WG21/N1097
J16/97-0059
Thomas Plum
1997-07-17

Clause 22 diffs

```
*** lib-locales Mon Jul 14 10:25:58 1997
--- lib-locales.new.txt Thu Jul 17 08:28:30 1997
***************
*** 6,9 ****
--- 6,17 ----
  support for character classification and string collation, numeric,
  monetary, and date/time formatting and parsing, and message retrieval.
+ .\" Germany 22/ lib.localization, actually a change to lib.iterator.requirement
+ .eN
+ Change clause [lib.iterator.requirement] so that is explicitly says
+ that _all_ functions in the library, that take ranges, have to define
+ a reasonable behavior for valid ranges and may exhibit undefined
+ behavior for invalid ranges. So far, the standard states this only
+ for algorithms.
+ .nE
  .P
  The following subclauses describe components for
***************
*** 38,56 ****
    template <class Facet> bool         has_facet(const locale&) throw();
  .Ce
  .Cb
  \f6// subclause _lib.locale.convenience_, convenience interfaces:\fP
!   template <class charT> bool isspace (charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> bool isprint (charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> bool iscntrl (charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> bool isupper (charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> bool islower (charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> bool isalpha (charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> bool isdigit (charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> bool ispunct (charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> bool isxdigit(charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> bool isalnum (charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> bool isgraph (charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> charT toupper(charT \f6c\fP, const locale& \f6loc\fP) const
;
!   template <class charT> charT tolower(charT \f6c\fP, const locale& \f6loc\fP) const
;
  .Ce
  .Cb
--- 46,65 ----
    template <class Facet> bool         has_facet(const locale&) throw();
  .Ce
+ .\" USA CD2-22-07 lib.locales [remove 'const' from non-member is* functions]
  .Cb
  \f6// subclause _lib.locale.convenience_, convenience interfaces:\fP
!   template <class charT> bool isspace (charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> bool isprint (charT \f6c\fP, const locale& \f6loc\fP);
```

```
!   template <class charT> bool iscntrl (charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> bool isupper (charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> bool islower (charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> bool isalpha (charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> bool isdigit (charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> bool ispunct (charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> bool isxdigit(charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> bool isalnum (charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> bool isgraph (charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> charT toupper(charT \f6c\fP, const locale& \f6loc\fP);
!   template <class charT> charT tolower(charT \f6c\fP, const locale& \f6loc\fP);
  .Ce
  .Cb
***************
*** 136,145 ****
      locale(const locale& \f6other\fP, const char* \f6std_name\fP, category);
      template <class Facet> locale(const locale& \f6other\fP, Facet* \f6f\fP);
-     template <class Facet> locale(const locale& \f6other\fP,
-                                   const locale& \f6one\fP);
      locale(const locale& \f6other\fP, const locale& \f6one\fP, category);
     ~locale() throw();  \f6// non-virtual\fP
      const locale& operator=(const locale& \f6other\fP) throw();
  .Ce
  .Cb
    \&\f6// locale operations:\fP\&
--- 145,154 ----
      locale(const locale& \f6other\fP, const char* \f6std_name\fP, category);
      template <class Facet> locale(const locale& \f6other\fP, Facet* \f6f\fP);
      locale(const locale& \f6other\fP, const locale& \f6one\fP, category);
     ~locale() throw();  \f6// non-virtual\fP
      const locale& operator=(const locale& \f6other\fP) throw();
+     template <class Facet> locale combine(const locale& \f6other\fP);
  .Ce
+ .\" USA CD2-22-005 22.1.1 lib.locale [eliminate  template<-> locale( other, one) ]
  .Cb
    \&\f6// locale operations:\fP\&
***************
*** 170,177 ****
  sense, it's just a class interface; at the same time, it's an index
  into a locale's set of facets.
  .P
  Access to the facets of a
  .CW locale
! is via two member function templates,
  .CW use_facet<>
  and
--- 179,189 ----
  sense, it's just a class interface; at the same time, it's an index
  into a locale's set of facets.
+ .\" Germany 22 22. (or lib.locale) ["pure virtual" - decision: no change]
  .P
+ .\" Netherlands _2211/a lib.locale [use_facet and has_facet aren't members]
+ .\" USA CD2-22-008 22.1.1 [duplicate issue]
  Access to the facets of a
  .CW locale
! is via two function templates,
  .CW use_facet<>
  and
***************
*** 188,191 ****
--- 200,205 ----
  .CW ostreambuf_iterator<charT,traits> .
  .Fe
+ .\" Netherlands _2211/b 22.1.1 lib.locale [cerberos needs ctor argument]
+ .\" USA CD2-22-009 22.1.1 lib.locale [duplicate issue]
  .Cb
    template <class charT, class traits>
```

```
***************
*** 193,197 ****
      operator<< (basic_ostream<charT,traits>& s, Date d)
    {
!     typename basic_ostream<charT,traits>::sentry cerberos;
      if (cerberos) {
        ios_base::iostate err = 0;
--- 207,211 ----
      operator<< (basic_ostream<charT,traits>& s, Date d)
    {
!     typename basic_ostream<charT,traits>::sentry cerberos(s);
      if (cerberos) {
        ios_base::iostate err = 0;
***************
*** 352,356 ****
        num_put<char>, num_put<wchar_t>

  _
! time  time_get<char>, time_put<wchar_t>,
        time_put<char>, time_put<wchar_t>

  _
--- 366,370 ----
        num_put<char>, num_put<wchar_t>

  _
! time  time_get<char>, time_get<wchar_t>,
        time_put<char>, time_put<wchar_t>

  _
***************
*** 358,362 ****
  .TE
  .Te
! .P
  For any locale \f6loc\fP
  either constructed, or returned by
--- 372,377 ----
  .TE
  .Te
! .\" Netherlands _221111/ 22.1.1.1.1 [fix time_get<wchar_t> in T52]
! .P
  For any locale \f6loc\fP
  either constructed, or returned by
***************
*** 374,377 ****
--- 389,393 ----
  facet templates identified as members of a category, and for those
  shown in Table \n+(Tn:
+ .\" Germany CD2-22-019(SecondPart) 2 actually 22.1.1.1.1 lib.locale.category
  .Ts "Required Instantiations"
  .na
***************
*** 388,395 ****
  monetary      moneypunct_byname<char,International>,
        moneypunct_byname<wchar_t,International>,
!       money_get<char,InputIterator>,
!       money_get<wchar_t,InputIterator>,
!       money_put<char,OutputIterator>,
!       money_put<wchar_t,OutputIterator>

  _
  numeric       numpunct_byname<char>, numpunct_byname<wchar_t>
--- 404,409 ----
  monetary      moneypunct_byname<char,International>,
        moneypunct_byname<wchar_t,International>,
!       money_get<C,InputIterator>,
!       money_put<C,OutputIterator>

  _
  numeric       numpunct_byname<char>, numpunct_byname<wchar_t>
***************
*** 404,420 ****
```

```
        time_put<wchar_t,OutputIterator>
        time_put_byname<wchar_t,OutputIterator>
  .TE
  .Te
  .P
! For the facets
! .CW num_get<>
  and
! .CW num_put<>
! the implementation provided must depend only on the corresponding facets
! .CW numpunct<>
  and
! .CW ctype<> ,
! instantiated on the same character type.  Other facets are allowed to
! depend on any other facet that is part of a standard category.
