IOStreams Issues List
Library Clause 27

By: John Hinke, Quantitative Data Systems
jhinke@qds.com

NOTE: I've included the Pre-Austin issues lists because some of those˝issues are still open. The closed issues are also here with a description of the resolution. The Pre-Austin Document˝numbers are: X3J16/95-0034
WG21/N0634

Summary of Issues

27.4.2 `ios_traits`

Active	27-001	Making `newline` locale aware
Active	27-002	`is_whitespace` is inconsistent
Active	27-003	Mention of base struct `string_char_traits`
Active	27-004	Adding `space` function that returns a "space" character.
Active	27-005	example of changing the behavior of `is_whitespace` is incorrect.
Active	27-006	`not_eof` specification

27.4.3 `ios_base`

Closed	27-101	`exceptions` called a method that required template arguments
Active	27-102	`fill` function needs to use template character type
Active	27-103	`ios_base::ios_base()` effects
Active	27-104	`ios_base` manipulators

27.4.4 `basic_ios`

27.5.2 `basic_streambuf`

Closed	27-301	imbued locale mentioned, but missing from the interface
Closed	27-302	`stossc` missing from the interface
Open	27-303	`sputc` parameters need to reflect the traits parameters
Active	27-304	imbuing on streambufs. When, how often, etc...
Active	27-305	`sungetc` has an incorrect return type
Active	27-306	`not_eof` needs to be used where appropriate
Active	27-307	`uflow` needs editing

27.6.1 `basic_istream`

Active	27-401	`isfx` what does it do?
Active	27-402	`ipfx` example is incorrect
Active	27-403	`readsome` returns an incorrect value

Active 47-404 Clarification of exceptions thrown

27.6.2 `basic_ostream`

Active 27-501 `op<<(char)` needs to be consistant with the other formatted inserters
Open 27-502 `op<<(void *)` should it be `const volatile void *`
Closed 27-503 `op<<(basic_streambuf&)`. Should it be `const basic_streambuf&`
Active 27-504 `put` has an incorrect return type

27.6.1-27.6.2 `basic_istream`, `basic_ostream`

Active 27-601 `op[<<|>>](basic_streambuf *)` Error status on `NULL` streambuf
Active 27-602 `op[<<|>>](ios_base&)` needed for manipulators
Closed 27-603 `op[<<|>>](basic_streambuf&)` differs from the original version
Active 27-604 positional typedefs in `istream`/`ostream` derived classes are not needed
Active 27-605 `read`/`write` should take a `void *` instead of a `char_type *`
Closed 27-606 `seekg`, `tellg`, `seekp`, `tellp` missing from the interface
Active 27-607 Should we require `ios::in` to be set for `istream`'s and `ios::out` to be set for `ostream`'s?
Active 27-608 Should get/put use iterators?

27.7 `basic_stringbuf`, `basic_istringstream`, `basic_ostringstream`

Closed 27-701 `basic_stringbuf` needs to mention behavior of `ios_base::openmode`
Active 27-702 `str()` needs to clarify return value on else clause
Active 27-703 string stream classes need to have `string_char_traits` and `allocator` parameters

27.8.2 `basic_filebuf`, `basic_ifstream`, `basic_ofstream`

Active 27-801 Table 83 (File Open Modes) incomplete
Active 27-802 `filebuf::underflow` example is incorrect
Active 27-803 `filebuf::open` calls `basic_streambuf` constructor

Miscellaneous

Closed 27-901 Do ios type classes access the streambuf on destruction...
Closed 27-902 Do streambuf classes access the buffer on destruction...
Closed 27-903 public typedefs in the interface, static functions in interface
Active 27-904 input/output of `unsigned char`, and `signed char`
Closed 27-905 `basic_` manipulators were mentioned. No longer needed
Active 27-906 default locale arguments for stream constructors
Active 27-907 `ipfx`/`opfx`/`isfx`/`osfx` not compatible with exceptions.
Active 27-908 `iosfwd` declarations incomplete
Active 27-909 `iostream` type classes are missing.
Active 27-910 add a typedef to access the traits parameter in each stream class.
Active 27-911 templatized function in `resetiosflags`
Active 27-912 `<ios>` synopsis is incomplete

