhabits basic\_ios.”

**Possible Resolution:**

The pre-Stockholm Iostreams WG deferred the issue to Stockholm.  
The problem is which parameter type(s) should the *setfill* manipulator accept. Certainly charT, but what about char or int? Problem is lack of a correct implicit “widen” from char to charT, because the conversion requires knowledge of the locale.

**Requestor:** P.J. Plauger (plauger!pjp@uunet.uu.net)  
Philippe Le Mouël (philippe@roguewave.com)

---

**Issue Number:** 27-652  
**Title:** smanip is not a single type  
**Section:** 27.6.3 Standard manipulators [**lib.std.manip**]  
**Status:** active  
**Description:**

P.J. Plauger wrote: “Description of manipulators strongly suggests that smanip is a single type. It was supposed to make clear that each manipulator can return a different type, as needed. (And more than one type is certainly needed here.)”

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends treating the issue as editorial. Jerry Schwarz will provide wording, along the lines of:

**Returns:** A value of some class lmanip such that `out << resetiosflags(m)` has the effect of `out.setf(0,m)`.

**Requestor:** P.J. Plauger (plauger!pjp@uunet.uu.net)  
Philippe Le Mouël (philippe@roguewave.com)

---

## string stream issues

**Issue Number:** 27-701  
**Title:** basic\_stringbuf::str() needs to clarify return value on else clause  
**Section:** 27.7.1.2 Member functions [lib.stringbuf.members]  
**Status:** active  
**Description:**

“Table 15 in [lib.stringbuf.members] describes the return values of basic\_stringbuf::str(). What does the "otherwise" mean?. Does it mean neither ios\_base::in nor ios\_base::out is set? What is the return value supposed to be if \_both\_ bits are set?”

### Possible Resolution:

My understanding is that if both ios\_base::in and ios\_base::out are set, you should return basic\_string<char\_type>(eback(),egptr()-eback()). I propose to change the **Returns:** clause to clarify this fact.

**Returns:** The return values of this function are indicated in Table 15 and the test that determine these values are carried out in the order shown in Table 15.

The pre-Stockholm Iostreams WG recommends defining a consistent model for stringstreams. Should stringbufs be more like filebufs or more like strstreams? Probably more like files. The String stream section needs a full review and possibly some rewriting to fit the model. Philippe Le Mouël will provide text for the proposal.

**Requestor:** Angelika Langer (langer@roguewave.com)  
Bernd Eggink (admin@rrz.uni-hamburg.de)

---

**Issue Number:** 27-702  
**Title:** string streams need allocator and string\_char\_traits parameters  
**Section:** 27.7.1 Template class basic\_stringbuf [lib\_stringbuf]  
**Status:** active  
**Description:**

The string streams are currently templated on the character type (charT) and the traits type (ios\_traits). String template parameters need to be added.

### Possible Resolution:

The Santa Cruz meeting, fixes the problem, by accepting doc: 96-0036R1=N0854R1 (Unification of Traits Revision1). But we are still left with the problem of taking or returning string arguments using an other allocator than the default one. See *basic\_stringbuf*, *basic\_istream*, *basic\_ostringstream* and *basic\_stringstream* constructors and *str* functions.

The pre-Stockholm Iostreams WG recommends defining a consistent model for stringstreams. Should stringbufs be more like filebufs or more like strstreams? Probably more like files. The String stream section needs a full review and possibly some rewriting to fit the model. Philippe Le Mouël will provide text for the proposal.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-703  
**Title:** stringbuf postconditions  
**Section:** 27.7.1.2 Member functions [lib.stringbuf.members]  
**Status:** active  
**Description:**

basic\_stringbuf::str(basic\_string s) Postconditions requires that str() == s. This is true only if which had in set at construction time. Condition should be restated.

**Possible Resolution:**

I think the real problem is in “Table 16 - str get/set areas”. Its second line says:

( which & ios\_base::out ) != 0 setp(str(),str(),str()+str.size())

First, the function setp takes only two parameters. Furthermore it should say:

( which & ios\_base::out ) != 0 setp(str(),str()+s.size())  
then if: ( which & (ios\_base::app | ios\_base::ate) ) != 0 pbump(s.size())

Then the postcondition requiring that str() == s in the function void str( const basic\_string<char\_type>& s) will be valid if “in” or “out” and “app” or “ate” are set at construction time.

Table 16 should be changed to:

**Table 16—str get/set areas**

Condition	Setting
(which & ios_base::in)!=0	setg(str(), str(), str()+s.size())
(which & ios_base::out)!=0	setp(str(),str()+s.size())
(which & (ios_base::app   ios_base::ate) )!=0	pbump(s.size())

The postcondition should be changed to:

**Postcondition:** if ios\_base::in, or ios\_base::out and ios\_base::app or ios\_base::ate are set at construction time, then str()==s. Otherwise str() == basic\_string<char\_type>(). If s.size() >0, set the get and/or put pointers as indicated in Table 16.

The pre-Stockholm Iostreams WG recommends defining a consistent model for stringstreams. Should stringbufs be more like filebufs or more like stringstream? Probably more like files. The String stream section needs a full review and possibly some rewriting to fit the model. Philippe Le Mouél will provide text for the proposal.

**Requestor:** Public Comment

---

**Issue Number:** 27-704  
**Title:** stringbuf::stringbuf constructor  
**Section:** 27.7.1.1 basic\_stringbuf constructors [lib.stringbuf.cons]  
**Status:** active  
**Description:**

basic\_stringbuf::basic\_stringbuf(basic\_string str, openmode which) Postconditions requires that str() == str. This is true only if which has in set. Condition should be restated.

**Possible Resolution:**

This is the same problem described in issue 27-703.

The real problem is in “Table 14 - str get/set areas”. The second line says:

```
( which & ios_base::out) != 0 setp(str(),str(),str()+str.size())
```

First, the function setp takes only two parameters. Furthermore it should say:

```
( which & ios_base::out )!= 0 setp(str(),str()+str.size())
then if: ( which & (ios_base::app | ios_base::ate) )!=0 pbump(str.size())
```

Then the postcondition requiring that str() == str in the function basic\_stringbuf::basic\_stringbuf(basic\_string str, openmode which) will be valid if “in” or “out” and “app” or “ate” are set.

Table 14 should be changed to:

**Table 14—str get/set areas**

Condition	Setting
(which & ios_base::in)!=0	setg(str(), str(), str()+str.size())
(which & ios_base::out)!=0	setp(str(),str()+str.size())
(which & (ios_base::app   ios_base::ate) )!=0	pbump(str.size())

The postcondition should be changed to:

**Postcondition:** if ios\_base::in, or ios\_base::out and ios\_base::app or ios\_base::ate are set at construction time, then str()==str. Otherwise str() == basic\_string<char\_type>(). If str.size() >0, set the get and/or put pointers as indicated in Table 14.

The pre-Stockholm Iostreams WG recommends defining a consistent model for stringstreams. Should stringbufs be more like filebufs or more like strmstreams? Probably more like files. The String stream section needs a full review and possibly some rewriting to fit the model. Philippe Le Mouél will provide text for the proposal.