  .P
  In declarations of facets, a template formal parameter with name
--- 418,451 ----
        time_put<wchar_t,OutputIterator>
        time_put_byname<wchar_t,OutputIterator>
+ _
+ messages      messages_byname<char>, messages_byname<wchar_t>
  .TE
  .Te
+ .\" Germany 221111/ lib.locale.category [add messages_byname to T53]
  .P
! .\" Germany CD2-22-004 22.1.1.1.1 lib.locale.category
! .\" USA CD2-22-004 22.1.1.1.1 [duplicate issue]
! The provided implementation of members of facets
! .CW num_get<charT>
  and
! .CW num_put<charT>
! calls
! .CW use_facet<F>(l)
! only for facet
! .CW F
! of types
! .CW numpunct<charT>
  and
! .CW ctype<charT>,
! and for locale
! .CW l
! the value obtained
! by calling member
! .CW getloc()
! on the
! .CW ios_base&
! argument to these functions.
! .\" Germany _221111/ 22.1.1.1.1 lib.locale.category [num_get, num_put, message_bynam
e]
! .\" TODO
  .P
  In declarations of facets, a template formal parameter with name
***************
*** 458,462 ****
  derived from
  .CW locale::facet
! and containing a declaration as follows:
  .Cb
        static ::std::locale::id id;
--- 489,501 ----
  derived from
  .CW locale::facet
! .\" USA CD2-22-010 22.1.1.1.2 [mention "public"]
! and containing a publicly-accessible declaration as follows:\*f
! .\" Germany CD2-22-019 22 actually 22.1.1.1.2 lib.locale.facet
! .Fs
```

```
! This is a complete list of requirements;
! there are no other requirements.
! Thus, a facet class need not have a public copy constructor,
! assignment, default constructor, destructor, etc.
! .Fe
  .Cb
        static ::std::locale::id id;
***************
*** 472,483 ****
  argument to the constructor is used for lifetime management.
  .LI
! If
! .CW "(\f6refs\fP == 0)"
! the facet's lifetime is managed by the locale or locales it is
! incorporated into;
! .LI
! if
! .CW "(\f6refs\fP == 1)"
! its lifetime is until explicitly deleted.
  .P
  Constructors of all
--- 511,531 ----
  argument to the constructor is used for lifetime management.
  .LI
! .\" USA CD2-22-011 22.1.1.1.2 lib.locale.facet [revise "refs" wording]
! .\" USA CD2-22-020 22.1.1.1.2 lib.locale.facet [use "destructed" not "delete"?]
! For
! .CW "\f6refs\fP == 0",
! the implementation performs
! .CW delete static_cast<locale::facet*>(f)
! (where
! .CW f
! is a pointer to the facet) when the last
! .CW locale
! object containing the facet is destroyed;
! for
! .CW "\f6refs\fP == 1",
! the implementation never destroys the facet.
! .eN
! This wording may require using "destroyed" instead of "deleted".
! .nE
  .P
  Constructors of all
***************
*** 607,627 ****
  .La Notes:
  The resulting locale has no name.
  .Pb
- template <class Facet> locale(const locale& \f6other\fP, const locale& \f6one\fP);
- .Pe
- .La Effects:
- Constructs a locale incorporating all facets from the first
- argument except that identified by
- .CW Facet ,
- and that facet from the second argument instead.
- .La Throws:
- .CW runtime_error
- if
- .CW "has_facet<\f6Facet\fP>(\f6one\fP)"
- is
- .CW false .
- .La Notes:
- The resulting locale has no name.
- .Pb
  locale(const locale& \f6other\fP, const locale& \f6one\fP, category \f6cats\fP);
  .Pe
--- 655,660 ----
```

```
  .La Notes:
  The resulting locale has no name.
+ .\" USA CD2-22-005 [also affecting 22.1.1.2 lib.locale.cons: delete one ctor]
  .Pb
  locale(const locale& \f6other\fP, const locale& \f6one\fP, category \f6cats\fP);
  .Pe
***************
*** 644,647 ****
--- 677,702 ----
  .H4 "\&\f7locale\fP\& members" lib.locale.members
  .\"
+ .\" USA CD2-22-005 22.1.1  actually 22.1.1.3 lib.locale.members [add combine]
+ .ix "[locale] [combine]"
+ .Pb
+ template <class Facet> locale combine(const locale& \f6other\fP);
+ .Pe
+ .La Effects:
+ Constructs a locale incorporating
+ all facets from
+ .CW *this
+ except for that one facet of
+ .CW other
+ that is identified by
+ .CW Facet .
+ .La Returns:
+ The newly created locale.
+ .La Throws:
+ .CW runtime_error
+ if
+ .CW has_facet<Facet>(other)
+ is false.
+ .La Notes:
+ The resulting locale has no name.
  .ix "[locale] [name]"
  .Pb
***************
*** 652,655 ****
--- 707,719 ----
  .CW *this ,
  if it has one; otherwise, the string \f5"*"\fP.
+ .\" USA CD2-22-012 22.1.1.2 lib.locale.cons
+ If
+ .CW *this
+ has a name, then
+ .CW locale(name())
+ is equivalent to
+ .CW *this.
+ Details of
+ the contents of the resulting string are otherwise implementation-defined.
  .\"----
  .H4 "\&\f7locale\fP\& operators" lib.locale.operators
***************
*** 752,759 ****
  .La Returns:
  a reference to the corresponding facet of \&\f6loc\fP, if present.
  .La Throws:
  .CW bad_cast
  if
! .CW "has_facet<Facet>(*this)"
  is
  .CW false .
--- 816,824 ----
  .La Returns:
  a reference to the corresponding facet of \&\f6loc\fP, if present.
+ .\" USA CD2-22-013 22.1.2 lib.locale.global.templates [change *this to loc]
  .La Throws:
  .CW bad_cast
```

```
  if
! .CW "has_facet<Facet>(loc)"
  is
  .CW false .
***************
*** 842,846 ****
  and
  .CW get() ,
! respectively.  Each such member function takes an
  .CW ios_base&
  .ix "[ios__base] [flags]"
--- 907,914 ----
  and
  .CW get() ,
! respectively.
! .\" Germany CD2-22-029 22 lib.localization [no change]
! .\" Explain why the put & get functions take non-const ios_base& ?
! Each such member function takes an
  .CW ios_base&
  .ix "[ios__base] [flags]"
***************
*** 901,907 ****
      explicit ctype(size_t \f6refs\fP = 0);
  .Ce
  .Cb
      bool         is(mask \f6m\fP, charT \f6c\fP) const;
!     const charT* is(const charT* \f6low\fP, const charT* \f6high\fP, mask* vec) cons
t;
      const charT* scan_is(mask \f6m\fP,
                           const charT* \f6low\fP, const charT* \f6high\fP) const;
--- 969,977 ----
      explicit ctype(size_t \f6refs\fP = 0);
  .Ce
+ .\" editorial font correction on  vec  below
+ .\" USA CD2-22-014 22.2.1.1 lib.locale.ctype [add arg types to toupper prototype (ab
ove)]
  .Cb
      bool         is(mask \f6m\fP, charT \f6c\fP) const;
!     const charT* is(const charT* \f6low\fP, const charT* \f6high\fP, mask* \f6vec\fP
) const;
      const charT* scan_is(mask \f6m\fP,
                           const charT* \f6low\fP, const charT* \f6high\fP) const;
***************
*** 908,912 ****
      const charT* scan_not(mask \f6m\fP,
                            const charT* \f6low\fP, const charT* \f6high\fP) const;
!     charT        toupper(charT) const;
      const charT* toupper(charT* \f6low\fP, const charT* \f6high\fP) const;
      charT        tolower(charT \f6c\fP) const;
--- 978,982 ----
      const charT* scan_not(mask \f6m\fP,
                            const charT* \f6low\fP, const charT* \f6high\fP) const;
!     charT        toupper(charT \f6c\fP) const;
      const charT* toupper(charT* \f6low\fP, const charT* \f6high\fP) const;
      charT        tolower(charT \f6c\fP) const;
***************
*** 956,961 ****
  for character classing during input parsing.