Editorial Boxes

Box 126	**27.1.1**	**Definitions**	**[lib.iostreams.definitions]**
Box 127	**27.3**	**Standard iostream objects**	**[lib.iostream.objects]**
Box 128	**27.3.1**	**Narrow stream objects**	**[lib.narrow.stream.objects]**
Box 129	**27.3.2**	**Wide stream objects**	**[lib.wide.stream.objects]**
Box 130	**27.4.2**	**Template `struct ios_traits`**	**lib.ios.traits]**
Box 131	**27.4.2.2**	**`ios_traits` value functions**	**[lib.ios.traits.values]**
Box 132	**27.4.2.2**	**`ios_traits` value functions**	**[lib.ios.traits.values]**
Box 133	**27.4.2.4**	**`ios_traits` conversion functions**	**[lib.ios.traits.convert]**
Box 134	**27.4.2.4**	**`ios_traits` conversion functions**	**[lib.ios.traits.convert]**
Box 135	**27.4.3**	**Class `ios_base`**	**[lib.ios.base]**
Box 136	**27.4.3**	**Class `ios_base`**	**[lib.ios.base]**
Box 137	**27.4.3.1.6**	**Class `ios_base::Init`**	**[lib.ios::Init]**
Box 138	**27.4.3.5**	**`ios_base` constructors**	**[lib.ios.base.cons]**
Box 139	**27.4.4**	**Template class `basic_ios`**	**[lib.ios]**
Box 140	**27.4.4**	**Template class `basic_ios`**	**[lib.ios]**
Box 141	**27.4.4.1**	**`basic_ios` constructors**	**[lib.basic.ios.cons]**
Box 142	**27.4.4.1**	**`basic_ios` constructors**	**[lib.basic.ios.cons]**
Box 143	**27.4.4.2**	**Member functions**	**[lib.basic.ios.members]**
Box 144	**27.5.2.1**	**`basic`_streambuf constructors**	**[lib.streambuf.cons]**
Box 145	**27.5.2.4.2**	**Buffer management and positioning**	**[lib.streambuf.virt.buffer]**
Box 146	**27.5.2.4.3**	**Get area**	**[lib.streambuf.virt.get]**
Box 147	**27.6.1.2.1**	**Common requirements**	**[lib.istream.formatted.reqmts]**
Box 148	**27.6.1.2.1**	**Common requirements**	**[lib.istream.formatted.reqmts]**
Box 149	**27.6.2.3**	**`basic_ostream` prefix and suffix functions**	**[lib.ostream.prefix]**
Box 150	**27.6.2.4.1**	**Common requirements**	**[lib.ostream.formatted.reqmts]**
Box 151	**27.6.2.4.1**	**Common requirements**	**[lib.ostream.formatted.reqmts]**
Box 152	**27.6.2.4.1**	**Common requirements**	**[lib.ostream.formatted.reqmts]**
Box 153	**27.6.2.4.1**	**Common requirements**	**[lib.ostream.formatted.reqmts]**
Box 154	**27.7.1**	**Template class `basic_stringbuf`**	**[lib.stringbuf]**
Box 155	**27.7.1.3**	**Overridden virtual functions**	**[lib.stringbuf.virtuals]**
Box 156	**27.7.1.3**	**Overridden virtual functions**	**[lib.stringbuf.virtuals]**
Box 157	**27.7.1.3**	**Overridden virtual functions**	**[lib.stringbuf.virtuals]**
Box 158	**27.8.1**	**File streams**	**[lib.fstreams]**
Box 159	**27.8.1.4**	**Overridden virtual functions**	**[lib.filebuf.virtuals]**
Box 160	**27.8.1.4**	**Overridden virtual functions**	**[lib.filebuf.virtuals]**
Box 161	**27.8.1.4**	**Overridden virtual functions**	**[lib.filebuf.virtuals]**
Box 162	**27.8.1.4**	**Overridden virtual functions**	**[lib.filebuf.virtuals]**
Box 163	**27.8.1.4**	**Overridden virtual functions**	**[lib.filebuf.virtuals]**

Open Issues

Issue Number: 27-001
Title: changing `traits::newline` to be locale aware
Section: 27.4.2.2 **`ios_traits` value functions**
Status: **active**
Description:

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

This proposal addresses the need for a locale-aware newline.

Possible Resolution:

Change `traits::newline` by adding a parameter for locale information:

```
static char_type newline(const ctype<char_type>& ct);
```

The default definition is as if it returns:

```
ct.widen('\n');
```

Some functions in `basic_istream` have a default parameter that is: `traits::newline()` (`getline`, `get`). These defaults will have to be changed to use the currently imbued locale. Changing the default value to: `traits::newline(getloc())` won't work because `getloc()` is not static. This would require that the functions that have `newline()` as a default value will need to be split into two functions. One function that has three parameters, and one function that has˝two parameters and calls the three parameter function with a "default" value. For example:

```
istream_type& getline(char_type *, streamsize, char_type delim);

istream_type& getline(char_type *s, streamsize n)
   { return getline(s, n, newline(getloc().template
   use<ctype<char_type> >())); }
```

The functions that would need to change are:

```
istream_type& get(char_type *, streamsize, char_type);
istream_type& get(streambuf_type&, char_type);
istream_type& getline(char_type *, streamsize, char_type);
```

Requestor: Nathan Myers (myersn@roguewave.com),
John Hinke(jhinke@qds.com)

Issue Number: 27-002
Title: `traits::is_whitespace()` is inconsistent
Section: 27.4.2.3 **`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(const ctype<char_type>&, char_type);
```

27.4.2.3 `ios_traits` test functions [lib.ios.traits.tests]

```
bool is_whitespace(char_type, const ctype<char_type>&);
```

27.6.1.1.2 `basic_istream::ipfx` [lib.istream.prefix]

Notes: ...uses the function

```
bool traits::is_whitespace(charT, const locale *)
```

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 problem is which signature is correct. The purpose of this function is to check for whitespace characters. It will most commonly be used inside a tight loop where the lookup of the `ctype` facet could be very expensive. I propose the following option:

```
static bool is_whitespace(char_type c, const ctype<char_type>& ct);
```

Returns: `true` if *`c`* represents one of the white space characters. The default definition is as if it returns *`ct`*`.is(`*`ct`*`.space, `*`c`*`)`.

Side note: 27.4.2.3 `ios_traits::is_whitespace`: The returns paragraph calls a method of `ctype` that does not exist.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-003
Title: mention of base struct `string_char_traits`
Section: 27.4.2.3 **`ios_traits` test functions** paragraph 1 **[lib.ios.traits.tests]**
Status: **active**
Description:

27.1.2.1 **Type *`CHAR_T`*** paragraph 2:
"The base class (or struct), `string_char_traits` provides the definitions common between the string class templates and the iostream class templates."

27.4.2.3 paragraph 1:
"...the following three functions provided from the base struct `string_char_traits<CHAR_T>`."

`ios_traits` is not derived from `string_char_traits`.

Possible Resolution:

Remove the sentence from 27.1.2.1. Remove 27.4.2.3 paragraph 1.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-004
Title: Add `ios_traits::space` that returns a space character.
Section: 27.4.2 **Template struct `ios_traits` [lib.ios.traits]**
Status: **active**
Description:

The space character should be made available in the `ios_traits` class. The default fill character is a "space" character. This character may not be available for all˝character sets and should be placed in the `ios_traits` class for specialization.

Possible Resolution:

Add the following member to `ios_traits`:

```
static charT space(const ctype<charT>& ct);
```

Returns: The space character for the given locale. The default definition is as if it˝evaluates:

```
return ct.widen(' ');
```

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-005
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:

The example of changing behavior of `is_whitespace` is incorrect. It should read:

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

Possible Resolution:
Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-006
Title: `not_eof` specification
Section: 27.4.2.2 `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:
Requestor: Per Bothner (bothner@cygnus.com)

Issue Number: 27-102
Title: `ios_base::fill`
Section: 27.4.3.2 **`ios_base` `fmtflags` state functions [lib.fmtflags.state]**
Status: **active**
Description:

The `ios_base::fill` functions use a type that depends on the template type; however, `ios_base` is a non-templatized class.

Possible Resolution:

Move the `fill` functions to `basic_ios`. This would allow them to be templatized on the character type. The fill character is really only used for output, but moving this to `basic_ostream` would break code that expected it to be in `ios`.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-103
Title: `ios_base::ios_base()` effects
Section: 27.4.3.5 **`ios_base` constructors [lib.ios.base.cons]**
Status: **active**
Description:

Table 72 lists the effects of calling the `ios_base` constructor. There are several issues relating to this constructor.

Issue 1:

The `rdstate()` element has a value that is based on *`sb`* which is a streambuf located in `basic_ios<>`. Change the value of `rdstate()` to `goodbit` and ignore checking the streambuf. The `basic_ios` constructor will handle checking the streambuf.

Issue 2:

The `fill()` element needs to be moved to `basic_ios`. This falls in line with Issue 27-102.

Possible Resolution:
Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-104
Title: `ios_base` manipulators
Section: 27.4.5 **`ios_base` manipulators [lib.std.ios.manip]**
Status: **active**
Description:

There is only one `ios_base` manipulator that says, "Does not affect any extractors." `(showbase)`

This implies that the rest of the manipulators affect extractors. If the˝manipulators only affect insertors (ignoring `skipws`), then perhaps they should be `ostream` manipulators instead of `ios_base` manipulators. If they are left as `ios_base` manipulators, then they should affect extractors as well as insertors.

The locale `num_get` facet says, "Reads characters from *`in`*, interpreting them according to *`str`*`.flags()`..." This implies that the manipulators affect the extraction of values˝from a stream.

A couple of cases:

```
unsigned int    ui;
int             i;

cout << -10;
cin >> ui;      // What should this read in?
cout << showpos << 10; // +10
cin >> ui;      // What about this?

cout << showbase << hex << 10; // 0xa
cin >> i;       // Should this be valid?
cin >> showbase >> hex >> i; // What about this?
```

Possible Resolution:

Keep all manipulators as they are but say something to the effect that the manipulators affect both insertors and extractors. Remove the Notes on `showbase`. This is different behavior than the original AT&T implementation.

Editorial Issue: These manipulators should be moved to the `ios_base` clause.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-303
Title: `basic_streambuf::sputc(int c)`
Section: 27
Status: closed
Description:

Bernd Eggink says,

"I think the parameter shold be '`int_type`' instead of '`int`'. May this be called with parameter `eof()`? If so, what will happen? What is the effect of calling `sputc()` with an arbitrary int parameter?"

Possible Resolution:

Eggink's proposal:

"A consistent solution would be to change 27.2.1.2.13 like this:

```
int_type sputc(int_type c);
```

If the output sequence does not have a write position available, returns `overflow(c)`. Otherwise, if `c != eof()`, returns `(*pnext++ = (char_type)c)`. Otherwise returns a value `!= eof()`."

Schwarz' proposal:

"The return type should clearly be `int_type`. I think that was just overlooked in the templatization. I think the argument type should be `char_type`.

For ordinary char streambuf's `sputc(-1)` is supposed to insert `0xFF`. At least that is what it did in the original implementation. Note that under your description the function˝would do different things depending on whether or not a write position is available. The orginal˝implementation does `overflow(zapeof(c))` when it calls overflow. `zapeof` was a macro that corresponds to the version of `not_eof` accepted at VF.

```
int_type not_eof(char_type);
```

So I think we should say it calls:

```
overflow(not_eof(c));
```

in case a write position isn't available."

The accepted solution was Schwarz'.

Discussion:

The WP does not reflect the accepted solution. The return type is still `int`. This should be changed to `int_type`.

Requestor: Bernd Eggink (admin@rrz.uni-hamburg.de)
John Hinke (jhinke@qds.com)

Issue Number: 27-304
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:

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.

Requestor: Nathan Myers (myersn@roguewave.com)

Issue Number: 27-305
Title: `int streambuf::sungetc()`
Section: 27.5.2.2.4 **Putback [lib.streambuf.pub.pback]**
Status: **active**
Description:

The function `int basic_streambuf::sungetc()` has a return type that should be `int_type`.

Possible Resolution:

Change **27.5.2: Template `class` `basic_streambuf<charT, traits>` [lib.streambuf]**

```
int_type sungetc();
```

Change **27.5.2.2.4: `basic_streambuf::sungetc` [lib.streambuf.pub.pback]**

```
int_type sungetc();
```

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-306
Title: `not_eof` needs to be used where appropriate
Section: 27
Status: **active**
Description:

27.5.2.2.3 Get area **[lib.streambuf.pub.get]**

```
int_type sbumpc();
```

Returns: "...returns `char_type(*gptr())`..."

This should be changed to say, "...returns `not_eof(*gptr())`..."

```
int_type sgetc();
```

Returns: "...returns `char_type(*gptr())`."

This should be changed to say, "...returns `not_eof(*gptr())`..."

Possible Resolution:
Requestor: Per Bothner (bothner@cygnus.com)

Issue Number: 27-307
Title: `uflow` needs editing
Section: 27
Status: **active**
Description:

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

```
int_type uflow();
```

Default behavior: "...returns `*gptr()`."

This should be changed to, "...returns `not_eof(*gptr())`."

Returns: `traits::not_eof(c)`

This should be changed to, "`traits::not_eof(*gptr())`"

Possible Resolution:
Requestor: Per Bothner (bothner@cygnus.com)

Issue Number: 27-401
Title: `istream::isfx`
Section: 27.6.1.1.2 **`basic_istream` prefix and suffix [lib.istream.prefix]**
Status: **active**
Description:

What is the purpose of this function? The WP says, "**Effects:** None." Should it do something more? Or is it implementation defined!

Possible Resolution:

This function should be deprecated in favor of 27-908

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-402
Title: examples for `ipfx`
Section: 27.6.1.1.2 **`basic_istream` prefix and suffix [lib.istream.prefix]**
Status: **active**
Description:

The example for a "typical" implementation of `ipfx()` has an incorrect function declaration. It should read:

```
template<class charT, class traits>
bool basic_istream<charT, traits>::ipfx(bool noskipws)
```

Possible Resolution:

This function should be fixed and deprecated in favor of 27-907

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-403
Title: `streamsize istream::readsome`
Section: 27.6.1.3 **Unformatted input functions [lib.istream.unformatted]**
Status: **active**
Description:

The return for `in_avail() > 0` will return a value that doesn't match the return type for `readsome`.

Possible Resolution:

Change the description for `in_avail > 0` to:

If `in_avail > 0`, calls `read(s, min(in_avail(), n))`, and returns `min(in_avail(), n)`.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-406
Title: Clarification of exceptions thrown
Section: 27.6.1.1 **Template `class basic_istream` [lib.istream]**
Status: **active**
Description:

27.6.1.1 paragraph 4 says

"If one of these called functions throws an exception, then unless noted˝otherwise the input function calls `setstate(badbit)` and if `badbit` is on in `exception()` (sic) rethrows the exception without completing its actions."

Problem: If `badbit` is on in `exceptions()` then `ios_base::clear`, which is called by `setstate(badbit)`, will throw an object of `ios_base::failure` and the original exception will NEVER be rethrown, i.e., it will be lost.

Discussion:

Jerry Schwarz,

"This has been discussed a lot. My preference has always been that if any˝of the virtuals throws an exception then

a) set `badbit` in error state
b) check `badbit` in exception state
  b1) if its on then rethrow the original exception
  b2) do not throw anything, treat as an error.

"Other implementors have complained that this was hard to do, and have˝preferred to just let the exception be passed through without being caught at all.

"Other people think that all iostream operations should only through `ios_base::failure`."

Possible Resolution:

Requestor: Modena Software (modena@netcom.com)

Issue Number: 27-501
Title: `ostream<<(char)` : formatting, padding, width
Section: 27.6.2.4.2 `basic_ostream::operator<<`
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.

Possible Resolution:

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.

Requestor: John Hinke (jhinke@qds.com),
Bernd Eggink (admin@rrz.uni-hamburg.de)

Issue Number: 27-502
Title: `ostream::operator<<(void *)`
Section: 27.2.4.1
Status: **Open**
Description:

`ostream& operator<<(void *)`

should take '`const volatile void *`' rather than `void *`.

Resolution:

The function now takes a `const void *`.

ReOpened:

Does anyone know why the resolution was for it to take a `const void *` rather than a `const volatile void *`?

I can't think of any good reason why we should make the code:

```
#include <iostream>
volatile int x;
int main() {
   cout << & x;
   return 0;
}
```

ill-formed.

Requestor: Fergus Henderson (fjh@munta.cs.mu.oz.au)

Issue Number: 27-504
Title: `int ostream::put(char_type c)`
Section: 27.6.2.5.1 **`basic_ostream::put`**
Status: **active**
Description:

The current return type should be changed from `int` to `int_type`.

Also, the **Effects:** says, "returns `(unsigned char)c`". This should be changed to some appropriate value (I'm not sure what that is!)

Possible Resolution:
Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-601
Title: `operator<<(streambuf *)`, and `operator>>(streambuf *)` error status on `NULL` streambuf
Section: 27.6.1.2.2, 27.6.2.4.2 **[lib.istream::extractors], [lib.ostream.inserters]**
Status: active
Description:

It is currently undefined as to what happens when a `NULL` streambuf is passed to these functions.

Possible Resolution:

Need to add wording that says if the streambuf is `NULL`, then `setstate(failbit)`.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-602
Title: `istream::operator>>(ios_base&)`, `ostream::operator<<(ios_base&)`
Section: 27.6.1.2.2, 27.6.2.4.2 **[lib.istream::extractors], [lib.ostream.inserters]**
Status: active
Description:

The `ios_base` manipulators 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 insertor/extractor for `istream` and `ostream` that does insertion/extraction for `ios_base`.

Possible Resolution:

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.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-604
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_istringstream
basic_ostringstream
basic_ifstream
basic_ofstream
```

Possible Resolution:

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

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-605
Title: `istream::read`, `ostream::write`
Section: 27.6.1.3, 27.6.2.5
[lib.istream.unformatted], [lib.ostream.unformatted]
Status: **active**
Description:

```
istream& istream::read(char_type *,streamsize);
ostream& ostream::write(const char_type *,streamsize);
```

These functions are typically used for binary data.

Possible Resolution:

These function 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` which parallels the behavior of `get`.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-607
Title: Opening an `istream` without `ios::in` set? or an `ostream` without `ios::out` set?
Section: 27.6.1.1, 27.6.2
[lib.input.streams], [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:

Should we enfore this policy? Does it ever make sense to open an `istream` for writing or an `ostream` for reading?

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

Issue Number: 27-608
Title: get/put type functuions should be able to use iterators.
Section: 27
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 took 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:

Requestor: Nathan Myers (myersn@roguewave.com)

Issue Number: 27-702

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 75 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:

Requestor: Angelika Langer (Angelika.Langer@mch.sni.de)
Bernd Eggink (admin@rrz.uni-hamburg.de)

Issue Number: 27-703

Title: string streams need allocator and `string_char_traits` parameters

Section: 27.7.1 Template `class basic_stringbuf`
Status: **active**
Description:

The string streams are currently templatized on the character type (`charT`) and the traits type (`ios_traits`). String template parameters need to be added.

Possible Resolution:

I propose to change the template parameters of the string streams from:

```
template<class charT, class traits = ios_traits<charT> >
```

to:

```
template<class charT, class IOS_traits = ios_traits<charT>,
   class STRING_traits = string_char_traits<charT>,
   class Allocator = allocator>
```

All references to `basic_string`, or any of the string stream classes will need to be fixed.