<b>Requestor:</b>	Public Comment
<b>Issue Number:</b>	27-705
<b>Title:</b>	Incorrect calls to setg and setp in Table 14
<b>Section:</b>	27.7.1.1 <b>basic_stringbuf constructors</b> [lib.stringbuf.cons]
<b>Status:</b>	<b>active</b>
<b>Description:</b>	

Table 14 describes calls to setg and setp with string arguments, for which no signature exists. Needs to be recast.

**Possible Resolution:**

Possible Resolution of issue 27-704 solves this problem.

The pre-Stockholm Iostreams WG recommends defining a consistent model for stringstreams. Should stringbufs be more like filebufs or more like strstreams? Probably more like files. The String stream section needs a full review and possibly some rewriting to fit the model. Philippe Le Mouél will provide text for the proposal.

**Requestor:** Public Comment

---

**Issue Number:** 27-706  
**Title:** Incorrect calls to setg and setp in table 16  
**Section:** 27.7.1.2 **Member functions** [**lib.stringbuf.members**]  
**Status:** active  
**Description:**

Table 16 describes calls to setg and setp with string arguments, for which no signature exists. Needs to be recast.

**Possible Resolution:**

Possible Resolution of issue 27-703 solves this problem

The pre-Stockholm Iostreams WG recommends defining a consistent model for stringstreams. Should stringbufs be more like filebufs or more like strstreams? Probably more like files. The String stream section needs a full review and possibly some rewriting to fit the model. Philippe Le Mouél will provide text for the proposal.

**Requestor:** Public Comment

---

**Issue Number:** 27-707  
**Title:** setbuf function is missing  
**Section:** 27.7.1 **Template class basic\_stringbuf** [**lib.stringbuf**]  
**Status:** active  
**Description:**

Steve Clamage wrote: “Section 27.7.1.3 should have a basic\_stringbuf override of the base class setbuf() function, but it is missing.”

**Possible Resolution:**

Add the following description in 27.7.1 **Template class basic\_stringbuf** [**lib.stringbuf**] and 27.7.1.3 **Overridden virtual functions** [**lib.stringbuf.virtuals**]:

```
basic_streambuf<charT,traits>* setbuf( char_type* s, int n);
```

**Effects:** If ( *mode* & *ios\_base::out* ) is true, proceed as follows:

If *s* is not a null pointer, and *n* > *pptr()* - *pbase()*, replace the current buffer (copy its contents and deallocate it) by the buffer of size *n* pointed at by *s*.

In the case where *s* is a null pointer, and *n* > *pptr()* - *pbase()* resize the current buffer to size *n*.

If the function fails, it returns a null pointer.

**Returns:** (`basic_streambuf<charT,traits>*`)(this)



I am not qualified enough to decide if the return type should be changed to `basic_stringbuf<charT,traits>*` as proposed by Steve Clamage in issue 27-809. I tried it with several compilers, and the results were just error messages. Basically, the compilers were complaining about the fact that the base class virtual function and the overridden virtual function should have the same return type.

The pre-Stockholm Iostreams WG recommends the behavior of the `setbuf` function to be implementation-defined, except that `setbuf(0,0)` has no effect. The return type in the WP is coreect. Steve Clamage will provide text for the proposal.

**Requestor:** Steve Clamage (stephen.clamage@eng.sun.com)

## file stream issues

**Issue Number:** 27-801  
**Title:** filebuf::underflow example  
**Section:** 27.8.1.4 **Overridden virtual functions** [lib.filebuf.virtuals]  
**Status:** active  
**Description:**

The “as if” example for basic\_filebuf::underflow has several “typos”. It should say:

```
char from_buf[FSIZE];
char* from_end;
char to_buf[TSIZE];
char* to_end;
typename traits::state_type st;

codecvt_base::result r =
    getloc().template use<codecvt<char, charT,
    typename traits::state_type>>().convert
    (st, from_buf, from_buf+FSIZE, from_end,
    to_buf, to_buf+TSIZE, to_end);
```

**Possible Resolution:**

We should correct the example as follows, and not as described above:

```
char from_buf[FSIZE];
char* from_end;
charT to_buf[TSIZE];
charT* to_end;
typename traits::state_type st;

codecvt_base::result r=
    use_facet<codecvt<char,charT,typename traits::state_type>>(getloc()).
    convert(st,from_buf,from_buf+FSIZE,from_end,to_buf,to_buf+TSIZE,to_end);
```

Should be treated with issue 27-812.

The pre-Stockholm Iostreams WG recommends to treat the issue as editorial. The modifications have already been incorporated to the WP.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-802  
**Title:** filebuf::is\_open is a bit confusing  
**Section:** 27.8.1.3 **Member functions** [lib.filebuf.members]  
**Status:** active  
**Description:**

It says, “**Returns:** true if the associated file is available and open.” What is the meaning of available? This seems a bit confusing.

**Possible Resolution:**

Change the **Returns:** statement to:

**Returns:** true after a successful call to the member function open, and before a successful call to member function close, otherwise false.

The pre-Stockholm Iostreams WG recommends accepting the above resolution. Jerry Schwarz will provide text for the proposal.

**Requestor:** John Hinke (hinke@roguewave.com),  
Bob Kline (bkline@cortex.nlm.nih.gov)

---

**Issue Number:** 27-803  
**Title:** ofstream constructor missing trunc as openmode  
**Section:** 27.8.1.9 **basic\_ofstream constructors** [**lib ofstream.cons**]  
**Status:** active  
**Description:**

basic\_ofstream::basic\_ofstream(const char \*s, openmode mode = out) has wrong default second argument. It should be `out | trunc', the same as for basic\_ofstream::open (in the definition at least).

**Possible Resolution:**

In 27.8.1.9 **basic\_ofstream constructors** [**lib ofstream.cons**] change:

explicit basic\_ofstream(const char\* s, openmode mode = out);

to:

explicit basic\_ofstream(const char\* s, ios\_base::openmode mode = ios\_base::out | ios\_base::trunc);

In 27.8.1.10 **Member functions** [**lib ofstream.members**] change:

void open(const char\* s, openmode mode = out);

to:

void open(const char\* s, ios\_base::openmode mode = ios\_base::out | ios\_base::trunc);

The pre-Stockholm Iostreams WG recommends fixing the words and Table 14 to show that “out” by itself is equivalent to “out | trunc”. Then “out | trunc” can be changed to “out” everywhere as editorial. All possible combinations of flags should be considered and given a meaning or declared invalid or undefined. Steve Clamage will provide text for the proposal.

**Requestor:** Public Comment

---

**Issue Number:** 27-804  
**Title:** ofstream::open missing trunc in openmode  
**Section:** 27.8.1.10 **Member functions** [**lib ofstream.members**]

**Status:** active

**Description:**

basic\_ofstream::open(const char \*s, openmode mode = out) has wrong default second argument. It should be `out | trunc`, the same as for basic\_ofstream::open in the definition.