  .P
! The base class implementation implements character classing appropriate
  to the implementation's native character set.
  .\"----
  .H5 "\&\f7ctype\fP\& members" lib.locale.ctype.members
--- 1026,1038 ----
  for character classing during input parsing.
  .P
! The instantiations required in Table 52, namely
```

```
! .CW ctype<char>
! and
! .CW ctype<wchar_t> ,
! implement character classing appropriate
  to the implementation's native character set.
+ .eN
+ Check the table number for "Table 52" hereafter.
+ .nE
  .\"----
  .H5 "\&\f7ctype\fP\& members" lib.locale.ctype.members
***************
*** 1043,1052 ****
  of type
  .CW ctype_base::mask .
! The second form places \f6M\fP for all
  .CW *\f6p\fP
  where
  .CW "(\f6low\fP<=\f6p\fP && \f6p\fP<\f6high\fP)" ,
! into
  .CW "\f6vec\fP[\f6p\fP-\f6low\fP]" .
  .La Returns:
  The first form returns the result of the expression
--- 1120,1132 ----
  of type
  .CW ctype_base::mask .
! The second form identifies a value \f6M\fP of type
! .CW ctype_base::mask
! for each
  .CW *\f6p\fP
  where
  .CW "(\f6low\fP<=\f6p\fP && \f6p\fP<\f6high\fP)" ,
! and places it into
  .CW "\f6vec\fP[\f6p\fP-\f6low\fP]" .
+ .\" Germany CD2-22-025 22.2.1.1.2 lib.locale.ctype.virtuals [clarify do_is]
  .La Returns:
  The first form returns the result of the expression
***************
*** 1300,1309 ****
      static const mask* classic_table() throw();
  .Ce
  .Cb
     ~ctype();  \f6// virtual\fP
!    virtual char        do_toupper(char) const;
     virtual const char* do_toupper(char* \f6low\fP, const char* \f6high\fP) const;
!    virtual char        do_tolower(char) const;
     virtual const char* do_tolower(char* \f6low\fP, const char* \f6high\fP) const;
   };
  }
--- 1380,1400 ----
      static const mask* classic_table() throw();
  .Ce
+ .\" USA CD2-22-001 22.2.1.3 lib.facet.ctype.special [and minor edits]
  .Cb
     ~ctype();  \f6// virtual\fP
!    virtual char        do_toupper(char \f6c\fP) const;
     virtual const char* do_toupper(char* \f6low\fP, const char* \f6high\fP) const;
!    virtual char        do_tolower(char \f6c\fP) const;
     virtual const char* do_tolower(char* \f6low\fP, const char* \f6high\fP) const;
+
+    virtual char        do_widen(char \f6c\fP) const;
+    virtual const char* do_widen(const char* \f6low\fP,
+                                 const char* \f6high\fP,
+                                 char* \f6to\fP) const;
+    virtual char        do_narrow(char \f6c\fP, char \f6dfault\fP) const;
+    virtual const char* do_narrow(const char* \f6low\fP,
+                                  const char* \f6high\fP,
+                                  char \f6dfault\fP, char* \f6to\fP) const;
```

```
+
    };
  }
***************
*** 1444,1454 ****
      char* \f6to\fP) const;
  .Pe
! .La Effects:
! .CW "::memcpy(\f6to\fP, \f6low\fP, \f6high\fP-\f6low\fP)"
! .ix "[memcpy]"
  .La Returns:
! .CW c
! or
! .CW hi
  .\"----
  .ix "[ctype<char>] [narrow]"
--- 1535,1542 ----
      char* \f6to\fP) const;
  .Pe
! .\" USA CD2-22-001 22.2.1.3.2 lib.facet.ctype.char.members
  .La Returns:
! .CW "do_widen(\f6low\fP, \f6high\fP, \f6to\fP)" .
! .ix "[do_widen]"
  .\"----
  .ix "[ctype<char>] [narrow]"
***************
*** 1458,1480 ****
                      char /*dfault*/, char* \f6to\fP) const;
  .Pe
- .La Effects:
- .CW "::memcpy(\f6to\fP, \f6low\fP, \f6high\fP-\f6low\fP)"
- .ix "[memcpy]"
  .La Returns:
! \f6c\fP or \f6high\fP.
! .eN
! The library WG feels that the members
! .CW widen " and"
! .CW narrow
! should delegate to virtual members
! .CW do_widen
! and
! .CW do_narrow ,
! as in the
! .CW ctype<>
! template, to permit
! .CW char
! encodings that differ from the basic execution character encoding.
! .nE
  .\"----
  .ix "[ctype<char>] [table]"
--- 1546,1552 ----
                      char /*dfault*/, char* \f6to\fP) const;
  .Pe
  .La Returns:
! .CW "do_narrow(\f6low\fP, \f6high\fP, \f6to\fP)" .
! .ix "[do_narrow]"
  .\"----
  .ix "[ctype<char>] [table]"
***************
*** 1498,1501 ****
--- 1570,1574 ----
  .\"===
  .H5 "\&\f7ctype<char>\fP\& virtual functions" lib.facet.ctype.char.virtuals
+ .\" USA CD2-22-001 22.2.1.3.4 lib.facet.ctype.char.virtuals
  .Pb
  char        do_toupper(char) const;
***************
```

```
*** 1503,1506 ****
--- 1576,1588 ----
  char        do_tolower(char) const;
  const char* do_tolower(char* low, const char* high) const;
+
+ virtual char        do_widen(char \f6c\fP) const;
+ virtual const char* do_widen(const char* \f6low\fP,
+                              const char* \f6high\fP,
+                              char* \f6to\fP) const;
+ virtual char        do_narrow(char \f6c\fP, char \f6dfault\fP) const;
+ virtual const char* do_narrow(const char* \f6low\fP,
+                               const char* \f6high\fP,
+                               char \f6dfault\fP, char* \f6to\fP) const;
  .Pe
  These functions are described identically as those members of the
***************
*** 1518,1525 ****
    protected:
     ~ctype_byname();  \f6// virtual\fP
!    virtual char        do_toupper(char) const;
     virtual const char* do_toupper(char* \f6low\fP, const char* \f6high\fP) const;
!    virtual char        do_tolower(char) const;
     virtual const char* do_tolower(char* \f6low\fP, const char* \f6high\fP) const;
    };
  }
--- 1600,1616 ----
    protected:
     ~ctype_byname();  \f6// virtual\fP
!    virtual char        do_toupper(char \f6c\fP) const;
     virtual const char* do_toupper(char* \f6low\fP, const char* \f6high\fP) const;
!    virtual char        do_tolower(char \f6c\fP) const;
     virtual const char* do_tolower(char* \f6low\fP, const char* \f6high\fP) const;
+
+    virtual char        do_widen(char \f6c\fP) const;
+    virtual const char* do_widen(char* \f6low\fP,
+                                 const char* \f6high\fP,
+                                 char* \f6to\fP) const;
+    virtual char        do_widen(char \f6c\fP) const;
+    virtual const char* do_widen(char* \f6low\fP,
+                                 const char* \f6high\fP) const;
+
    };
  }
***************
*** 1568,1576 ****
       const internT* \f6from\fP, const internT* \f6from_end\fP, const internT*& \f6f
rom_next\fP,
             externT* \f6to\fP,         externT* \f6to_limit\fP,       externT*& \f6t
o_next\fP) const;
-    virtual result do_unshift(stateT& \f6state\fP,
-             externT* \f6to\fP,         externT* \f6to_limit\fP,       externT*& \f6t
o_next\fP) const;
     virtual result do_in(stateT& \f6state\fP,
       const externT* \f6from\fP, const externT* \f6from_end\fP, const externT*& \f6f
rom_next\fP,
             internT* \f6to\fP,         internT* \f6to_limit\fP,       internT*& \f6t
o_next\fP) const;
     virtual int do_encoding() const throw();
     virtual bool do_always_noconv() const throw();
--- 1659,1667 ----
       const internT* \f6from\fP, const internT* \f6from_end\fP, const internT*& \f6f
rom_next\fP,
             externT* \f6to\fP,         externT* \f6to_limit\fP,       externT*& \f6t
o_next\fP) const;
     virtual result do_in(stateT& \f6state\fP,
       const externT* \f6from\fP, const externT* \f6from_end\fP, const externT*& \f6f
rom_next\fP,
```

```
              internT* \f6to\fP,         internT* \f6to_limit\fP,       internT*& \f6t
o_next\fP) const;
+     virtual result do_unshift(stateT& \f6state\fP,
+             externT* \f6to\fP,         externT* \f6to_limit\fP,       externT*& \f6t
o_next\fP) const;
      virtual int do_encoding() const throw();
      virtual bool do_always_noconv() const throw();
***************
*** 1593,1610 ****
  argument selects the pair of codesets being mapped between.
