

# **Linux Standard Base C++ Specification for IA32 3.1**

## **Linux Standard Base C++ Specification for IA32 3.1**

Copyright © 2004, 2005, 2006 Free Standards Group

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Portions of the text are copyrighted by the following parties:

- The Regents of the University of California
- Free Software Foundation
- Ian F. Darwin
- Paul Vixie
- BSDI (now Wind River)
- Andrew G Morgan
- Jean-loup Gailly and Mark Adler
- Massachusetts Institute of Technology

These excerpts are being used in accordance with their respective licenses.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

UNIX is a registered trademark of The Open Group.

LSB is a trademark of the Free Standards Group in the United States and other countries.

AMD is a trademark of Advanced Micro Devices, Inc.

Intel and Itanium are registered trademarks and Intel386 is a trademark of Intel Corporation.

PowerPC is a registered trademark and PowerPC Architecture is a trademark of the IBM Corporation.

S/390 is a registered trademark of the IBM Corporation.

OpenGL is a registered trademark of Silicon Graphics, Inc.

# Contents

<b>I Introductory Elements .....</b>	<b>1</b>
1 Scope.....	1
1.1 General.....	1
1.2 Module Specific Scope.....	1
2 Normative References.....	2
3 Requirements .....	3
3.1 Relevant Libraries .....	3
3.2 LSB Implementation Conformance .....	3
3.3 LSB Application Conformance.....	4
4 Definitions .....	5
5 Terminology .....	6
6 Documentation Conventions .....	8
<b>II Base Libraries.....</b>	<b>9</b>
7 Libraries .....	10
7.1 Interfaces for libstdcxx.....	10
7.2 Interface Definitions for libstdcxx.....	142
<b>A GNU Free Documentation License (Informative).....</b>	<b>143</b>
A.1 PREAMBLE .....	143
A.2 APPLICABILITY AND DEFINITIONS .....	143
A.3 VERBATIM COPYING .....	144
A.4 COPYING IN QUANTITY .....	144
A.5 MODIFICATIONS.....	145
A.6 COMBINING DOCUMENTS .....	146
A.7 COLLECTIONS OF DOCUMENTS .....	147
A.8 AGGREGATION WITH INDEPENDENT WORKS .....	147
A.9 TRANSLATION.....	147
A.10 TERMINATION.....	147
A.11 FUTURE REVISIONS OF THIS LICENSE .....	148
A.12 How to use this License for your documents .....	148