**Possible Resolution:**

See issue 27-803.

**Requestor:** Public Comment

---

**Issue Number:** 27-805

**Title:** filebuf::imbue semantics

**Section:** 27.8.1.4 **Overridden virtual functions [lib.filebuf.virtuals]**

**Status:** active

**Description:**

basic\_filebuf::imbue has silly semantics. Whether or not sync() succeeds has little bearing on whether you can safely change the working codecvt facet. The most sensible thing is to establish this facet at construction. (Then pubimbue and imbue can be scrubbed completely.) Next best is while is\_open() is false. (Then imbue can be scrubbed, since it has nothing to do.) Next best is to permit any imbue that doesn't change the facet or is at beginning of file. Next best is to permit change of facet any time provided either the current or new facet does not mandate state-dependent conversions. (See comments under seekoff.)

**Possible Resolution:**

In my current version of the WP, I do not have any description for the virtual filebuf imbue function. See issue 27-814.

The pre-Stockholm Iostreams WG deferred the issue to Stockholm.

**Requestor:** Public Comment

---

**Issue Number:** 27-806

**Title:** filebuf::seekoff **Effects:** clause needs work

**Section:** 27.8.1.4 **Overridden virtual functions [lib.filebuf.virtuals]**

**Status:** active

**Description:**

basic\_filebuf::seekoff Effects is an interesting exercise in creative writing. It should simply state that if the stream is opened as a text file or has state-dependent conversions, the only permissible seeks are with zero offset relative to the beginning or current position of the file. (How to determine that predicate is another matter -- should state for codecvt that even a request to convert zero characters will return noconv.) Otherwise, behavior is largely the same as for basic\_stringstream, from whence the words should be cribbed. The problem of saving the stream state in a traits::pos\_type object remains unsolved. The primitives described for ios\_traits are inadequate.

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends accepting in principle the recommendations in the public comment. Eight bit streams behave like stdio FILEs. For wide streams, seeks are allowed provided the external file meets certain criteria to be worked out by Tom Plum, who will provide text for the proposal.

**Requestor:** Public Comment

---

**Issue Number:** 27-807  
**Title:** filebuf::underflow performance questions  
**Section:** 27.8.1.4 **Overridden virtual functions [lib.filebuf.virtuals]**  
**Status:** active  
**Description:**

basic\_filebuf::underflow is defined unequivocally as the function that calls codecvt, but there are performance advantages to having this conversion actually performed in uflow. If the specification cannot be broadened sufficiently to allow either function to do the translation, then uflow loses its last rationale for being added in the first place. Either the extra latitude should be granted implementors or uflow should be removed from basic\_streambuf and all its derivatives.

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends that *filebuf* overrides both *uflow* and *underflow*, instead of just *underflow*. Semantic are left as-is. Jerry Schwarz will provide text for the proposal.

**Requestor:** Public Comment

---

**Issue Number:** 27-808  
**Title:** Editorial fixes in wording for fstreams  
**Section:** 27.8.1 **File streams [lib.fstreams]**  
**Status:** active  
**Description:**

27.8.1 [lib.fstreams], paragraph 2: "... the type name FILE is a synonym for the type FILE." This seems like an odd sort of synonym, doesn't it? Also, the last sentence of this subsection, "Because of necessity of the conversion between the external source/sink streams and wide character sequences." is incomplete.

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends to treat the issue as editorial. Make clear that FILE is a typedef in C, but not necessarily in C++. The change as already been incorporated.

**Requestor:** Public Comment

---

**Issue Number:** 27-809  
**Title:** Description of function setbuf is missing  
**Section:** 27.8.1.4 **Overridden virtual functions [lib.filebuf.virtuals]**  
**Status:** active  
**Description:**

Steve Clamage wrote: "basic\_filebuf version of setbuf( ) needs a description, and the return type shown in the draft is basic\_streambuf\*, which is probably wrong. It was correct before covariant return types were added to the draft. Now it should probably return basic\_filebuf\*."

### Possible Resolution:

Add the following description in 27.8.1.4 **Overridden virtual functions** [**lib.filebuf.virtuals**]:

```
basic_streambuf<charT,traits>* setbuf( char_type* s, streamsize n);
```

**Effects:** If *s* is not a null pointer, flush the buffer by calling *overflow( traits::eof() )* and if the return value is not *traits::eof()*, deallocate the current buffer and replace it by the buffer of size *n* pointed at by *s*.

In the case where *s* is a null pointer, resize the current buffer to size *n* (this can result in flushing the buffer).

If the function fails, it returns a null pointer.

**Returns:** (`basic_streambuf<charT,traits>*`)(this)

I am not qualified enough to decide if the return type should be changed or not as proposed by Steve Clamage. I tried it with several compilers, and the results were just error messages. Basically, the compilers were complaining about the fact that the base class virtual function and the overridden virtual function should have the same return type.

The pre-Stockholm Iostreams WG recommends the behavior of the *setbuf* function to be implementation-defined, except that *setbuf(0,0)* if called before any I/O occurs on the stream, makes the stream unbuffered. The return type in the WP is correct. Steve Clamage will provide text for the proposal.

**Requestor:** Steve Clamage (stephen.clamage@eng.sun.com)

---

**Issue Number:** 27-810

**Title:** Openmode notation is not consistent in `basic_ifstream` and `basic_ofstream`

**Section:** 27.8.1.5 **Template class `basic_ifstream`** [**lib.ifstream**]  
27.8.1.8 **Template class `basic_ofstream`** [**lib.ofstream**]

**Status:** active

**Description:**

`basic_ifstream`, `basic_ofstream` *constructors* and member functions *open* describe the type `ios_base::openmode` as *openmode* and its values as *in* and *out* rather than `ios_base::in` and `ios_base::out` as everywhere else in the library.

### Possible Resolution:

In 27.8.1.5 **Template class `basic_ifstream`** [**lib.ifstream**] , 27.8.1.6 **`basic_ifstream` constructors** [**lib.ifstream.cons**] and 27.8.1.7 **member functions** [**lib.ifstream.members**] change the following functions:

```
explicit basic_ifstream(const char* s, openmode mode = in);
```

to:

```
explicit basic_ifstream(const char* s, ios_base::openmode mode = ios_base::in);
```

```
void open(const char* s, openmode mode = in);
```

to:

```
void open(const char* s, ios_base::openmode mode = ios_base::in);
```

In 27.8.1.8 **Template class `basic_ofstream`** [**lib.ofstream**] , 27.8.1.9 **`basic_ofstream` constructors** [**lib.ofstream.cons**] and 27.8.1.10 **member functions** [**lib.ofstream.members**] change the following functions:

```
explicit basic_ofstream(const char* s, openmode mode = out | trunc);
to:
explicit basic_ofstream(const char* s, ios_base::openmode mode = ios_base::out | ios_base::trunc);

void open(const char* s, openmode mode = out | trunc);
to:
void open(const char* s, ios_base::openmode mode = ios_base::out | ios_base::trunc);
```

The pre-Stockholm Iostreams WG recommends to treat as editorial. The “trunc” issue is handled by 27-803,804.