  .P
! Implementations are required to provide instantiations for
! .CW <wchar_t,char,mbstate_t>
  and
! .CW <char,char,mbstate_t> .
! The base class instance of the latter implements a degenerate conversion:
! its member
! .CW always_noconv()
! returns
! .CW true
! and
! .CW max_length()
! returns
! .CW 1 .
! .\" "The following sentences are from (N0699,22-015)."
  Instantiations on
  .CW mbstate_t
--- 1684,1698 ----
  argument selects the pair of codesets being mapped between.
  .P
! .\" Germany 22-019(SecondPart) 22 actually 22.2.1.5 lib.locale.codecvt
! The instantiations required in Table 52, namely
! .CW codecvt<wchar_t,char,mbstate_t>
  and
! .CW codecvt<char,char,mbstate_t> ,
! convert the implementation-defined native character set.
! .CW codecvt<char,char,mbstate_t>
! implements a degenerate conversion;
! it does not convert at all.
! .CW codecvt<wchar_t,char,mbstate_t>
! converts between the native character sets for tiny and wide characters.
  Instantiations on
  .CW mbstate_t
***************
*** 1617,1622 ****
  the specialized
  .CW do_convert
! member.  The base class implementations convert the
! implementation-defined native execution codeset.
  .\"----
  .H5 "\&\f7codecvt\fP\& members" lib.locale.codecvt.members
--- 1705,1709 ----
  the specialized
  .CW do_convert
! member.
  .\"----
  .H5 "\&\f7codecvt\fP\& members" lib.locale.codecvt.members
***************
*** 1766,1770 ****
  .CW stateT()
  .Fe
! The base class implementation stores no characters.
  .La "Returns"
  An enumeration value, as summarized in Table \n+(Tn:
--- 1853,1868 ----
  .CW stateT()
  .Fe
```

```
! The instantiations required in Table 52, namely
! .CW codecvt<wchar_t,char,mbstate_t>
! and
! .CW codecvt<char,char,mbstate_t> ,
! store no characters.
! .\" Germany CD2-22-023 lib.locale.codecvt.virtuals
! .\" USA CD2-22-023 [duplicate issue]
! Stores no more than
! .CW "(\f6to_limit\fP-\f6to\fP)
! destination elements.
! It always leaves the \&\f6to_next\fP\& pointer
! pointing one beyond the last element successfully stored.
  .La "Returns"
  An enumeration value, as summarized in Table \n+(Tn:
***************
*** 1783,1787 ****
  .TE
  .Te
! The base class implementation returns
  .CW noconv .
  .\"---
--- 1881,1886 ----
  .TE
  .Te
! .CW codecvt<char,char,mbstate_t> ,
! returns
  .CW noconv .
  .\"---
***************
*** 1813,1823 ****
  returns
  .CW noconv
! for all valid argument values.  The base class implementation
! for the instantiation
! .CW <char,char,mbstate_t>
  returns
! .CW true ;
! others return
! .CW false .
  .\"----
  .\"
--- 1912,1921 ----
  returns
  .CW noconv
! for all valid argument values.
! .CW codecvt<char,char,mbstate_t> ,
  returns
! .CW true .
! .\" Germany CD2-33-024 22.2.1.5 lib.locale.codecvt
! .\" USA Cd2-22-024 [duplicate issue]
  .\"----
  .\"
***************
*** 1845,1849 ****
  or fewer valid complete characters of type
  .CW \f6internT\fP .
! The base class implementation returns the lesser of
  .CW \f6max\fP
  and
--- 1943,1951 ----
  or fewer valid complete characters of type
  .CW \f6internT\fP .
! The instantiations required in Table 52, namely
! .CW codecvt<wchar_t,char,mbstate_t>
! and
! .CW codecvt<char,char,mbstate_t> ,
! return the lesser of
```

```
  .CW \f6max\fP
  and
***************
*** 1864,1867 ****
--- 1966,1971 ----
  value
  .CW \f6state\fP .
+ .CW codecvt<char,char,mbstate_t>
+ returns 1.
  .\"
  .\"---
***************
*** 1868,1871 ****
--- 1972,1976 ----
  .H4 "Template class \&\f7codecvt_byname\fP\&" lib.locale.codecvt.byname
  .ix "[codecvt__byname]"
+ .\" USA CD2-22-006 22.2.1.6 lib.locale.codecvt.byname [add do_unshift]
  .Cb
  namespace std {
***************
*** 1882,1885 ****
--- 1987,1992 ----
        const externT* \f6from\fP, const externT* \f6from_end\fP, const externT*& \f6f
rom_next\fP,
              internT* \f6to\fP,         internT* \f6to_limit\fP,       internT*& \f6t
o_next\fP) const;
+     virtual result do_unshift(stateT& \f6state\fP,
+             externT* \f6to\fP,         externT* \f6to_limit\fP,       externT*& \f6t
o_next\fP) const;
      virtual int do_encoding() const throw();
      virtual bool do_always_noconv() const throw();
***************
*** 1886,1889 ****
--- 1993,1998 ----
      virtual int do_length(const stateT&, const externT* \f6from\fP, const externT* \
f6end\fP,
                            size_t \f6max\fP) const;
+  virtual result do_unshift(stateT& \f6state\fP,
+          externT* \f6to\fP, externT* \f6to_limit\fP, externT*& \f6to_next\fP) const;
      virtual int do_max_length() const throw();
    };
***************
*** 1892,1895 ****
--- 2001,2005 ----
  .\"===
  .H3 "The numeric category" lib.category.numeric
+ .\" Germany _2222/ 22.2.2 lib.category.numeric [C and C++ locales - No changes]
  .P
  The classes
***************
*** 1909,1913 ****
  .Fe
  .P
! The base class implementation refers to the
  .CW ios_base&
  argument for formatting specifications (_lib.locale.categories_),
--- 2019,2037 ----
  .Fe
  .P
! .\" Germany CD2-22-019(SecondPart) 22 actually 22.2.2 lib.category.numeric
! All specifications of member functions for
! num_put
! and
! num_get
! in the subclauses of _lib.category.numeric_ only apply to the
! instantiations required in Table 52 and Table 53, namely
! .CW num_get<char> ,
```

```
! .CW num_get<wchar_t> ,
! .CW num_get<C,InputIterator> ,
! .CW num_put<char> ,
! .CW num_put<wchar_t> ,
! and
! .CW num_put<C,OutputIterator> .