All references to `traits` should be replaced by either `IOS_traits` or `STRING_traits`.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-801
Title: Table 83 (File Open Modes) is incomplete
Section: 27
Status: **active**
Description:

The table is incomplete in describing all possible combination of modes. Something should be said about modes not listed in this table?

Possible Resolution:

Either:

- fill in the table and list all possible combinations
- or say that the unlisted modes are undefined.

I think that the table should be filled in.

Requestor: Jerry Schwarz (jss@declarative.com)

Issue Number: 27-802
Title: `filebuf::underflow` example
Section: 27
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:

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-803
Title: `filebuf::open` calls `basic_streambuf` constructor
Section: 27
Status: **active**
Description:

The **Effects:** clause says, "calls `basic_streambuf<charT, traits>::basic_streambuf()`" This type of call is illegal.

Possible Resolution:

A member function should be added to `basic_streambuf` that does the initialization. Then change `basic_filebuf::open` so that it calls that function.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-904
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:

I propose two different solutions. Both of them solve the problem.

Solution #1

I propose to change the current member functions that "use" `charT`'s as the argument type to `char` and `wchar_t`. For example:

Replace:

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

With:

```
istream& istream::operator>>(char&);
istream& istream::operator>>(signed char&);
istream& istream::operator>>(unsigned char&);
istream& istream::operator>>(wchar_t&);
```

Users can easily add a new global insertion/extraction operator for their new character type. They can also derive from `istream` or `ostream` and add their own unformatted IO functions for their new character type.

This would also solve the problem of inserting skinny characters into a wide stream or wide characters into a skinny stream.

For the unformatted IO functions, we replace:

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

with:

```
istream& istream::get(char *, streamsize, char);
istream& istream::get(unsigned char *, streamsize, unsigned char);
istream& istream::get(signed char *, streamsize, signed char);
istream& istream::get(wchar_t *, streamsize, wchar_t);
```

We would also need to replace the other members that make sense reading or writing `unsigned char`, or `signed char` values.

This would still allow users to have streams of `unsigned char`, or any other type.

Solution #2

Leave the classes as they are, but add several new member functions. For example:

Leave this member function:

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

and add these member functions:

```
istream& istream::operator>>(unsigned char&);
istream& istream::operator>>(signed char&);
```

For the unformatted IO functions we leave this member function:

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

and add these member functions:

```
istream& istream::get(unsigned char *, streamsize, unsigned char);
istream& istream::get(signed char *, streamsize, signed char);
```

This would still allow users to create their own character type class and also provide backward compatibility. However, this would mean that users could not have `istream<unsigned char>`, which I think is a resonable restriction.

This would not solve the skinny-character-on-wide-stream problem, though. To solve this problem, we can overload the formatted functions:

We can define global inserters/extractors for these special cases:

```
namespace std {
 ostream&  operator<<(ostream&, wchar_t);
 wostream& operator<<(wostream&, char);

 istream&  operator>>(istream&, wchar_t&);
 wistream& operator>>(wistream&, char&);
}
```

This would still not allow us to to insert a skinny-character-on-wide-stream using the unformatted IO routines. I'm not sure if that is a real problem or not. If you need to use the unformatted operations, you could easily use either `read` or `write`.

The following functions would need to be changed for either solution:

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

ostream& operator<<(char_type *);
ostream& operator<<(char_type);
```

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-906
Title: default locale arguments
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::classic());
```

or perhaps:

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

Norihiro Kumagai <kuma@slab.tnr.sharp.co.jp> replies:

> In order to cordinate the C-language locale model, I believe that the default locale value should not be '`locale::classic ()`', what we call "C" locale, but be '`locale::global()`', the current global locale.

Most likely, it should probably be `locale::global()`.

The next issue is when can the locale change? There is nothing that says a user cannot change the current locale. In fact, an interface exists in both `ios_base` and `basic_streambuf` for changing the locale at

any time. If we were to use `locale::transparent`, the locale could change even if the user didn't want it to. This isn't to say that the user couldn't imbue `locale::transparent`.

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_istringstream,
basic_ostringstream,
basic_ifstream,
basic_ofstream
```

Question: Should we say anything about `str` streams?

Requestor: Nathan Myers (myersn@roguewave.com)
Norihiro Kumagai (kuma@slab.tnr.sharp.co.jp)

Issue Number: 27-907
Title: [io]{pfs|sfx} and exceptions
Section: 27.2.2.1, 27.2.4.1
Status: **active**
Description:

The members `ipfx()`/`opfx` and `isfx()`/`osfx()` of the streams are not compatible with exceptions. We need to eliminate them in favor of member classes whose constructor/destructor perfom the same actions, in the manner of custodian classes.

Possible Resolution:

In order for `istream`/`ostream` to be safe with exceptions the *pfx and *sfx functions need to be called in pairs. I propose to introduce a new class in `basic_istream` and `basic_ostream`. This class will be responsible for "doing" *pfx type operations in the constructor and *sfx type operations in the destructor. This will guarantee that *pfx and *sfx will be called in pairs even if an exception is thrown.

Add the following class to `basic_istream`:

```
class sentry {
      bool ok_; // exposition only
   public:
      explicit sentry(bool noskipws = false);
      ~sentry();

      operator bool();
};
```

Add the following class to `basic_ostream`:

```
class sentry {
      bool ok_; // exposition only
   public:
      explicit sentry();
      ~sentry();
```

```
        operator bool();
};
```

Typical usage will be something like:

```
template<class charT, class traits>
basic_istream<charT, traits>&
basic_istream<charT, traits>::
operator>>(short& s)
{
   if(sentry cerberus(false)) {
      // read in short
   }

   return *this;
}
```

Class `basic_istream::sentry`

The class `sentry` defines a class that is responsible for doing ipfx and isfx type operations. This class makes prefix and suffix operations exception safe.

```
explicit sentry(bool noskipws = false);
```

Effects: *Same as `ipfx()`, except that the return value is stored in* `ok_`.

```
~sentry();
```

Effects: *Same as `isfx()`*

```
operator bool();
```

Effects: Returns `ok_`.

Class `basic_ostream::sentry`

The class `sentry` defines a class that is responsible for doing opfx and osfx type operations. This class makes prefix and suffix operations exception safe.

