IOStreams Issues List
Library Clause 27 R4

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

History

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

Active 27-050 definition for get_state
Active 27-051 definition for get_pos

27.4.3 ios_base

Active 27-150 ios_base::width semantics are incorrect
Active 27-151 proposal for adding ios_base::maxwidth

27.4.4 basic_ios

27.5.2 basic_streambuf

Active 27-350 two return clauses for streambuf::underflow
Active 27-351 streambuf::pbackfail has incorrect Notes: clause
Active 27-352 caching results of calls to locale functions

27.6.1 basic_istream

Active 27-450 confusing english in formatted requirements
Active 27-451 operator>>(char_type *) failure
Active 27-452 operator>>(char_type) failure
Active 27-453 ws manipulator
27.6.2 basic_ostream

Active 27-550 exceptions in ostream
Active 27-551 incorrect conversion specifier for operator<<(unsigned long)

27.6.1-27.6.2 basic_istream, basic_ostream

27.7 string streams

Active 27-750 stringbuf postconditions
Active 27-751 stringbuf::stringbuf constructor
Active 27-752 Incorrect calls to setg and setp in Table 78
Active 27-753 Incorrect calls to setg and setp in table 80

27.8.2 fstreams

Active 27-850 ofstream constructor missing trunc as openmode
Active 27-851 ofstream::open missing trunc in openmode
Active 27-852 filebuf::imbue semantics
Active 27-853 filebuf::seekoff Effects: clause needs work
Active 27-854 filebuf::underflow performance questions
Active 27-855 Editorial fixes in wording for fstreams

Miscellaneous

Active 27-950 The use of specialization
Active 27-951 missing descriptions of specializations
Active 27-952 Editorial changes
Active 27-953 Validity of OFF_T to POS_T conversion
Active 27-954 Question on Table 66 assertions
Active 27-955 destination of clog and wclog

D strstreams

Active 27-1001
**ios_traits issues**

Issue Number: 27-050  
Title: definition for get_state  
Section: 27.4.2.3  
Status: active  
Description:  
The definition of `ios_traits::get_state` is incomplete. Here is the complete description:

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

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

Possible Resolution:

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

---

Issue Number: 27-051  
Title: definition for get_pos  
Section: 27.4.2.3  
Status: active  
Description:  
The definition of `ios_traits::get_pos` is incomplete. Here is the complete description:

```cpp
pos_type get_pos(pos_type 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:

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

---

**ios_base issues**

Issue Number: 27-150  
Title: `ios_base::width` semantics are incorrect  
Section: 27.4.3.2 `ios_base fmtflags state functions` [lib.fmtflags.state]  
Status: active  
Description:  
The current description for `ios_base::width()` is:

**Returns:** The field width (number of characters) to generate on certain output conversions.

It should read "**Returns:** The minimum field width ...."

Possible Resolution:
This is probably too late to make it into the standard (unless the process rolls into further extensive revisions and balloting anyway, which -- judging from the state of the Input/Output library section -- seems likely :->), but I'll point it out it all the same. If we really want programs to use the iostreams package instead of the FILE I/O calls, the iostreams package should provide as a minimum the same facilities as the older library. Specifically, the standard C I/O package provides a convenient method for controlling the maximum number of characters to write in formatted I/O, e.g.:

    fprintf(fp, "FONT NAME: %.16s\n", font_desc.font_name);

This handles the case of a structure which has enough space for a string which will not necessarily be NUL-terminated if the maximum number of characters are stored for the string (a common enough situation when one is manipulating data structures written by someone else's software).

What are the reasons for leaving this out of the iostreams package? Also (while on the topic of rounding out iostreams to match what the competition can do), how difficult would it be to provide the ability to control the (minimum) number of digits in the exponent for a formatted floating point number written using scientific notation (as, for example, one can do in Ada)?

Possible Resolution:

Requestor: Public Comment
Issue Number: 27-351
Title: streambuf::pbackfail has incorrect Notes: clause
Section: 27.5.2.4.4
Status: active
Description: basic_streambuf::pbackfail Notes begins a sentence with “Other calls shall.” Can’t apply “shall” to user program behavior, by the accepted conformance model.

Possible Resolution:

Editorial...

Requestor: Public Comment

---

Issue Number: 27-352
Title: caching results of calls to locale functions
Section: 27.5.2.4.1
Status: active
Description: "Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained." Does this mean that changes in locale can be effectively ignored by the streambuf?

Possible Resolution:

Requestor: Public Comment

---

### basic_istream issues

Issue Number: 27-450
Title: confusing english in formatted requirements
Section: 27
Status: active
Description: 27.6.1.2.1 [lib.stream.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:

Editorial...

Requestor: Public Comment

---

Issue Number: 27-451
Title: operator>>(char_type *) failure
Section: 27.6.1.2.2 [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:

Requestor: Public Comment

Issue Number: 27-452
Title: operator>>(char_type)& failure
Section: 27.6.1.2.2 [lib.istream::extractors]
Status: active
Description: 27.6.1.2.2 [lib.istream::extractors]: Paragraph 2:

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:

Requestor: Public Comment

Issue Number: 27-453
Title: ws manipulator
Section: 27.6.1.4 [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:

Requestor: Public Comment

---

**basic_ostream issues**

Issue Number: 27-550
Title: exceptions in ostream
Section: 27.6.2.4.2
Status: active
Description: 27.6.2.4.2:
basic_ostream::operator<<(basic_streambuf<charT,traits>* sb), last line of Effects paragraph 2 can’t happen. Previous sentence says that if “an exception was thrown while extracting a character, it calls setstate(failbit) (which may throw ios_base::failure).” Then the last sentence says, “If an exception was thrown while extracting a character and failbit is on in exceptions() the caught exception is rethrown.” But in this case, setstate has already thrown ios_base::failure. Besides, I can find no committee resolution that calls for exceptions() to be queried in this event. And an earlier sentence says unconditionally that the exception is rethrown. Last sentence should be struck.

Possible Resolution:

Issue Number: 27-551
Title: incorrect conversion specifier for operator<<(unsigned long)
Section: 27.6.2.4.2 [lib.ostream.inserters]
Status: active
Description:

27.6.2.4.2 [lib.ostream.inserters]:

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:

Issue Number: 27-750
Title: stringbuf postconditions
Section: 27.7.1
Status: active
Description:

27.7.1:

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:

Requestor: Public Comment

X3J16/95-0221  WG21/N0821  7
Title:            stringbuf::stringbuf constructor
Section:         27.7.1
Status:          active
Description:

27.7.1:
basic_stringbuf::basic_stringbuf(basic_string, openmode) Postconditions requires that str() == str. This is true only if which has in set. Condition should be restated.

Possible Resolution:

Requestor:       Public Comment

Issue Number:    27-752
Title:           Incorrect calls to setg and setp in Table 78
Section:         27.7.1
Status:          active
Description:

27.7.1:
Table 78 describes calls to setg and setp with string arguments, for which no signature exists. Needs to be recast.

Possible Resolution:

Requestor:       Public Comment

Issue Number:    27-753
Title:           Incorrect calls to setg and setp in table 80
Section:         27.7.1.2
Status:          active
Description:

27.7.1.2:
Table 80 describes calls to setg and setp with string arguments, for which no signature exists. Needs to be recast.

Possible Resolution:

Requestor:       Public Comment

file stream issues

Issue Number:    27-850
Title:           ofstream constructor missing trunc as openmode
Section:         27.8.1.9
Status:          active
Description:

27.8.1.9:
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:

Requestor: Public Comment

Issue Number: 27-851
Title: ofstream::open missing trunc in openmode
Section: 27.8.1.10
Status: active
Description:

27.8.1.10:
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:

Requestor: Public Comment

Issue Number: 27-852
Title: filebuf::imbue semantics
Section: 27.8.1.4
Status: active
Description:

27.8.1.4:
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:

Requestor: Public Comment

Issue Number: 27-853
Title: filebuf::seekoff Effects: clause needs work
Section: 27.8.1.4
Status: active
Description:

27.8.1.4:
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:

Requestor: Public Comment

Issue Number: 27-854
Title: filebuf::underflow performance questions
Section: 27.8.1.4
Status: active
Description:

27.8.1.4:

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:

Requestor: Public Comment

Issue Number: 27-855
Title: Editorial fixes in wording for fstreams
Section: 27.8.1
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:

Requestor: Public Comment

---

**Miscellaneous issues**

Issue Number: 27-950
Title: The use of specialization
Section: 27
Status: active
Description:

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...”

Requestor: John Hinke (jhinke@qds.com)

Issue Number: 27-951
Title: missing descriptions of specializations
Section: 27
Status: active
Description:
For compatibility, each templatized class has two specializations. One for skinny characters and one for wide characters. For example:

```cpp
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_istringstream`:

27.7.2 Template class `basic_istringstream` [lib.istringstream]
The class `istringstream` is a specialization of the template class `basic_istringstream` specialized by the type `char`.

The class `wistringstream` is a specialization of the template class `basic_istringstream` 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`.

Requestor: John Hinke (jhinke@qds.com)

<table>
<thead>
<tr>
<th>Issue Number</th>
<th>27-952</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Editorial changes</td>
</tr>
<tr>
<td>Section</td>
<td>27.1.2</td>
</tr>
<tr>
<td>Status</td>
<td>active</td>
</tr>
<tr>
<td>Description</td>
<td>27.1.2 [lib.iostreams.type.reqmts]: Last sentence: &quot;... expects to the character container class.&quot; should read &quot;... expects of the character container class.&quot;</td>
</tr>
</tbody>
</table>

Possible Resolution:

Requestor: Public Comment

<table>
<thead>
<tr>
<th>Issue Number</th>
<th>27-953</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Validity of OFF_T to POS_T conversion</td>
</tr>
<tr>
<td>Section</td>
<td>27.1.2.3</td>
</tr>
<tr>
<td>Status</td>
<td>active</td>
</tr>
<tr>
<td>Description</td>
<td>27.1.2.3 [lib.iostreams.off.t]: Paragraph 4: &quot;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.&quot; Of what use is the conversion, then?</td>
</tr>
</tbody>
</table>

Possible Resolution:

Requestor: Public Comment

<table>
<thead>
<tr>
<th>Issue Number</th>
<th>27-954</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Question on Table 66 assertions</td>
</tr>
<tr>
<td>Section</td>
<td>27.1.2.4</td>
</tr>
<tr>
<td>Status</td>
<td>active</td>
</tr>
<tr>
<td>Description</td>
<td>27.1.2.4 [lib.iostreams.pos.t]: table 66: first row has assertion &quot;p == P(i)&quot; but p does not appear in the expression for that row; also, that row has the note &quot;a destructor is assumed&quot; -- what does this mean?</td>
</tr>
</tbody>
</table>

Possible Resolution:

Requestor: Public Comment
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.