! These instantiations refer to the
  .CW ios_base&
  argument for formatting specifications (_lib.locale.categories_),
***************
*** 2042,2050 ****
  .CW "use_facet< ctype<charT> >(\f6loc\fP)" ,
  and
! .CW "use_facet< numpunct<charT> >(\f6loc\fP)" .
  If an error occurs, \f6val\fP
  is unchanged; otherwise it is set to the resulting value.
  .P
! The details of this operation occur in two stages
  .LI
  Stage 1: Determine a conversion specifier
--- 2166,2179 ----
  .CW "use_facet< ctype<charT> >(\f6loc\fP)" ,
  and
! .CW "use_facet< numpunct<charT> >(\f6loc\fP)" ,
! where
! .I loc
! is
! .CW \f6str\fP.getloc() .
  If an error occurs, \f6val\fP
  is unchanged; otherwise it is set to the resulting value.
  .P
! .\" USA CD2-22-015 22.2.2.1.2 lib.facet.num.get.virtuals ["three" stages]
! The details of this operation occur in three stages
  .LI
  Stage 1: Determine a conversion specifier
***************
*** 2059,2063 ****
  .br
  The details of the stages are presented below.
! .IR in .
  .La "Stage 1:"
  The function initializes local variables via
--- 2188,2192 ----
  .br
  The details of the stages are presented below.
! .I in .
  .La "Stage 1:"
  The function initializes local variables via
***************
*** 2066,2077 ****
      fmtflags basefield = (flags & ios_base::basefield);
      fmtflags uppercase = (flags & ios_base::uppercase);
-     fmtflags basefield = (flags & ios_base::basefield);
      fmtflags boolalpha = (flags & ios_base::boolalpha);
  .Ce
  .La ""
  For conversion to an integral type, the
! function determines the integral conversion specifier as indicated in Table \n+(Tn:
! .Ts "Integer conversions"
  .na
  .TS
  box center;
--- 2195,2210 ----
      fmtflags basefield = (flags & ios_base::basefield);
      fmtflags uppercase = (flags & ios_base::uppercase);
      fmtflags boolalpha = (flags & ios_base::boolalpha);
  .Ce
```

```
+ .\" Germany 1 Editorial 22.2.2.1.2 lib.facet.num.get.virtuals ["basefield" twice?]
  .La ""
+ .\" Germany _222212/(TheSecondOne) 22.2.2.1.2 lib.facet.num.get.virtuals [re Table 5
6]
+ .\" Germany _222212/d 22.2.2.1.2 lib.facet.num.get.virtuals [resolved by /TheSecondO
ne]
  For conversion to an integral type, the
! function determines the integral conversion specifier as indicated in Table \n+(Tn.
! The table is ordered.
! That is, the first line whose condition is true applies.
  .na
+ .\" Germany _222212/b 22.2.2.1.2 lib.facet.num.get.virtuals [redundant %x in table 5
6]
  .TS
  box center;
***************
*** 2082,2087 ****
  basefield == oct      %o
  _
- basefield == hex && !uppercase        %x
- _
  basefield == hex      %X
  _
--- 2215,2218 ----
***************
*** 2132,2135 ****
--- 2263,2270 ----
  .CW charT
  is taken from \f6in\fP and local variables are initialized as if by
+ .\" USA CD2-22-002 22.2.2.2.1.2 lib.facet.num.get.virtuals [revise without 'narrow']
+ .eN
+ Re-write without "narrow", as per 22-002 - NCM.
+ .nE
  .Cb
      char_type ct = *in ;
***************
*** 2150,2154 ****
  returned by stage 1. If so it is accumulated.
  .La ""
! If the character is neither discarded nor accumulated then \f6in\fP
  is advanced by
  .CW ++in
--- 2285,2291 ----
  returned by stage 1. If so it is accumulated.
  .La ""
! .\" Germany _222212/(TheFirstOne) 22.2.2.1.2 lib.facet.num.get.virtuals ["either/or"
]
! .\" USA CD2-22-016 22.2.2.1.2 [duplicate issue]
! If the character is either discarded or accumulated then \f6in\fP
  is advanced by
  .CW ++in
***************
*** 2161,2165 ****
  has been accumulated in stage 2 that is converted (according to the
  rules of
! .CW std::scanf)
  to a value of the type of \f6val\fP.  This value is
  stored in \f6val\fP\& and
--- 2298,2302 ----
  has been accumulated in stage 2 that is converted (according to the
  rules of
! .CW scanf)
  to a value of the type of \f6val\fP.  This value is
  stored in \f6val\fP\& and
***************
*** 2171,2183 ****
  accumulated in stage 2 would have caused scanf to report an input failure.
```

```
  .CW ios_base::failbit
! is assigned to
! .IR err .
  .br
  Digit grouping is checked.  That is, the positions of discarded
  separators is examined for consistency with
! .CW "use_facet<numpunct<charT> >(\f6loc\fP).grouping()"
! .eN
! Is the treatment of separators here clear?
! .nE
  If they are not consistent then
  .CW ios_base::failbit
--- 2308,2319 ----
  accumulated in stage 2 would have caused scanf to report an input failure.
  .CW ios_base::failbit
! is assigned to
! .I err .
  .br
+ .\" Germany _222212/c 22.2.2.1.2 lib.facet.num.get.virtuals [re-edit the "grouping"]
+ .P
  Digit grouping is checked.  That is, the positions of discarded
  separators is examined for consistency with
! .CW "use_facet<numpunct<charT> >(\f6loc\fP).grouping()" .
  If they are not consistent then
  .CW ios_base::failbit
***************
*** 2194,2201 ****
  .Pe
  .La Effects:
  If
  .CW "(\f6str\f7.flags()&&ios_base::boolalpha)==0"
! then input proceeds as it would for an
! .CW int
  except that if a value is being stored into \f6val\fP,
  the value is determined according to the following:
--- 2330,2338 ----
  .Pe
  .La Effects:
+ .\" Germany _222212/(TheThirdOne) 22.2.2.1.2 lib.facet.num.get.virtuals [int->long]
  If
  .CW "(\f6str\f7.flags()&&ios_base::boolalpha)==0"
! then input proceeds as it would for a
! .CW long
  except that if a value is being stored into \f6val\fP,
  the value is determined according to the following:
***************
*** 2298,2303 ****
--- 2435,2443 ----
  .H5 "\&\f7num_put\fP\& members" lib.facet.num.put.members
  .ix "[num__put] [put]"
+ .\" Germany _222221/ 22.2.2.2.1 lib.facet.num.put.members [add put for bool]
  .Pb
  iter_type put(iter_type \f6out\fP, ios_base& \f6str\fP, char_type \f6fill\fP,
+               bool \f6val\fP) const;
+ iter_type put(iter_type \f6out\fP, ios_base& \f6str\fP, char_type \f6fill\fP,
                long \f6val\fP) const;
  iter_type put(iter_type \f6out\fP, ios_base& \f6str\fP, char_type \f6fill\fP,
***************
*** 2336,2341 ****
      locale \f6loc\fP = \f6str\fP.getloc();
  .Ce
  .Pa
! The base class implementation is described in several stages
  .LI
  Stage 1:
--- 2476,2482 ----
      locale \f6loc\fP = \f6str\fP.getloc();
```

```
  .Ce
+ .\" Germany CD2-22-019(SecondPart) 22 actually 22.2.2.2.2 lib.facet.num.put.virtuals
  .Pa
! The details of this operation occur in several stages:
  .LI
  Stage 1:
***************
*** 2345,2349 ****
  given this conversion specifier for
  .Cb
!     std::printf(\&\f6spec\fP, \f6val\fP\&)
  .Ce
  assuming that the current locale is
--- 2486,2490 ----
  given this conversion specifier for
  .Cb
!     printf(\&\f6spec\fP, \f6val\fP\&)
  .Ce
  assuming that the current locale is
***************
*** 2372,2377 ****
  The first action of stage 1 is to determine a conversion specifier.