**Requestor:** Philippe Le Mouël (philippe@roguewave.com)

---

**Issue Number:** 27-811  
**Title:** Description of function sync is missing  
**Section:** 27.8.1.4 **Overridden virtual functions [lib.filebuf.virtuals]**  
**Status:** active  
**Description:**

Description of the overridden sync( ) function in class basic\_filebuf is missing.

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends adapting wording from C standard for output. The intent is clear, for in/out files, flush output then really seek to current file position. For input files, implementation-defined. Tom Plum will provide text for the proposal.

**Requestor:** Philippe Le Mouël (philippe@roguewave.com)

---

**Issue Number:** 27-812  
**Title:** filebuf::overflow example is incorrect  
**Section:** 27.8.1.4 **Overridden virtual functions [lib.filebuf.virtuals]**  
**Status:** active  
**Description:**

The “as if” example for basic\_filebuf::overflow has several “typos”. It should say:

```
charT* b = pbase();
charT* p = pptr();
charT* end;
char buf[BSIZE];
char* ebuf;
typename traits::state_type st;

codecvt_base::result r=
    use_facet<codecvt<charT,char,typename traits::state_type>>(getloc()).
    convert(st,b,p,end,buf,buf+BSIZE,ebuf);
```

**Possible Resolution:**

Should be treated with issue 27-801.

The pre-Stockholm Iostreams WG recommends to treat the issue as editorial. The modifications have already been incorporated to the WP.

**Requestor:** Public Comment

---

**Issue Number:** 27-813  
**Title:** basic\_filebuf::overflow does not specifies its return value on success  
**Section:** 27.8.1.4 **Overridden virtual functions [lib.filebuf.virtuals]**  
**Status:** active  
**Description:**

The function basic\_filebuf::overflow does not specifies its return value on success.

**Possible Resolution:**

The **Returns:** clause should be changed to:

**Returns:** If *traits::eq\_int\_type(c, traits::eof())* returns false, the function signals success by returning *c*. Otherwise If *traits::eq\_int\_type(c, traits::eof())* returns true, the function signals success by returning *traits::not\_eof( c )*. The function returns *traits::eof()* to indicate failure.

The pre-Stockholm Iostreams WG recommends to return *not\_eof( c )* on success. Jerry Schwarz will provide text for the proposal.

**Requestor:** Public Comment

---

**Issue Number:** 27-814  
**Title:** basic\_filebuf::imbue has no description  
**Section:** 27.8.1.4 **Overridden virtual functions [lib.filebuf.virtuals]**  
**Status:** active  
**Description:**

The virtual function basic\_filebuf::imbue has no description.

**Possible Resolution:**

The pre-Stockholm Iostreams WG deferred the issue to Stockholm.

**Requestor:** Philippe Le Mouël (philippe@roguewave.com)

---

**Issue Number:** 27-815  
**Title:** description of function seekpos is missing  
**Section:** 27.8.1.4 **Overridden virtual functions [lib.filebuf.virtuals]**  
**Status:** active  
**Description:**

basic\_filebuf::seekpos has no sementics. Needs to be supplied.

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends that the function sets the file position to value of its argument, which must have come from seekoff(0). Philippe Le Mouël will provide text for the proposal.



**Requestor:** Public Comment

---

**Issue Number:** 27-816

**Title:** (i)(o)fstream open functions should not use `is_open`

**Section:** 27.8.1.7 **Member functions** [**lib.ifstream.members**]

27.8.1.10 **Member functions** [**lib.ofstream.members**]

**Status:** active

**Description:**

(i)(o)fstream *open* functions should not use `is_open` to determine if the operation fails (and as a result setting *failbit*). The problem arises if you do not close the (i)(o)fstream and then try to open another file with it. In this case the *filebuf open* function will fail, but `is_open` will still return true.

**Possible Resolution:**

The Effects: clause for *open* should be changed to:

**Effects:** Calls `Rdbuf()->open(s,mode)`. If that function returns a null pointer, calls `Setstate(failbit)` (which may throw `ios_base::failure`).

Note: What about (i)(o)fstream constructors ? *failbit* should be set in the case they fail to open the file.

The pre-Stockholm Iostreams WG recommends accepting the above resolution. Philippe Le Mouël will provide text for the proposal.

**Requestor:** Philippe Le Mouël (philippe@roguewave.com)

---

## Miscellaneous issues

**Issue Number:** 27-901  
**Title:** input/output of unsigned charT  
**Section:** 27  
**Status:** active  
**Description:**

NOTE: istream here means basic\_istream.  
ostream here means basic\_ostream.

This issue details all of the issues with inserting or extracting characters.

Currently, IOStreams does not allow the insertion/extraction of unsigned charT or signed charT. There are two types of functions that could insert or extract these character types: formatted IO, and unformatted IO. Formatted IO use overloaded operators. Example:

```
istream& istream::operator>>(charT&);  
ostream& ostream::operator<<(charT);
```

Examples of unformatted IO are:

```
istream& istream::get(charT *, streamsize, charT);  
int_type ostream::put(charT);
```

This does not allow us to overload on unsigned charT. We can make the formatted operators global, and then overload (“specialize”) on char, and wchar\_t, but that doesn’t solve the unformatted problem.

There is also a problem of inserting or extracting wide-characters from a skinny stream or skinny characters from a wide-stream:

```
char c;  
wchar_t wc;  
  
cout << wc;  
wcout << c;
```

### Possible Resolution:

The pre-Stockholm Iostreams WG deferred the issue to Stockholm.

See papers: X3J16/96-0100 WG21/N0918 and X3J16/96-0101 WG21/N0919

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-904  
**Title:** iosfwd declarations: incomplete  
**Section:** 27.2 **Forward declarations [lib.iostream.forward]**  
**Status:** active  
**Description:**

The list of forward declarations is incomplete. Should it contain all of the forward declarations available? Forward declarations for template classes `basic_ios`, `basic_istream`, and `basic_ostream` should have two class parameters, not one. It is equally dicey to define `ios`, `istream`, etc. by writing just one parameter for the defining classes. All should have the second parameter supplied, which suggests the need for a forward reference to template class `ios_char_traits` as well, or at least the two usual specializations of that class.