```
explicit sentry();
```

Effects: *Same as `opfx()`, except that the return value is stored in* `ok_`.

```
~sentry();
```

Effects: *Same as `osfx()`*

```
operator bool();
```

Effects: Returns `ok_`.

Deprecate `ipfx`/`opfx`/`isfx`/`osfx` in favor of this technique.

Requestor: Nathan Myers (myersn@roguewave.com),

John Hinke (jhinke@qds.com),
Jerry Schwarz (jss@declarative.com)

Issue Number: 27-908
Title: iosfwd declarations: incomplete
Section: 27.2 Forward declarations
Status: **active**
Description:

The list of forward declarations is incomplete. Should it contain all of the forward declarations available?

Possible Resolution:
Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-909
Title: Add `iostream`, `fstream`, `stringstream`, and `strstream`
Section: 27
Status: **active**
Description:

These classes were removed from the WP (date unknown). Users will complain about this. Library vendors will probably add this to make their users happy. There has been some discussion of this on comp.std.c++.

Add the classes back to the WP. There is a way around this problem, but it requires users to change more of their code. If at all possible, I think it would be excellent if we could reduce the amount of code that users will have to change.

Without these classes, code such as:

```
fstream   inout("test.txt");
```

Would have to be replaced by code such as:

```
filebuf   fb("test.txt");
istream   in(&fb);
ostream   out(&fb);
```

The problem with this is that there would still be code like:

```
inout << "Something";
inout >> someVar;
```

That would have to be changed and that could be a lot of work.

Possible Resolution:

Option 1:

Add the classes back following the original AT&T implementation.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-910
Title: add a typedef to access the traits parameter for a class.
Section: 27
Status: **active**
Description:

Some classes like `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(cin.getloc().
   template use<ctype<cin.char_type> >()));
```

There is no way to get the traits type without saying something like: `ios_traits<cin.char_type>` which is almost resonable, 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:

Add the following to each templatized class:

```
typedef traits traits_type;
```

Where `traits` is the template parameter

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-911
Title: The example return function for `resetiosflags` is templatized.
Section: 27.6.3 **[lib.std.manip]**
Status: **active**
Description:

The function returned in `resetiosflags` is templatized. It doesn't need to be templatized.

Possible Resolution:

Looks like an editorial issue. Just need to remove the template.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-912
Title: `<ios>` synopsis is incomplete
Section: 27.4 **[lib.iostreams.base]**
Status: **active**
Description:

There are missing typedefs. The missing typedefs are:

```
typedef POS_T streampos;
typedef POS_T wstreampos;
```

Possible Resolution:

Add these typedefs to the synopsis.

Requestor: John Hinke (jhinke@qds.com)

Editorial Boxes

Issue Number: Box 126
Title: Definitions
Section: 27.1.1 [lib.iostreams.definitions]
Status: **active**
Description:

"Move these to an Annex containing a Glossary of terms"

Possible Resolution:

Issue Number: Box 127
Title: Standard iostream objects
Section: 27.3 [lib.iostream.objects]
Status: **active**
Description:

"These objects need to be constructed and associations established before˝dynamic initialization of file scope variables is begun."

Possible Resolution:

Issue Number: Box 128
Title: Narrow stream objects
Section: 27.3.1 [lib.narrow.stream.objects]
Status: **active**
Description:

"The destination of `clog` ought to be implementation defined"

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.

Issue Number: Box 129
Title: Wide stream objects
Section: 27.3.2 [lib.wide.stream.objects]
Status: **active**
Description:

"The destination of `wlog` ought to be implementation defined"

Possible Resolution:

Change 27.3.2 **Wide stream objects [lib.wide.stream.objects]** Paragraph 6 to:

> The object `wlog` controls output to an implementation defined stream buffer.

Issue Number: Box 130
Title: Template `struct ios_traits`
Section: 27.4.2 [lib.ios.traits]
Status: **active**
Description:

"The `newline()` member needs to depend on a `ctype<char_type>` parameter, just as does `is_whitespace()`"

Possible Resolution:
See Issue #27-001

Issue Number: Box 131
Title: `ios_traits` value functions
Section: 27.4.2.2 [lib.ios.traits.values]
Status: **active**
Description:
"Should the argument type be `int_type`"

Possible Resolution:

Issue Number: Box 132
Title: `ios_traits` value functions
Section: 27.4.2.2 [lib.ios.traits.values]
Status: **active**
Description:
"The `newline()` member needs to depend on a `ctype<char_type>` parameter, just as does `is_whitespace()`. As such, we should overload `getline()` with and without the parameter so that the caller need not obtain a `ctype<char_type>` reference to pass to `getline()`."

Possible Resolution:
See Issue #27-001

Issue Number: Box 133
Title: `ios_traits` conversion functions
Section: 27.4.2.4 [lib.ios.traits.convert]
Status: **active**
Description:
"To be specified"

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

Possible Resolution:

Issue Number: Box 134
Title: `ios_traits` conversion functions
Section: 27.4.2.4 [lib.ios.traits.convert]
Status: **active**
Description:
"To be specified"

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

Possible Resolution:

Issue Number: Box 135
Title: Class `ios_base`
Section: 27.4.3 [lib.ios.base]

Status: **active**
Description:

"Add the following declarations:

```
// 27.4.4.3 iostate flags:
operator bool() const;
bool operator!() const;
iostate rdstate() 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 still be a version of `copyfmt()` specified for `basic_ios`. The task of "copying the state" can be divided between these two functions: `ios_base::copyfmt()` copies the current state of the `fmtflags`, while `basic_ios::copyfmt()` copies the `tie()` state (if that is indeed what is involved in copying the *format*)."

Possible Resolution:

Move all of these functions to class `ios_base` except for `clear` and `setstate`.

Issue Number: Box 136
Title: Class `ios_base`
Section: 27.4.3 [lib.ios.base]
Status: **active**
Description:

"Move the following to class `basic_ios`:

```
int_type fill() const;
int_type fill(int_type ch);
```

Possible Resolution:

Issue Number: Box 137
Title: Class `ios_base::Init`
Section: 27.4.3.1.6 [lib.ios::Init]
Status: **active**
Description:

"For the sake of exposition, the maintained data is presented here as:

`static int init_cnt`, counts the number of constructor and destructor calls for class `Init`, initialized to zero."

Possible Resolution:

Issue Number: Box 138
Title: `ios_base` constructors
Section: 27.4.3.5 [lib.ios.base.cons]
Status: **active**
Description:

"The initialization of the value returned by `getloc()` remains an open issue, as described in 95-0026/N0626; it may be changed to, `locale()`, the global locale in effect at the time of initialization."

Possible Resolution:

Issue Number: Box 139
Title: Template class `basic_ios`
Section: 27.4.4 [lib.ios]
Status: **active**
Description:
See Box 135

Possible Resolution:

Issue Number: Box 140
Title: Template class `basic_ios`
Section: 27.4.4 [lib.ios]
Status: **active**
Description:
See Box 136
Possible Resolution:

Issue Number: Box 141
Title: `basic_ios` constructors
Section: 27.4.4.1 [lib.basic.ios.cons]
Status: **active**
Description:
"TBS"

Possible Resolution:

Issue Number: Box 142
Title: `basic_ios` constructors
Section: 27.4.4.1 [lib.basic.ios.cons]
Status: **active**
Description:
"TBS"

Possible Resolution:

Issue Number: Box 143
Title: **Member functions**
Section: **27.4.4.2 [lib.basic.ios.members]**
Status: **active**
Description:
Need to modify so as to describe the occurrence of imbueing `getloc()::codecvt` into the argument stream buffer.

Possible Resolution:

Issue Number: Box 144
Title: **`basic`_streambuf constructors**

Section: **27.5.2.1 [lib.streambuf.cons]**
Status: **active**
Description:

The choice of `locale::classic()` vs. `locale()` for the initial value of `getloc()` remains open: 95-0026/N0626.

Possible Resolution:

The default value of `getloc()` should most likely be the current global locale or `locale()`. The reason for this is that `locale::classic()` seems very constraining for implementations that regularly use a locale other than the "C" locale.

Issue Number: Box 145
Title: Buffer management and positioning
Section: 27.5.2.4.2 [lib.streambuf.virt.buffer]
Status: **active**
Description:

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

Possible Resolution:

I think it is perfectly resonable to be unable to synchronize the input˝sequence all of the time.

Issue Number: Box 146
Title: Get area
Section: 27.5.2.4.3 [lib.streambuf.virt.get]
Status: **active**
Description:

Is this correct? 94-0035/N0422 said: "Returns an estimate of the number of˝characters available in the sequence, or -1. If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.

Possible Resolution:

Issue Number: Box 147
Title: Common requirements
Section: 27.6.1.2.1 [lib.istream.formatted.reqmts]
Status: **active**
Description:

Is this table clear with regards to `%x` vs. `%X`?

Possible Resolution:

Issue Number: Box 148
Title: Common requirements
Section: 27.6.1.2.1 [lib.istream.formatted.reqmts]
Status: **active**
Description:

Can the current `num_put/num_get` facet handle `basefield` specification? Needs more discussion.

Possible Resolution:

Issue Number: Box 149
Title: `basic_ostream` prefix and suffix functions
Section: 27.6.2.3 [lib.ostream.prefix]
Status: **active**
Description:

Need to append the `locale` dependency on appropriate inserters. In particular, descriptions must allow for digit group separators.

Possible Resolution:

Issue Number: Box 150
Title: Common requirements
Section: 27.6.2.4.1 [lib.ostream.formatted.reqmts]
Status: **active**
Description:

Needs work: NTBS

Possible Resolution:

Issue Number: Box 151
Title: Common requirements
Section: 27.6.2.4.1 [lib.ostream.formatted.reqmts]
Status: **active**
Description:

Is this table clear with regards to `%x` vs. `%X`?

Possible Resolution:

Issue Number: Box 152
Title: Common requirements
Section: 27.6.2.4.1 [lib.ostream.formatted.reqmts]
Status: **active**
Description:

Can the current `num_put/num_get` facet handle `basefield` specification? Needs more discussion.

Possible Resolution:

Issue Number: Box 153
Title: Common requirements
Section: 27.6.2.4.1 [lib.ostream.formatted.reqmts]
Status: **active**
Description:

Is this table clear with regards to `%e` vs. `%E`?

Possible Resolution:

Issue Number: Box 154
Title: Template class `basic_stringbuf`
Section: 27.7.1 [lib.stringbuf]
Status: **active**
Description:

For the sake of exposition, the maintained data is presented here as:

`ios_base::openmode mode`, has `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

Possible Resolution:

Issue Number: Box 155
Title: Overridden virtual functions
Section: 27.7.1.3 [lib.stringbuf.virtuals]
Status: **active**
Description:

Check vs. 27.5.2.4 and 27.8.1.4

Possible Resolution:

Issue Number: Box 156
Title: Overridden virtual functions
Section: 27.7.1.3 [lib.stringbuf.virtuals]
Status: **active**
Description:

Check vs. 27.8.1.4

Possible Resolution:

Issue Number: Box 157
Title: Overridden virtual functions
Section: 27.7.1.3 [lib.stringbuf.virtuals]
Status: **active**
Description:

Check vs. 27.8.1.4

Possible Resolution:

Issue Number: Box 158
Title: File streams
Section: 27.8.1 [lib.fstreams]
Status: **active**
Description:

`basic_filebuf<charT, traits>` should be specified so that it treats a file as a sequence of `charT`. Except for `filebuf` and `wfilebuf` that implies it treats the file as binary.

Possible Resolution:

Issue Number: Box 159
Title: Overridden virtual functions
Section: 27.8.1.4 [lib.filebuf.virtuals]
Status: **active**
Description:

Check vs. 27.5.2.4

Possible Resolution:

Issue Number: Box 160

Title: Overridden virtual functions
Section: 27.8.1.4 [lib.filebuf.virtuals]
Status: **active**
Description:

[To Be Filled]

Check vs. 27.5.2.4 and 27.7.1.3

Possible Resolution:

Issue Number: Box 161
Title: Overridden virtual functions
Section: 27.8.1.4 [lib.filebuf.virtuals]
Status: **active**
Description:

The member `get_offstate()` is not defined anywhere.

Possible Resolution:

Issue Number: Box 162
Title: Overridden virtual functions
Section: 27.8.1.4 [lib.filebuf.virtuals]
Status: **active**
Description:

[To Be Filled]

Check vs. 27.5.2.4 and 27.7.1.3

Possible Resolution:

Issue Number: Box 163
Title: Overridden virtual functions
Section: 27.8.1.4 [lib.filebuf.virtuals]
Status: **active**
Description:

[To Be Filled]

Check vs. 27.5.2.4

Possible Resolution:

Closed Issues

Issue Number: 27-101
Title: `ios_base::exceptions(...)`
Section: 27.4.3.2.11
Status: closed
Description:

The `ios_base::exceptions(iostate except_arg)` function calls a member of `basic_ios`, which isn't accessible. The **Effects** clause says: Calls `clear(rdstate())`.

Resolution:

The **Effects** clause was removed.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-301
Title: imbued locales for streambufs mentioned, but missing
Section: 27
Status: closed
Description:

There was mention of an imbued locale for the streambufs, but there was no˝reference to it in the WP.

Possible Resolution:

This has been fixed. There is now functions for imbueing a locale into a˝streambuf.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-302
Title: basic_streambuf::stossc()
Section: 27
Status: closed
Description:

Bernd Eggink says,

> "The function is missing and will break a lot of code."

Resolution:

The function is now deprecated.

Requestor: Bernd Eggink (admin@rrz.uni-hamburg.de)

Issue Number: 27-503
Title: `ostream::operator<< (basic_streambuf&)`
Section: 27
Status: closed
Description:

```
ostream& operator<<(ostream&, basic_streambuf&)
```

should take '`const basic_streambuf&`' rather than '`basic_streambuf&`'

Possible Resolution:

It doesn't make sense to have a `const basic_streambuf&` for this function because the streambuf will change. Therefore, no action was required for this issue.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-603
Title: `istream::operator>>(streambuf &)` `ostream::operator<<(streambuf &)`
Section: 27.2.2.1, 27.2.4.1
Status: closed
Description:

The original IOStreams contained functions:

```
istream& istream::operator>>(streambuf *);
ostream& ostream::operator<<(streambuf *);
```

I see in the current draft that they have been changed to take references˝instead of pointers.

Resolution:

They have been changed back to take pointers. See also 27-601.

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

Issue Number: 27-606
Title: `seekg`, `tellg`, `seekp`, `tellp`
Section: 27.2.2.1
Status: closed
Description:

The following functions were missing from the WP. Here `istream` refers to `basic_istream`, and `ostream` refers to `basic_ostream`.

```
pos_type istream::tellg();
istream& istream::seekg(pos_type&);
istream& istream::seekg(off_type&,
                        ios_type::seekdir);

pos_type ostream::tellp();
ostream& ostream::seekp(pos_type&);
ostream& ostream::seekp(off_type&,
                        ios_type::seekdir);
```

Resolution:

The functions were added to the WP.

Requestor: John Hinke (jhinke@qds.com)
Bernd Eggink (admin@rrz.uni-hamburg.de)

Issue Number: 27-701
Title: `stringstream` constructors
Section: 27
Status: closed
Description:

Bernd Eggink says,

"The effect of `basic_ios::ate` should be mentioned. A behavior corresponding to `basic_filebuf::open()` would be desirable: If the bit `basic_ios::app` is set, insertions take place at the end of the string. Maybe this is a major issue, but I think it should be considered for the sake of consistency."

This should be considered for `basic_strstreambuf`, `basic_ostrstream`, `basic_istrstream`, and the string stream classes.

Possible Resolution:

Requestor: Bernd Eggink (admin@rrz.uni-hamburg.de)

Issue Number: 27-901

Title: Do ios type classes access the streambuf on destruction...

Status: closed

Description:

Angelika Langer asks,

"Is it correct that all classes in the '`ios`' class family which do not provide the stream buffer themselves, i.e. the buffer is externally provided, do not access the stream buffer on˝construction or destruction? This is explicitly stated for `~istream()` and `~ostream()` in the draft and will be added for `~ios()` as well, as far as I've gathered."

Resolution:

No action required. Some classes like `ifstream` and `ofstream` need to access the streambuf on destruction to flush and close the file.

Requestor: Angelika Langer (Angelika.Langer@mch.sni.de)

Issue Number: 27-902

Title: Do streambuf classes access the buffer on destruction...

Status: closed

Description:

Angelika Langer asks,

"Does the same hold for all classes in the '`streambuf`' family in case when the character buffer is provided externally via `setbuf()` or `pubsetbuf()`? (Access on construction is no issue here, as there is no constructor which allows attaching the character buffer. So the question˝is only about access on destruction.)"

Resolution:

Requestor: Angelika Langer (Angelika.Langer@mch.sni.de

Issue Number: 27-903

Title: public typedefs in the interface, static eof/newline functions

Section: 27

Status: closed

Description:

All of the IOStreams classes have a public section that only contains either˝typedefs for abreviation, or helper functions that are not part of the interface. Should these typedefs˝and functions be moved to the private section. An example is:

```
typedef charT   char_type;
static int_type eof()
   { return traits::eof(); }
```

Resolution:

The typedefs are part of the interface. They are not removed. The eof/newline functions are removed. The `eof` function caused problems with the `ios_base::eof()` function. To refer to the eof character, use `traits::eof()`.

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-905
Title: basic_ functions
Section: 27.1
Status: closed
Description:

What is the purpose of the basic_ functions (basic_dec, basic_oct...?) The same purpose can be achieved without using the basic_ prefix.

Resolution:

These functions were removed. The manipulators are now no-longer templatized because ios_base is no longer templatized.

Requestor: John Hinke (jhinke@qds.com)