  The tables that describe this determination use the following local variables
  .Cb
!     fmtflags flags = this->flags() ;
      fmtflags basefield =  (flags & (ios_base::basefield));
      fmtflags uppercase =  (flags & (ios_base::uppercase));
--- 2513,2520 ----
  The first action of stage 1 is to determine a conversion specifier.
  The tables that describe this determination use the following local variables
+ .\" Germany _222222/a 22.2.2.2.2 lib.facet.num.put.virtuals [use str.flags()]
+ .\" Netherlands _222222/ [duplicate issue]
  .Cb
!     fmtflags flags = str.flags() ;
      fmtflags basefield =  (flags & (ios_base::basefield));
      fmtflags uppercase =  (flags & (ios_base::uppercase));
***************
*** 2382,2386 ****
  .La ""
  All tables used in describing stage 1 are ordered.
! That is. the first line whose condition is true applies.
  A line without a condition is the default behavior when none of the earlier
  lines apply.
--- 2525,2530 ----
  .La ""
  All tables used in describing stage 1 are ordered.
! .\" trivial edit, the next comma was a period
! That is, the first line whose condition is true applies.
  A line without a condition is the default behavior when none of the earlier
  lines apply.
***************
*** 2435,2438 ****
--- 2579,2583 ----
  type a length modifier is added to the
  conversion specifier as indicated in  Table \n+(Tn.
+ .\" Germany _222222/f 22.2.2.2.2 lib.facet.num.put.virtuals [remove "short" in table
]
  .Ts "Length modifier"
  .na
***************
*** 2443,2450 ****
  type  length modifier
  =
- short h
- _
- unsigned short        h
- _
```

```
  long  l
  _
--- 2588,2591 ----
***************
*** 2469,2473 ****
  =
  T{
! an integral type other than a character type
  T}    flags & showpos +
  \^    flags & showbase        #
--- 2610,2614 ----
  =
  T{
! an integral type
  T}    flags & showpos +
  \^    flags & showbase        #
***************
*** 2478,2488 ****
  .ad
  .Te
  .La ""
  For conversion from a floating-point type, if
! .CW "(flags() & fixed) != 0"
  or if
! .CW "precision() > 0" ,
  then
! .CW precision()
  is specified in the conversion specification.
  .La ""
--- 2619,2630 ----
  .ad
  .Te
+ .\" Germany _222222/g 22.2.2.2.2 lib.facet.num.put.virtuals [drop "other than char"]
  .La ""
  For conversion from a floating-point type, if
! .CW "(flags & fixed) != 0"
  or if
! .CW "\f6str\fP.precision() > 0" ,
  then
! .CW \f6str\fP.precision()
  is specified in the conversion specification.
  .La ""
***************
*** 2499,2506 ****
  .sp
  .La "Stage 2:"
  Any character \f6c\fP other than a decimal point(.) is converted to a
  .CW charT
  via
! .CW "charT(\&\f6c\fP\&)"
  .P
  A local variable \f6punct\fP is initialized via
--- 2641,2649 ----
  .sp
  .La "Stage 2:"
+ .\" Germany _222222/e 22.2.2.2.2 lib.facet.num.put.virtuals [use "widen" not "charT"
]
  Any character \f6c\fP other than a decimal point(.) is converted to a
  .CW charT
  via
! .CW "use_facet<ctype<charT> >(\f6loc\fP).widen(\&\f6c\fP\&)"
  .P
  A local variable \f6punct\fP is initialized via
***************
*** 2525,2529 ****
  .Ce
  .La ""
```

```
! The location of any padding is determined according to Table \n+(Tn:
  .\" X3J16/95-0149==WG21/N0749
  .Ts "Fill padding"
--- 2668,2681 ----
  .Ce
  .La ""
! The location of any padding\*f is determined according to Table \n+(Tn:
! .Fs
! The conversion specification
! .CW #o
! generates a leading
! .CW 0
! which is
! .I not
! a padding character.
! .Fe
  .\" X3J16/95-0149==WG21/N0749
  .Ts "Fill padding"
***************
*** 2550,2577 ****
  T}    pad after x or X
  _
! \f6otherwise\fP       pad before stage 2 sequence
  .TE
  .ad
  .Te
! .Fs
! The conversion specification
! .CW #o
! generates a leading
! .CW 0
! which is
! .I not
! a padding character.
! .Fe
  .La ""
! .CW width()
  is nonzero and the number of
  .CW charT 's
! in the sequence after stage 3 is less than
! .CW width() ,
  then enough \f6fill\fP characters are added to the sequence at the position
  indicated for padding to bring the length of the sequence to
! .CW width() .
  .La ""
! .CW width(0)
  is called.
  .sp
--- 2702,2723 ----
  T}    pad after x or X
  _
! \f6otherwise\fP       pad before
  .TE
  .ad
  .Te
! .\" Germany _222222/c 22.2.2.2.2 lib.facet.num.put.virtuals [drop "stage 2 sequence"
]
  .La ""
! .\" Germany _222222/b 22.2.2.2.2 lib.facet.num.put.virtuals ["If", and "stage 2"]
! If
! .CW \f6str\fP.width()
  is nonzero and the number of
  .CW charT 's
! in the sequence after stage 2 is less than
! .CW \f6str\fP.width() ,
  then enough \f6fill\fP characters are added to the sequence at the position
  indicated for padding to bring the length of the sequence to
```

```
! .CW \f6str\fP.width() .
  .La ""
! .CW \f6str\fP.width(0)
  is called.
  .sp
***************
*** 2583,2589 ****
      *\f6out\fP++ = c
  .Ce
! If at any point
! .CW out.failed()
! becomes true, then output is terminated.
  .Pb
  iter_type put(iter_type \f6out\fP, ios_base& \f6str\fP, char_type \f6fill\fP,
--- 2729,2733 ----
      *\f6out\fP++ = c
  .Ce
! .\" USA CD2-22-003 22.2.2.2 lib.facet.num.put.virtuals [out.failed() is undefined]
  .Pb
  iter_type put(iter_type \f6out\fP, ios_base& \f6str\fP, char_type \f6fill\fP,
***************
*** 2594,2599 ****
  .CW (\f6str\fP.flags()&ios_base::boolalpha)==0
  then do
  .Cb
!     \f6out\fP = put(\f6out\fP\&, \f6str\fP\&, \f6fill\fP\&, (int)\f6val\fP\&)
  .Ce
  Otherwise do
--- 2738,2744 ----
  .CW (\f6str\fP.flags()&ios_base::boolalpha)==0
  then do
+ .\" Germany _222222/d 22.2.2.2.2 lib.facet.num.put.virtuals ["put" becomes "do_put"]
  .Cb
!     \f6out\fP = do_put(\f6out\fP\&, \f6str\fP\&, \f6fill\fP\&, (int)\f6val\fP\&)
  .Ce
  Otherwise do
***************
*** 2604,2608 ****
  .Ce
  and then insert the characters of \f6s\fP into \f6out\fP.