**Possible Resolution:**

Replace “**Header <iosfwd> synopsis**” by:

```
namespace std {
    template<class charT> struct char_traits;
    template<class charT, class traits = char_traits<charT> > class basic_ios;
    template<class charT, class traits = char_traits<charT> > class basic_streambuf;
    template<class charT, class traits = char_traits<charT> > class basic_istream;
    template<class charT, class traits = char_traits<charT> > class basic_ostream;
    template<class charT, class traits = char_traits<charT> > class basic_iostream;
    template<class charT, class traits = char_traits<charT> > class basic_stringbuf;
    template<class charT, class traits = char_traits<charT> > class basic_istreamstream;
    template<class charT, class traits = char_traits<charT> > class basic_ostringstream;
    template<class charT, class traits = char_traits<charT> > class basic_stringstream;
    template<class charT, class traits = char_traits<charT> > class basic_filebuf;
    template<class charT, class traits = char_traits<charT> > class basic_ifstream;
    template<class charT, class traits = char_traits<charT> > class basic_ofstream;
    template<class charT, class traits = char_traits<charT> > class basic_fstream;
    template<class charT, class traits= char_traits<charT> > class ostreambuf_iterator;
    template<class charT, class traits= char_traits<charT> > class istreambuf_iterator;

    typedef basic_ios<char> ios;
    typedef basic_streambuf<char> streambuf;
    typedef basic_istream<char> istream;
    typedef basic_ostream<char> ostream;
    typedef basic_iostream<char> iostream;
    typedef basic_stringbuf<char> stringbuf;
    typedef basic_istreamstream<char> istringstream;
    typedef basic_ostringstream<char> ostringstream;
    typedef basic_stringstream<char> stringstream;
    typedef basic_filebuf<char> filebuf;
    typedef basic_ifstream<char> ifstream;
    typedef basic_ofstream<char> ofstream;
    typedef basic_fstream<char> fstream;
    typedef basic_ios<wchar_t> wios;
    typedef basic_streambuf<wchar_t> wstreambuf;
    typedef basic_istream<wchar_t> wistream;
    typedef basic_ostream<wchar_t> wostream;
    typedef basic_iostream<wchar_t> wiostream;
    typedef basic_stringbuf<wchar_t> wstringbuf;
    typedef basic_istreamstream<wchar_t> wistringstream;
    typedef basic_ostringstream<wchar_t> wostringstream;
    typedef basic_stringstream<wchar_t> wstringstream;
    typedef basic_filebuf<wchar_t> wfilebuf;
```

```

        typedef basic_ifstream<wchar_t>          wifstream;
        typedef basic_ofstream<wchar_t>         wofstream;
        typedef basic_fstream<wchar_t>         wfstream;
    }

```

The pre-Stockholm Iostreams WG recommends adding decls for streampos (see 27-910). Nathan Myers will provide text for the proposal.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-906  
**Title:** add a typedef to access the traits parameter for a class.  
**Section:** 27  
**Status:** active  
**Description:**

Some classes; such as istream don't have access to the traits template parameter. Perhaps each class should provide a typedef for the traits parameter.

You need the traits parameter when you want to say stuff like:

```
cin.ignore(100, traits::newline(use_facet<ctype<cin.char_type>>(cin.getloc())) )
```

There is no way to get the traits type without saying something like: ios\_traits<cin.char\_type> which is almost reasonable, but it would be nicer to say something like: cin::traits\_type. There are some cases where ios\_traits is not the traits used to instantiate the stream.

**Possible Resolution:**

The Tokyo meeting recommends acceptance of the following:

Add "typedef traits traits\_type;" to basic\_ios and basic\_streambuf.

Where traits is the template parameter

The pre-Stockholm Iostreams WG recommends accepting the above resolution. Nathan Myers will provide text for the proposal.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-907  
**Title:** Use of "instance of" vs. "version of" in descriptions of class ios  
**Section:** 27.2 **Forward declarations [lib.iostream.forward]**  
**Status:** active  
**Description:**

Paragraph 2 and 3 describe the class ios and the class wios. One is described as "an instance of the template..." the other is described as "a version of the template...".

**Possible Resolution:**

Change paragraph 3 to:

“The class `wios` is an instance of the template class `basic_ios`, specialized by the type `wchar_t`”

Jerry Schwarz fixed the problem as editorial with the approval of the pre-Stockholm Iostreams WG. Therefore the issue should be closed.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-908  
**Title:** unnecessary ‘;’ (semicolons) in tables  
**Section:** 27  
**Status:** active  
**Description:**

There are unnecessary semicolons in tables in chapter 27. These probably should be removed.

**Possible Resolution:**

The only semicolons I can find are in section 27.1.2.6 **Type `POS_T` [lib.iostreams.pos.t]** “Table 2-Position type requirements”.

Jerry Schwarz fixed the problem as editorial with the approval of the pre-Stockholm Iostreams WG. Therefore the issue should be closed.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-909  
**Title:** Editorial issues (typo’s)  
**Section:** 27  
**Status:** active  
**Description:**

Here are a list of “typo’s” and other possible editorial issues.

**Editorial Issue #1**

**Description:**  
**27.4.4.3 `basic_ios` iostate flags functions [lib.iostate.flags]**  
The description of `ios_base::exceptions` is listed under the `basic_ios` clause.

**Possible Resolution:**  
This needs to be moved back to the `ios_base` clause.

Jerry Schwarz fixed the problem as editorial with the approval of the pre-Stockholm Iostreams WG. Therefore the issue should be closed.

**Editorial Issue #2**

**Description:**  
**27.1.2.4 `Type POS_T` [lib.iostreams.pos.t]**  
Description of type `POS_T` contains many awkward phrases. Needs rewriting for clarity.

The pre-Stockholm Iostreams WG declared the issue moot.

**Editorial Issue #3**

**Description:**27.1.2.3 **Type OFF\_T** [**lib.iostreams.off.t**]

Footnote 1 should say ``for one of" instead of ``for one if." Also, it should ``whose representation has at least" instead of ``whose representation at least."

The pre-Stockholm Iostreams WG recommends treating as editorial.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-910  
**Title:** remove streampos in favor of pos\_type  
**Section:** 27  
**Status:** active  
**Description:**

There are editorial boxes in Chapter 27 that say that streampos was deprecated but no resolution on what to do with functions that use it as an argument type has been offered.

**Possible Resolution:**

John wrote: "Change all references to streampos as an argument type to pos\_type. Each class in Chapter 27 has a typedef for, or access to, pos\_type."

My understanding is that we should add back streampos, has being the pos\_type used on streams instantiated on tiny characters. We also need to replace all the references to streampos by references to pos\_type throughout the iostreams chapter ( when streams are instantiated on char, pos\_type == streampos ).

The pre-Stockholm Iostreams WG recommends accepting the above resolution, and adding streampos to <iosfwd>. Tom Plum will provide text for the proposal.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-911  
**Title:** stdio synchronization  
**Section:** 27.3.1 Narrow stream objects [**lib.narrow.stream.objects**]  
**Status:** active  
**Description:**

Doing measurements on the performance of streambufs attached to stdin on a variety of systems, I found that the performance of a simple loop:

```
while ((c = cin.get()) != EOF) ...
```

was from 5 to 20 times slower than the equivalent

```
while ((c = getc(stdin)) != EOF) ...
```

To my horror, I found that this is a result of a mandate in the WP, that stdin and cin (and also stdout and cout) must be synchronized. As a goal this seems laudable, but if the consequence in many (most) environments is either:

1. an order of magnitude slower input, or

## 2. breaking link compatibility with C,

maybe we should reconsider this choice, and instead allow-but-not-require that the two be synchronized.