## List of Tables

2-1 Normative References .....	2
3-1 Standard Library Names.....	3
7-1 libstdcxx Definition.....	10
7-2 libstdcxx - C++ Runtime Support Function Interfaces .....	10
7-3 Primary vtable for <code>type_info</code> .....	11
7-4 <code>typeinfo</code> for <code>type_info</code> .....	11
7-5 Primary vtable for <code>__cxxabiv1::__enum_type_info</code> .....	11
7-6 <code>typeinfo</code> for <code>__cxxabiv1::__enum_type_info</code> .....	12
7-7 Primary vtable for <code>__cxxabiv1::__array_type_info</code> .....	12
7-8 <code>typeinfo</code> for <code>__cxxabiv1::__array_type_info</code> .....	13
7-9 Primary vtable for <code>__cxxabiv1::__class_type_info</code> .....	13
7-10 <code>typeinfo</code> for <code>__cxxabiv1::__class_type_info</code> .....	14
7-11 libstdcxx - Class <code>__cxxabiv1::__class_type_info</code> Function Interfaces.....	14
7-12 Primary vtable for <code>__cxxabiv1::__ibase_type_info</code> .....	14
7-13 <code>typeinfo</code> for <code>__cxxabiv1::__ibase_type_info</code> .....	15
7-14 Primary vtable for <code>__cxxabiv1::__pointer_type_info</code> .....	15
7-15 <code>typeinfo</code> for <code>__cxxabiv1::__pointer_type_info</code> .....	16
7-16 Primary vtable for <code>__cxxabiv1::__function_type_info</code> .....	16
7-17 <code>typeinfo</code> for <code>__cxxabiv1::__function_type_info</code> .....	17
7-18 Primary vtable for <code>__cxxabiv1::__si_class_type_info</code> .....	17
7-19 <code>typeinfo</code> for <code>__cxxabiv1::__si_class_type_info</code> .....	18
7-20 libstdcxx - Class <code>__cxxabiv1::__si_class_type_info</code> Function Interfaces .....	18
7-21 Primary vtable for <code>__cxxabiv1::__vmi_class_type_info</code> .....	18
7-22 <code>typeinfo</code> for <code>__cxxabiv1::__vmi_class_type_info</code> .....	19
7-23 libstdcxx - Class <code>__cxxabiv1::__vmi_class_type_info</code> Function Interfaces...	20
7-24 Primary vtable for <code>__cxxabiv1::__fundamental_type_info</code> .....	20
7-25 <code>typeinfo</code> for <code>__cxxabiv1::__fundamental_type_info</code> .....	20
7-26 Primary vtable for <code>__cxxabiv1::__pointer_to_member_type_info</code> .....	21
7-27 <code>typeinfo</code> for <code>__cxxabiv1::__pointer_to_member_type_info</code> .....	21
7-28 Primary vtable for <code>__gnu_cxx::stdio_sync_filebuf&lt;char, char_traits&lt;char&gt; &gt;</code> ..	22
7-29 Primary vtable for <code>__gnu_cxx::stdio_sync_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;</code> ..	23
7-30 libstdcxx - Class <code>__gnu_cxx::__pool_alloc_base</code> Function Interfaces.....	24
7-31 Primary vtable for <code>exception</code> .....	24
7-32 <code>typeinfo</code> for <code>exception</code> .....	25
7-33 Primary vtable for <code>bad_typeid</code> .....	25
7-34 <code>typeinfo</code> for <code>bad_typeid</code> .....	25
7-35 Primary vtable for <code>logic_error</code> .....	26
7-36 <code>typeinfo</code> for <code>logic_error</code> .....	26
7-37 Primary vtable for <code>range_error</code> .....	26
7-38 <code>typeinfo</code> for <code>range_error</code> .....	26
7-39 Primary vtable for <code>domain_error</code> .....	27
7-40 <code>typeinfo</code> for <code>domain_error</code> .....	27
7-41 Primary vtable for <code>length_error</code> .....	27
7-42 <code>typeinfo</code> for <code>length_error</code> .....	28
7-43 Primary vtable for <code>out_of_range</code> .....	28
7-44 <code>typeinfo</code> for <code>out_of_range</code> .....	28
7-45 Primary vtable for <code>bad_exception</code> .....	28
7-46 <code>typeinfo</code> for <code>bad_exception</code> .....	29
7-47 Primary vtable for <code>runtime_error</code> .....	29
7-48 <code>typeinfo</code> for <code>runtime_error</code> .....	29