! .IR out .
  .\"===
  .H3 "The numeric punctuation facet" lib.facet.numpunct
--- 2749,2753 ----
  .Ce
  and then insert the characters of \f6s\fP into \f6out\fP.
! .I out .
  .\"===
  .H3 "The numeric punctuation facet" lib.facet.numpunct
***************
*** 2645,2653 ****
  .CW numpunct<>
  specifies numeric punctuation.
! The base class provides classic
  .CW C \('' \(``
! numeric formats, while the
! .CW _byname \('' \(``...
! version supports named locale (e.g. POSIX, X/Open) numeric formatting semantics.
  .P
  The syntax for number formats is as follows, where
--- 2790,2806 ----
  .CW numpunct<>
  specifies numeric punctuation.
! .\" Germany CD2-22-019(SecondPart) 22 actually 22.2.3.1 lib.locale.numpunct
! The instantiations required in Table 52, namely
! .CW numpunct<wchar_t>
! and
```

```
! .CW numpunct<char> ,
! provide classic
  .CW C \('' \(``
! numeric formats,
! i.e. they contain information equivalent to that contained in the
! .CW C \('' \(``
! locale or their wide character counterparts as if obtained by
! a call to
! .CW widen.
  .P
  The syntax for number formats is as follows, where
***************
*** 2737,2741 ****
  .La Returns:
  A character for use as the decimal radix separator.
! The base class implementation returns \f5'.'\fP.
  .\"-----
  .ix "[numpunct] [do__thousands__sep]"
--- 2890,2896 ----
  .La Returns:
  A character for use as the decimal radix separator.
! .\" USA CD2-22-017 22.2.3.1.2 lib.facet.numpunct.virtuals [values might be wide]
! .\" Germany CD2-22-019(SecondPart) 2 actually 22.2.3.1.2 lib.facet.numpunct.virtuals
! The required instantiations return \f5'.'\fP or \f5L'.'\fP.
  .\"-----
  .ix "[numpunct] [do__thousands__sep]"
***************
*** 2745,2749 ****
  .La Returns:
  A character for use as the digit group separator.
! The base class implementation returns \f5',\fP.
  .\"----
  .ix "[numpunct] [do__grouping]"
--- 2900,2904 ----
  .La Returns:
  A character for use as the digit group separator.
! The required instantiations return \f5',\fP or \f5L',\fP.
  .\"----
  .ix "[numpunct] [do__grouping]"
***************
*** 2757,2762 ****
  represents the number of digits\*f
  .Fs
  Thus, the string \f5"\e003"\fP specifies groups of 3 digits each, and
! \f5"3"\fP probably indicates groups of 51 (!) digits each.
  .Fe
  in the group at position \f6i\fP, starting with position 0 as the
--- 2912,2921 ----
  represents the number of digits\*f
  .Fs
+ .\"
  Thus, the string \f5"\e003"\fP specifies groups of 3 digits each, and
! \f5"3"\fP probably indicates groups of 51 (!) digits each,
! because 51 is the ASCII value of \f5"3"\fP.
! .\" Germany CD2-22-032 22.2.3.1.2 lib.facet.numpunct.virtuals [explain "51"]
! .\" USA CD2-22-032 [duplicate issue]
  .Fe
  in the group at position \f6i\fP, starting with position 0 as the
***************
*** 2770,2774 ****
  the size of the digit group is unlimited.
  .br
! The base class implementation returns the empty string, indicating
  no grouping.
  .\"-----
--- 2929,2934 ----
  the size of the digit group is unlimited.
```

```
  .br
! .\" Germany CD2-22-019(SecondPart) 2 actually 22.2.3.1.2 lib.facet.numpunct.virtuals
! The required instantiations return the empty string, indicating
  no grouping.
  .\"-----
***************
*** 2786,2790 ****
  .br
  In the base class implementation these names are
! \f5"true"\fP and \f5"false"\fP.
  .\"----
  .H4 "Template class \&\f7numpunct_byname\fP\&" lib.locale.numpunct.byname
--- 2946,2950 ----
  .br
  In the base class implementation these names are
! \f5"true"\fP and \f5"false"\fP, or \f5L"true"\fP and \f5L"false"\fP.
  .\"----
  .H4 "Template class \&\f7numpunct_byname\fP\&" lib.locale.numpunct.byname
***************
*** 2851,2855 ****
  uses the collate facet to allow a locale to act directly as the predicate
  argument for standard algorithms (_lib.algorithms_) and containers operating on stri
ngs.
! The base class implementation applies lexicographic ordering (_lib.alg.lex.compariso
n_).
  .P
  Each function compares a string of characters
--- 3011,3019 ----
  uses the collate facet to allow a locale to act directly as the predicate
  argument for standard algorithms (_lib.algorithms_) and containers operating on stri
ngs.
! The instantiations required in Table 52, namely
! .CW collate<char>
! and
! .CW collate<wchar_t> ,
! apply lexicographic ordering (_lib.alg.lex.comparison_).
  .P
  Each function compares a string of characters
***************
*** 2893,2897 ****
  if the first string is greater than the second,
  .CW -1
! if less, zero otherwise.  The base class implementation implements
  a lexicographical comparison (_lib.alg.lex.comparison_).
  .\"
--- 3057,3067 ----
  if the first string is greater than the second,
  .CW -1
! if less, zero otherwise.
! .\" Germany CD2-22-019(SecondPart) 22 actually  22.2.4.1 lib.locale.collate
! The instantiations required in Table 52, namely
! .CW collate<char>
! and
! .CW collate<wchar_t> ,
! implement
  a lexicographical comparison (_lib.alg.lex.comparison_).
  .\"
***************
*** 2952,2955 ****
--- 3122,3147 ----
  .CW time_put<charT,OutputIterator>
  provide date and time formatting and parsing.
+ .\" Germany _222/5 22.2
+ .eN
+ The effect of the ios_base argument to member functions of time_put
+ and time_get is unspecified. Do they have any effect at all? Which
+ format flags are ignored? Or is it undefined, or
```

```
+ implementation-dependent?
+
+ Actually, the same is true for the monetary facets. For instance, do
+ the dec, hex, oct flags have any effect on the formatting of monetary
+ amounts, or is it implementation-dependent whether they have an
+ effect?
+
+ Date/time and money formats are
+ implementation-defined.
+ .nE
+ .\" Germany CD2-22-019(SecondPart) 22 actually 22.2.5 lib.category.time
+ All specifications of member functions for
+ time_put
+ and
+ time_get
+ in the subclauses of _lib.category.time_ only apply to the
+ instantiations required in Table 52 and Table 53.
  Their members use their
  .CW ios_base& ,
***************
*** 3254,3257 ****
--- 3446,3454 ----
  .CW ctype<>::narrow()
  to identify format specifiers.
+ .\" Germany CD2-22-034 2.2.5.3.1 lib.locale.time.put.members
+ .\" USA CD2-22-034 [duplicate issue]
+ .eN
+ Add wording as per 22-034 - NCM.
+ .nE
  .N[
  This implies that if
***************
*** 3314,3321 ****
  local or international monetary formats are to be used.