### Possible Resolution:

Nathan wrote:

“One possibility would be to reintroduce "sync\_with\_stdio" but give it a boolean argument. sync\_with\_stdio(true) would cause synchronization, while sync\_with\_stdio(false) would cause unsynchronization.

This would be agreeable to me. I take it this would be a static member of ios\_base? How would it default? I assume that the call with false could be a no-op.”

The pre-Stockholm Iostreams WG recommends that synchronization should be the default (status quo). Nathan Myers will write a proposal about adding in synch/unsynch functions, and will discuss when synch/unsynch can be called.

**Requestor:** Nathan Myers (ncm@cantrip.org)

---

**Issue Number:** 27-912  
**Title:** removing **Notes:** from the text  
**Section:** 27  
**Status:** active  
**Description:**

This issue is in response to Mats Meta list. It is an attempt to remove normative text from the WP. This issue removes **Notes:** from the text. Some **Notes:** clauses that need to be incorporated into the text will be handled in another issue.

Remove all **Notes:** clauses from the following:

**27.4.2.1 ios\_traits value functions [lib.ios.traits.values]**  
int\_type not\_eof(char\_type c)

**27.4.3.4 ios\_base storage functions [lib.ios.base.storage]**  
void \* & pword(int idx)

**27.5.2.2.3 Get area [lib.streampub.get]**  
int\_type snextc()

**27.5.2.4.3 Get area [lib.streampub.virt.get]**  
int showmanyc()

**27.5.2.4.3 Get area [lib.streampub.virt.get]**  
streamsize xsgetn(char\_type \*s, streamsize n)

**27.6.1.2.2 basic\_istream::operator>> [lib.istream::extractors]**  
basic\_istream<charT, traits>& operator>>(char\_type \*s)

**27.7.1.3 Overridden virtual functions [lib.stringbuf.virtuals]**

int\_type pbackfail(int\_type c)

#### 27.7.1.3 Overridden virtual functions [lib.stringbuf.virtuals]

int\_type overflow(int\_type c)

#### 27.8.1.4 Overridden virtual functions [lib.filebuf.virtuals]

int showmanyc()

### Possible Resolution:

The pre-Stockholm Iostreams WG recommends that Non-normative notes to “[Note:” style, change normative “Notes” to some other heading word. Jerry Schwarz will provide text for the proposal.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-913

**Title:** Incorporating **Notes:** into the text

**Section:** 27

**Status:** active

### Description:

The following **Notes:** clauses need to be incorporated into the WP text:

#### 27.5.2.1 basic\_streambuf constructors [lib.streambuf.cons]

basic\_streambuf()

#### 27.5.2.4.1 Locales [lib.streambuf.virt.locales]

void imbue(const locale&)

#### 27.5.2.4.3 Get area [lib.streambuf.virt.get]

int\_type underflow()

#### 27.5.2.4.4 Putback [lib.streambuf.virt.pback]

int\_type pbackfail(int c)

#### 27.5.2.4.5 Put area [lib.streambuf.virt.put]

int\_type overflow(int\_type c)

#### 27.6.1.1.1 basic\_istream constructors [lib.basic.istream.cons]

virtual ~basic\_istream()

#### 27.6.1.2.2 basic\_istream::operator>> [lib.istream::extractors]

basic\_istream<charT, traits>& operator>>(bool& n)

#### 27.6.1.3 Unformatted input functions [lib.istream.unformatted]

basic\_istream<charT, traits>& ignore(int n, int\_type delim)

#### 27.6.2.2 basic\_ostream constructors [lib.ostream.cons]

virtual ~basic\_ostream()

#### 27.6.2.4.2 basic\_ostream::operator<< [lib.ostream.inserters]

basic\_ostream<charT, traits>& operator<<(char\_type c)



Change this **Notes:** clause to a **Requires:** clause.

**27.7.1.1 basic\_stringbuf constructors [lib.stringbuf.cons]**  
explicit basic\_stringbuf(ios\_base::openmode)

**27.8.1.4 Overridden virtual functions [lib.filebuf.virtuals]**  
int\_type pbackfail(int\_type c)

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends that Non-normative notes to “[Note:” style, change normative “Notes” to some other heading word. Jerry Schwarz will provide text for the proposal.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-914  
**Title:** rethrowing exceptions  
**Section:** 27.6.2.4.1 Common requirements [lib ostream.formatted.reqmts]  
**Status:** active  
**Description:**

*[NOTE: This follows directly with 27-903 --John Hinke]*

The typical operator<< looks like this, given current semantics for exceptions:

```
{
  sentry cerberos(*this); if (!cerberos) return;
  iostate save = exceptions(); exceptions(0);

  try {
    if (use_facet< num_put<charT,ostreambuf_iterator<charT,traits> >>(
        getloc()).put(*this,*this,fill(),getloc(),val).failed())
      setstate(failbit); // won't throw
  }
  catch (...) { exceptions(save); setstate(badbit); throw; }
  exceptions(save); setstate(rdstate());
}
```

If we change exception semantics so that ios\_base::failure just gets rethrown, without setting badbit, we have instead:

```
{
  sentry cerberos(*this);
  if (!cerberos) return;
  try {
    if (use_facet< num_put<charT,ostreambuf_iterator<charT,traits> >>(
        getloc()).put(*this,*this,fill(),getloc(),val).failed())
      setstate(failbit); // might throw
  }
  catch (const ios_base::failure&) { throw; }
  catch (...) { setstate(badbit); throw; }
}
```

The examples don't constitute an argument for or against the change, but rather are suggestions for the example code that should appear in `[lib ostream.formatted.reqmts]` according to what is decided.

For the record, I am in favor of the change.

**Possible Resolution:**

We do not have any example code in 27.6.2.4.1 **Common requirements** `[lib ostream.formatted.reqmts]`, but if we want to add one, the ones described above are not correct.

The pre-Stockholm Iostreams WG recommends to add an example (editorial), but details will depend on a pending proposal from Jerry Schwarz about the “sentry” types.

<b>Requestor:</b>	Nathan Myers (ncm@cantrip.org)
<b>Issue Number:</b>	27-915
<b>Title:</b>	The use of specialization
<b>Section:</b>	27
<b>Status:</b>	active
<b>Description:</b>	

There is wording in Clause 27 such as:

- “...iostream classes are the instantiations of the...”
- “...class ios is an instance of the...”
- “...class wios is a version of the...”

This wording needs to be consistent with the rest of the document.

**Possible Resolution:**

Make the following changes to be consistent:

**27.1.1 Definitions [lib.iostreams.definitions]**  
Replace: “-- **narrow-oriented iostream classes** ...iostream classes are the instantiations of the...”  
With: “--**narrow-oriented iostream classes** ...iostream classes are specializations of the...”

**27.1.1 Definitions [lib.iostreams.definitions]**  
Replace: “-- **wide-oriented iostream classes** ...iostream classes are the instantiations of the...”  
With: “-- **wide-oriented iostream classes** ...iostream classes are specializations of the...”

