Doc No: N1029/95-0211
Date: Nov 12, 1996
Project: Programming Language C++
Reply to: Jerry Schwarz
          jerry@intrinsa.com

Proposed Changes to Clause 22 (locales)

```
*** lib-locales   Tue Nov 12 10:59:49 1996
--- lib-loc Tue Nov 12 23:48:00 1996


========  Add Allocator parameter to basic_string

***************
*** 151,158 ****
  .Ce
  .Cb
      template <class charT,Traits>
!       bool operator()(const basic_string<charT,Traits>& s1,
!                       const basic_string<charT,Traits>& s2) const;
  .Ce
  .Cb
    \&\f6// global locale objects:\fP\&
--- 151,158 ----
  .Ce
  .Cb
      template <class charT,Traits>
!       bool operator()(const basic_string<charT,Traits,Allocator>& s1,
!                       const basic_string<charT,Traits,Allocator>& s2)
const;
  .Ce
  .Cb
    \&\f6// global locale objects:\fP\&

=========== add all of execution set to characters known to  =========
=========== widen and narrow                                 =========

***************
*** 1151,1162 ****
  .CW char
  values to the corresponding
  .CW charT
! value or values.
  The only characters for which unique transformations are required
! are the digits, alphabetic characters,
! .CW '-' ,
! .CW '+' ,
! newline, and space.
  .br
  For any named
  .CW ctype
--- 1151,1165 ----
  .CW char
  values to the corresponding
  .CW charT
! value or values.\f*
! .Fs
! The char argument of
! .CW do_widen
! is intended to accept values derived from character literals for
conversion
```

```
! the locale's encoding.
! .Fe
  The only characters for which unique transformations are required
! are those in the basic source character set (_lex.charset_).
  .br
  For any named
  .CW ctype
***************
*** 1197,1207 ****
  values to the corresponding
  .CW char
  value or values.
! The only characters for which unique transformations are required
! are the digits, alphabetic characters,
! .CW '-' ,
! .CW '+' ,
! newline, and space.
  .br
  For any named
  .CW ctype
--- 1200,1211 ----
  values to the corresponding
  .CW char
  value or values.
! .br
! For any character \f6c\fP in the basic source character
set(_lex.charset_)
! the transformation is such that
! .Cb
! do_widen(do_narrow(c),d) == c
! .Ce
  .br
  For any named
  .CW ctype

==========  Add unshift to codecvt ==================

***************
*** 1541,1546 ****
--- 1545,1552 ----
      result out(stateT& \f6state\fP,
        const internT* \f6from\fP, const internT* \f6from_end\fP, const
internT*& \f6from_next\fP,
              externT*   \f6to\fP,       externT* \f6to_limit\fP,
externT*& \f6to_next\fP) const;
+     result unshift(stateT& \f6state\fP,
+            externT*   \f6to\fP,       externT* \f6to_limit\fP,
externT*& \f6to_next\fP) const;
      result in(stateT& \f6state\fP,
        const externT* \f6from\fP, const externT* \f6from_end\fP, const
externT*& \f6from_next\fP,
              internT*   \f6to\fP,       internT* \f6to_limit\fP,
internT*& \f6to_next\fP) const;
***************
*** 1559,1564 ****
--- 1565,1572 ----
      virtual result do_out(stateT& \f6state\fP,
        const internT* \f6from\fP, const internT* \f6from_end\fP, const
internT*& \f6from_next\fP,
              externT* \f6to\fP,         externT* \f6to_limit\fP,
externT*& \f6to_next\fP) const;
+     virtual result unshift(stateT& \f6state\fP,
+             externT* \f6to\fP,         externT* \f6to_limit\fP,
externT*& \f6to_next\fP) const;
```

```
      virtual result do_in(stateT& \f6state\fP,
        const externT* \f6from\fP, const externT* \f6from_end\fP, const
externT*& \f6from_next\fP,
              internT* \f6to\fP,        internT* \f6to_limit\fP,
internT*& \f6to_next\fP) const;
***************
*** 1620,1625 ****
--- 1628,1641 ----
  .La Returns:
  .CW "do_out(\f6state\fP, \f6from\fP,\f6from_end\fP,\f6from_next\fP,
\f6to\fP,\f6to_limit\fP,\f6to_next\fP)"
  .\"
+ .ix "[codecvt] [unshift]"
+ .Pb
+ result unshift(stateT& \f6state\fP,
+        externT* \f6to\fP, externT* \f6to_limit\fP, externT*&
\f6to_next\fP) const;
+ .Pe
+ .La Returns:
+ .CW "do_unshift(\f6state\fP, \f6to\fP,\f6to_limit\fP,\f6to_next\fP)"
+ .\"
  .ix "[codecvt] [in]"
  .Pb
  result in(stateT& \f6state\fP,
***************
*** 1732,1737 ****
--- 1748,1783 ----
  indicates that either the destination sequence has not absorbed all the
  available destination elements, or that additional source elements are
  needed before another destination element can be produced.
+ .\"---
+ .ix "[codecvt] [do__unshift]"
+ .Pb
+ result do_unshift(stateT& \f6state\fP,
+   externT* \f6to\fP, externT* \f6to_limit\fP, externT*& \f6to_next\fP)
const;
+ .Pe
+ .La "Effects"
+ Places characters starting at \f6to\fP that should be appended
+ to terminate a sequence when the current
+ .CW stateT
+ is given by \f6state\fP.\*f
+ .Fs
+ Typically these will be characters to return the state to
+ .CW stateT()
+ .Fe
+ .La "Returns"
+ An enumeration value, as summarized in Table \n+(Tn:
+ .Ts "\f7convert result\fP values"
+ .TS
+ box center;
+ Cf3 Cf3.
+ Value     Meaning
+ =
+ .T&
+ Lf5 Lf1.
+ ok  completed the sequence
+ partial   more characters need to be supplied to complete termination
+ noconv    no termination is needed for this \&\f5state_type\fP\&
+ .TE
+ .Te
  .\"---
  .\"
  .ix "[codecvt] [do__encoding]"
```