  .P
! .CW money_get<>
  and
! .CW money_put<>
! members use their
  .CW ios_base& ,
  .CW ios_base::iostate& ,
--- 3511,3522 ----
  local or international monetary formats are to be used.
  .P
! .\" Germany CD2-22-019(SecondPart) 22 actually 22.2.2 lib.category.numeric
! All specifications of member functions for
! .CW money_put
  and
! .CW money_get
! in the subclauses of _lib.category.monetary_ only apply to the
! instantiations required in Table 52 and Table 53.
! Their members use their
  .CW ios_base& ,
  .CW ios_base::iostate& ,
***************
*** 3327,3330 ****
--- 3528,3533 ----
  .CW ctype<>
  facets, to determine formatting details.
+ .\" Germany CD2-22-026 22.2.6 lib.category.monetary [decision: no change]
+ .\" USA CD2-22-026 [duplicate issue, no change]
  .\"----
  .H4 "Template class \&\f7money_get\fP\&" lib.locale.money.get
***************
*** 3351,3356 ****
                    string_type& \f6digits\fP) const;
  .Ce
```

```
  .Cb
-     static const bool intl = Intl;
      static locale::id id;
  .Ce
--- 3554,3561 ----
                    string_type& \f6digits\fP) const;
  .Ce
+ .\" Germany CD2-22-028 22.2.6, actually 22.2.6.1, lib.category.monetary
+ .\" Germany _2226/ [additional discussion]
+ .\" [There was no Intl template parm for the static.]
  .Cb
      static locale::id id;
  .Ce
***************
*** 3358,3362 ****
    protected:
     ~money_get();  \f6// virtual\fP
!     virtual iter_type do_get(iter_type, , iter_type, bool, ios_base&,
                              ios_base::iostate& \f6err\fP, long double& \f6units\fP)
 const;
      virtual iter_type do_get(iter_type, iter_type, bool, ios_base&,
--- 3563,3567 ----
    protected:
     ~money_get();  \f6// virtual\fP
!     virtual iter_type do_get(iter_type, iter_type, bool, ios_base&,
                              ios_base::iostate& \f6err\fP, long double& \f6units\fP)
 const;
      virtual iter_type do_get(iter_type, iter_type, bool, ios_base&,
***************
*** 3455,3460 ****
  An iterator pointing immediately beyond the last character recognized
  as part of a valid monetary quantity.
  .eN
! The description above needs further review.
  .nE
  .\"----
--- 3660,3666 ----
  An iterator pointing immediately beyond the last character recognized
  as part of a valid monetary quantity.
+ .\" Germany CD2-22-031 22.2.6.1.2 lib.locale.money.get.virtuals [more about errors]
  .eN
! Incorporate discussion of error state as per 22-031 - NCM.
  .nE
  .\"----
***************
*** 3462,3467 ****
  .ix "[money_put]"
  .Cb
  namespace std {
!   template <class charT, bool Intl = false,
              class OutputIterator = ostreambuf_iterator<charT> >
    class money_put : public locale::facet {
--- 3668,3674 ----
  .ix "[money_put]"
  .Cb
+ .\" Germany _2226/ 22.2.6 actually 22.2.6.2 [drop Intl]
  namespace std {
!   template <class charT,
              class OutputIterator = ostreambuf_iterator<charT> >
    class money_put : public locale::facet {
***************
*** 3522,3526 ****
  .CW moneypunct<charT,true>
  or
! .CW moneypunch<charT,false>
  facet of \f6loc\fP (depending on whether \f6intl\fP is
  .CW true
```

```
--- 3729,3733 ----
  .CW moneypunct<charT,true>
  or
! .CW moneypunct<charT,false>
  facet of \f6loc\fP (depending on whether \f6intl\fP is
  .CW true
***************
*** 3536,3543 ****
  .La Notes:
  The currency symbol is generated only if
! .CW "(\f6str\fP.flags() & ios_type::showbase)"
  is true.
  If
! .CW "((\f6str\fP.flags() & ios_type::adjustfield) == ios_type::internal)"
  the fill characters are placed where
  .CW none
--- 3743,3751 ----
  .La Notes:
  The currency symbol is generated only if
! .\" Germany _222622/ 22.2.6.2.2. lib.locale.money.put.virtuals
! .CW "(\f6str\fP.flags() & ios_base::showbase)"
  is true.
  If
! .CW "((\f6str\fP.flags() & ios_base::adjustfield) == ios_base::internal)"
  the fill characters are placed where
  .CW none
***************
*** 3622,3625 ****
--- 3830,3845 ----
  decimal point is exactly the value returned by
  .CW frac_digits .
+ .\" Germany CD2-22-021 22.2.6.3 lib.locale.moneypunct [more about money_base::part]
+ .eN
+ More description of money_base::part as per 22-021 - NCM.
+ .nE
+ .\" Germany CD2-22-022 22.2.6.3 lib.locale.moneypunct
+ .eN
+ More about why "pattern" is "char", as per 22-022 - NCM.
+ .nE
+ .\" Germany CD2-22-030 22.2.6.4, really 22.2.6.3, lib.locale.moneypunct
+ .nE
+ More about "space" and "none", as per 22-030 - NCM.
+ .nE
  .\"----
  .H5 "\&\f7moneypunct\fP\& members" lib.locale.moneypunct.members
***************
*** 3715,3718 ****
--- 3935,3939 ----
  int do_frac_digits() const;
  .Pe
+ .\" Germany _222632/ 22.2.6.3.2 lib.locale.moneypunct.virtuals [no changes]
  .La Returns:
  The number of digits after the decimal radix separator, if any.\*f
***************
*** 3746,3754 ****
  if present, is neither first nor last.
  Otherwise, the elements may appear in any order.
! The base class implementation returns an object of type
  .CW pattern
  initialized to
! .CW "{ symbol, sign, none, value }" ;
! this value is also returned for all international instantiations.\*f
  .Fs
  Note that the international symbol returned by
--- 3967,3981 ----
  if present, is neither first nor last.
  Otherwise, the elements may appear in any order.
```

```
! .\" Germany CD2-22-019(SecondPart) 22 actually 22.2.6.3.2 lib.locale.moneypunct.virt
uals
! The instantiations required in Table 52, namely
! .CW moneypunct<char> ,
! .CW moneypunct<wchar_t> ,
! .CW moneypunct<char,true> ,
! and
! .CW moneypunct<wchar_t,true> ,
! return an object of type
  .CW pattern
  initialized to
! .CW "{ symbol, sign, none, value }" .\*f
  .Fs
  Note that the international symbol returned by
***************
*** 3879,3882 ****
--- 4106,4110 ----
  is used for character set code conversion when retrieving
  messages, if needed.
+ .\" Germany _222712/a 22.2.7.1.2 lib.locale.messages.virtuals [no changes]
  .\"
  .ix "[messages] [do__get]"
***************
*** 4054,4057 ****
--- 4282,4287 ----
    }
  .Ce
+ .\" Netherlands _2211/c 22.1.2 actually 22.2.8 [cerberos needs ctor argument]
+ .\" USA CD2-22-009 22.1.1 lib.locale [duplicate issue]
  .Cb
    std::istream& operator>>(std::istream& s, Date& d)
***************
*** 4058,4062 ****
    {
      using namespace std;
!     istream::sentry cerberos;
      if (cerberos) {
        ios_base::iostate err = goodbit;
--- 4288,4292 ----
    {
      using namespace std;
!     istream::sentry cerberos(s);
      if (cerberos) {
        ios_base::iostate err = goodbit;
```