**27.2 Forward declarations [lib.iostream.forward] paragraph 2**  
Replace: “The class ios is an instance of the template...”  
With: “The class ios is a specialization of the template...”

**27.2 Forward declarations [lib.iostream.forward] paragraph 3**  
Replace: “The class wios is a version of the template...”  
With: “The class wios is a specialization of the template...”

#### **27.4.2 Template struct ios\_traits [lib.ios.traits] paragraph 2**

Replace: “An implementation shall provide the following two instantiations of ios\_traits:”

With: “An implementation shall provide the following two specializations of ios\_traits:”

#### **27.5.2 Templates class basic\_streambuf<charT, traits> [lib.streambuf] paragraph 2**

Replace: “The class streambuf is an instantiation of the template...”

With: “The class streambuf is a specialization of the template...”

#### **27.5.2 Templates class basic\_streambuf<charT, traits> [lib.streambuf] paragraph 3**

Replace: “The class wstreambuf is an instantiation of the template...”

With: “The class wstreambuf is a specialization of the template...”

The pre-Stockholm Iostreams WG recommends to treat as editorial, but more work is needed to preserve distinction between explicit and implicit specialization.

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-916

**Title:** missing descriptions of specializations

**Section:** 27

**Status:** active

**Description:**

For compatibility, each templated class has two specializations. One for skinny characters and one for wide characters. For example:

```
template<class charT, class traits>
class basic_ios : public ios_base {
    //...
};
```

Class ios is a specialization of...

Class wios is a specialization of...

These descriptions are missing for some of the classes. This proposal adds these missing descriptions.

#### **Possible Resolution:**

Add the following descriptions to the appropriate sections:

For class basic\_ios:

##### **27.4.4 Template class basic\_ios [lib.ios]**

The class ios is a specialization of the template class basic\_ios specialized by the type char.

The class wios is a specialization of the template class basic\_ios specialized by the type wchar\_t.

For class basic\_istream:

##### **27.6.1.1 Template class basic\_istream [lib.istream]**

The class istream is a specialization of the template class basic\_istream specialized by the type char.

The class `wistream` is a specialization of the template class `basic_istream` specialized by the type `wchar_t`.

For class `basic_ostream`:

**27.6.2.1 Template class `basic_ostream` [`lib.ostream`]**

The class `ostream` is a specialization of the template class `basic_ostream` specialized by the type `char`.

The class `wostream` is a specialization of the template class `basic_ostream` specialized by the type `wchar_t`.

For class `basic_stringbuf`:

**27.7.1 Template class `basic_stringbuf` [`lib.stringbuf`]**

The class `stringbuf` is a specialization of the template class `basic_stringbuf` specialized by the type `char`.

The class `wstringbuf` is a specialization of the template class `basic_stringbuf` specialized by the type `wchar_t`.

For class `basic_istreamream`:

**27.7.2 Template class `basic_istreamream` [`lib.istreamream`]**

The class `istreamream` is a specialization of the template class `basic_istreamream` specialized by the type `char`.

The class `wistreamream` is a specialization of the template class `basic_istreamream` specialized by the type `wchar_t`.

For class `basic_ostringstream`:

**27.7.2.3 Template class `basic_ostringstream` [`lib.ostringstream`]**

The class `ostringstream` is a specialization of the template class `basic_ostringstream` specialized by the type `char`.

The class `wostringstream` is a specialization of the template class `basic_ostringstream` specialized by the type `wchar_t`.

For class `basic_filebuf`:

**27.8.1.1 Template class `basic_filebuf` [`lib.filebuf`]**

The class `filebuf` is a specialization of the template class `basic_filebuf` specialized by the type `char`.

The class `wfilebuf` is a specialization of the template class `basic_filebuf` specialized by the type `wchar_t`.

For class `basic_ifstream`:

**27.8.1.5 Template class `basic_ifstream` [`lib.ifstream`]**

The class `ifstream` is a specialization of the template class `basic_ifstream` specialized by the type `char`.

The class `wifstream` is a specialization of the template class `basic_ifstream` specialized by the type `wchar_t`.

For class `basic_ofstream`:

**27.8.1.8 Template class `basic_ofstream` [`lib.ofstream`]**

The class ofstream is a specialization of the template class basic\_ofstream specialized by the type char.

The class wofstream is a specialization of the template class basic\_ofstream specialized by the type wchar\_t.

The pre-Stockholm Iostreams WG recommends closing the issue with no change to the WP, since the proposal is subsumed by proposed change to <iosfwd> (27-904).

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-917  
**Title:** Editorial changes  
**Section:** 27.1.2 Type requirements [**lib.iostreams.type.reqmts**]  
**Status:** active  
**Description:**

27.1.2 [**lib.iostreams.type.reqmts**]: Last sentence: "... expects to the character container class." should read "... expects of the character container class."

**Possible Resolution:**

**Requestor:** Public Comment

---

**Issue Number:** 27-918  
**Title:** Validity of OFF\_T to POS\_T conversion  
**Section:** 27.1.2.3 Type OFF\_T [**lib.iostreams.off.t**]  
**Status:** active  
**Description:**

27.1.2.3 [**lib.iostreams.off.t**]: Paragraph 4: "Type OFF\_T is convertible to type POS\_T. But no validity of the resulting POS\_T value is ensured, whether or not the OFF\_T value is valid." Of what use is the conversion, then?

**Possible Resolution:**

The pre-Stockholm Iostreams WG recommends to treat as editorial and added a footnote.

**Requestor:** Public Comment

---

**Issue Number:** 27-919  
**Title:** Question on Table 2 assertions  
**Section:** 27.1.2.4 Table2 Position type requirements [**lib.iostreams.pos.t**]  
**Status:** active  
**Description:**

27.1.2.4 [**lib.iostreams.pos.t**]: table 2: first row has assertion "p == P(i)" but p does not appear in the expression for that row; also, that row has the note "a destructor is assumed" -- what does this mean?

**Possible Resolution:**

The first row of table 2 should be deleted. The second row already specifies the construction and assignment from an integer value.

The pre-Stockholm Iostreams WG recommends accepting the above resolution. A larger issue is the table was voted out of the iostreams chapter as part of traits consolidation, but needs to be somewhere. The discussions of OFF\_T and POS\_T should be consolidated in iostreams chapter, with a note added to string chapter referring to this chapter. Philippe Le Mouël will provide text for the proposal.

**Requestor:** Public Comment

---

**Issue Number:** 27-920  
**Title:** destination of clog and wclog  
**Section:** 27.3.1 Narrow stream objects [**lib.narrow.stream.objects**],  
27.3.2 Wide stream objects [**lib.wide.stream.objects**]  
**Status:** active  
**Description:**

There is currently an editorial box concerning the destination of clog and wclog. I would like to propose the following resolution:

**Possible Resolution:**

Change **27.3.1 Narrow stream objects** [**lib.narrow.stream.objects**] paragraph 6 to:  
The object clog controls output to an implementation defined stream buffer.

Change **27.3.2 Wide stream objects** [**lib.wide.stream.objects**] paragraph 6 to:  
The object wclog controls output to an implementation defined stream buffer.

The pre-Stockholm Iostreams WG recommends closing with no change to the WP (delete the editorial box).

**Requestor:** John Hinke (hinke@roguewave.com)

---

**Issue Number:** 27-921  
**Title:** default locale argument to constructor  
**Section:** 27  
**Status:** active  
**Description:**

Default locale arguments for stream constructors.

istream and ostream constructors (and all derivations) should have a default locale argument, in the manner of

```
obogusstream(const char *name,const locale& l = locale());
```

**Possible Resolution:**

Add a new argument to the standard stream constructors:

```
const locale& l = locale::global()
```

Add this new argument to the following classes' constructors:

```
basic_istream,  
basic_ostream,  
basic_istreamstream,  
basic_ostreamstream,  
basic_ifstream,  
basic_ofstream  
istrstream  
ostrstream
```

The pre-Stockholm Iostreams WG recommends closing with no change to the WP

**Requestor:** Nathan Myers (ncm@cantrip.org)  
Norihiro Kumagai (kuma@slab.tnr.sharp.co.jp)

---

**Issue Number:** 27-922  
**Title:** Cleaning the iostreams header synopsis #include of other headers  
**Section:** Chapter 27 and Annex D  
**Status:** active  
**Description:**

Several public comments pointed out that the C++ header synopsis #include of other C++ headers were not correct.

**Possible Resolution:**

Add to header <iostream> synopsis: <istream> and <ostream>  
Add to header <sstream> synopsis: <string> ( already brought by locale )  
Add to header <streambuf> synopsis: <locale>  
Add to header <strstream> synopsis: <streambuf>

**Requestor:** Judy Ward

---

## Annex D issues

**Issue Number:** 27-1001  
**Title:** description of function `setbuf` is not sufficient  
**Section:** D.6.1.3 **strstreambuf overridden virtual functions** [`depr.strstreambuf.virtuals`]  
**Status:** active  
**Description:**

Description of the overridden `setbuf(char* s, streamsize n)` function in class `strstreambuf` is not sufficient.

**Possible Resolution:**

Change the current description of function `setbuf(char* s, streamsize n)` in D.6.1.3 **strstreambuf overridden virtual functions** [`depr.strstreambuf.virtuals`] to:

```
streambuf* setbuf( char* s, streamsize n);
```

and not:

```
streambuf<char>* setbuf(char* s, streamsize n);
```

**Effects:** If *s* is not a null pointer, and  $n > pptr() - pbase()$ , replace the current buffer (copy its contents and deallocate it) by the buffer of size *n* pointed at by *s*.

In the case where *s* is a null pointer, and  $n > pptr() - pbase()$  resize the current buffer to size *n*.

If the function fails, it returns a null pointer.

**Returns:** (`streambuf*`)(`this`)

I am not qualified enough to decide if the return type should be changed to `strstreambuf*` as proposed by Steve Clamage in issue 27-809. I tried it with several compilers, and the results were just error messages. Basically, the compilers were complaining about the fact that the base class virtual function and the overridden virtual function should have the same return type.

The pre-Stockholm Iostreams WG recommends the behavior of the `setbuf` function to be implementation-defined, except that `setbuf(0,0)` has no effect. The return type in the WP is correct. Steve Clamage will provide text for the proposal.

**Requestor:** philippe Le Mouël ([philippe@roguewave.com](mailto:philippe@roguewave.com))

---

**Issue Number:** 27-1002  
**Title:** `strstreambuf` Editorial issues  
**Section:** D.6.1 **Class `strstreambuf`** [`depr.strstreambuf`]  
**Status:** active  
**Description:**

Class `strstreambuf` contains several typos and is also missing some typedefs.

**Possible Resolution:**

The following typedefs need to be added to class `strstreambuf` (D.6.1 **Class `strstreambuf`** [`depr.strstreambuf`]) :



- typedef char\_traits<char>::int\_type int\_type;

This typedef is used in the `strstreambuf` overridden virtual functions *overflow*, *pbackfail* and *underflow*.

- typedef char\_traits<char>::pos\_type pos\_type;

This typedef is used in the `strstreambuf` overridden virtual functions *seekoff* and *seekpos*.

- typedef char\_traits<char>::off\_type off\_type;

This typedef is used in the `strstreambuf` overridden virtual function *seekoff*.

In D.6.1 **Class `strstreambuf` [`depr.strstreambuf`]** the notation of the `strstreambuf` base class is wrong it should say:

```
class strstreambuf : public basic_streambuf<char>
```

and not:

```
class strstreambuf : public streambuf<char> // does not exist
```

In D.6.1 **Class `strstreambuf` [`depr.strstreambuf`]** the declaration of function `freeze` is missing the argument name. It should say:

```
void freeze(bool freeze_flg = 1);
```

and not:

```
void freeze(bool = 1);
```

The pre-Stockholm Iostreams WG recommends treating as editorial. The corrections have already been made.

---

**Requestor:** Philippe Le Mouël (philippe@roguewave.com)

**Issue Number:** 27-1003

**Title:** istream Editorial issues (typos)

**Section:** D.6.2 **Template class `istream` [`depr.istream`]**

**Status:** active

**Description:**

Class `istream` contains several typos.

**Possible Resolution:**

In D.6.2 **Template Class `istream` [`depr.istream`]** the previous title should be changed to “D.6.2 Class `istream`”, because the class is not a template class.

In D.6.2 **Template Class `istream` [`depr.istream`]** the notation of the `istream` base class is wrong. It should say:

```
class istream : public basic_istream<char>
```

and not:

```
class istream : public istream<char> // does not exist
```

The pre-Stockholm Iostreams WG recommends treating as editorial. The corrections have already been made.

**Requestor:** Philippe Le Mouël (philippe@roguewave.com)

---

**Issue Number:** 27-1004  
**Title:** ostream Editorial issues (typos)  
**Section:** D.6.3 **Template class ostream** [depr.ostream]  
**Status:** active  
**Description:**

Class ostream contents several typos.

**Possible Resolution:**

In D.6.3 **Template Class ostream** [depr.ostream] the previous title should be changed to “D.6.3 Class ostream”, because the class is not a template class.

In D.6.3 **Template Class ostream** [depr.ostream] the notation of the ostream base class is wrong. It should say:

```
class ostream : public basic_ostream<char>
and not:
class ostream : public ostream<char> // does not exist
```

In D.6.3 **Template Class ostream** [depr.ostream] and D.6.3.2 **Member functions**[depr.ostream.members] the declaration of function `void freeze( int freezeFl = 1 )` is not consistent with the declaration in D.6.1 **Class ostreambuf** [depr.ostreambuf], which is `void freeze( bool freezeFl = 1 )`. The argument should be `bool` or `int`, but not `bool` in one and `int` in the other.

The pre-Stockholm Iostreams WG recommends treating as editorial. The corrections have already been made.

**Requestor:** Philippe Le Mouël (philippe@roguewave.com)

---