7-49 Primary vtable for overflow_error .....	30
7-50 typeid for overflow_error .....	30
7-51 Primary vtable for underflow_error .....	30
7-52 typeid for underflow_error .....	30
7-53 Primary vtable for invalid_argument .....	31
7-54 typeid for invalid_argument .....	31
7-55 Primary vtable for bad_cast .....	31
7-56 typeid for bad_cast .....	32
7-57 Primary vtable for bad_alloc .....	32
7-58 typeid for bad_alloc .....	32
7-59 typeid for ctype_base .....	35
7-60 Primary vtable for __ctype_abstract_base<char> .....	35
7-61 Primary vtable for __ctype_abstract_base<wchar_t> .....	36
7-62 Primary vtable for ctype<char> .....	37
7-63 libstdcxx - Class ctype<char> Function Interfaces .....	38
7-64 Primary vtable for ctype<wchar_t> .....	38
7-65 typeid for ctype<wchar_t> .....	39
7-66 libstdcxx - Class ctype<wchar_t> Function Interfaces .....	39
7-67 Primary vtable for ctype_byname<char> .....	39
7-68 typeid for ctype_byname<char> .....	40
7-69 libstdcxx - Class ctype_byname<char> Function Interfaces .....	40
7-70 libstdcxx - Class ctype_byname<wchar_t> Function Interfaces .....	40
7-71 libstdcxx - Class basic_string<char, char_traits<char>, allocator<char> > Function Interfaces .....	41
7-72 libstdcxx - Class basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces .....	45
7-73 Primary vtable for basic_stringstream<char, char_traits<char>, allocator<char> > .....	50
7-74 Secondary vtable for basic_stringstream<char, char_traits<char>, allocator<char> > .....	50
7-75 Secondary vtable for basic_stringstream<char, char_traits<char>, allocator<char> > .....	51
7-76 VTT for basic_stringstream<char, char_traits<char>, allocator<char> > ....	51
7-77 libstdcxx - Class basic_stringstream<char, char_traits<char>, allocator<char> > Function Interfaces .....	51
7-78 Primary vtable for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	52
7-79 Secondary vtable for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	52
7-80 Secondary vtable for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	53
7-81 VTT for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	53
7-82 libstdcxx - Class basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces .....	53
7-83 Primary vtable for basic_istringstream<char, char_traits<char>, allocator<char> > .....	54
7-84 Secondary vtable for basic_istringstream<char, char_traits<char>, allocator<char> > .....	54
7-85 VTT for basic_istringstream<char, char_traits<char>, allocator<char> > ...	54
7-86 libstdcxx - Class basic_istringstream<char, char_traits<char>, allocator<char> > Function Interfaces .....	55
7-87 Primary vtable for basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	55

7-88	Secondary vtable for basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	55
7-89	VTT for basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	56
7-90	libstdcxx - Class basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces.....	56
7-91	Primary vtable for basic_ostringstream<char, char_traits<char>, allocator<char> >.....	56
7-92	Secondary vtable for basic_ostringstream<char, char_traits<char>, allocator<char> >.....	57
7-93	VTT for basic_ostringstream<char, char_traits<char>, allocator<char> > ..	57
7-94	libstdcxx - Class basic_ostringstream<char, char_traits<char>, allocator<char> > Function Interfaces .....	57
7-95	Primary vtable for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	58
7-96	Secondary vtable for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	58
7-97	VTT for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	59
7-98	libstdcxx - Class basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces.....	59
7-99	Primary vtable for basic_stringbuf<char, char_traits<char>, allocator<char> >.....	59
7-100	typeinfo for basic_stringbuf<char, char_traits<char>, allocator<char> > ..	60
7-101	libstdcxx - Class basic_stringbuf<char, char_traits<char>, allocator<char> > Function Interfaces.....	61
7-102	Primary vtable for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	61
7-103	typeinfo for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > .....	62
7-104	libstdcxx - Class basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces.....	63
7-105	Primary vtable for basic_iostream<char, char_traits<char> >.....	63
7-106	Secondary vtable for basic_iostream<char, char_traits<char> > .....	63
7-107	Secondary vtable for basic_iostream<char, char_traits<char> > .....	63
7-108	VTT for basic_iostream<char, char_traits<char> > .....	64
7-109	libstdcxx - Class basic_iostream<char, char_traits<char> > Function Interfaces.....	64
7-110	Primary vtable for basic_iostream<wchar_t, char_traits<wchar_t> > .....	64
7-111	Secondary vtable for basic_iostream<wchar_t, char_traits<wchar_t> > ...	65
7-112	Secondary vtable for basic_iostream<wchar_t, char_traits<wchar_t> > ...	65
7-113	VTT for basic_iostream<wchar_t, char_traits<wchar_t> > .....	65
7-114	libstdcxx - Class basic_iostream<wchar_t, char_traits<wchar_t> > Function Interfaces .....	66
7-115	Primary vtable for basic_istream<char, char_traits<char> > .....	66
7-116	Secondary vtable for basic_istream<char, char_traits<char> > .....	66
7-117	VTT for basic_istream<char, char_traits<char> > .....	67
7-118	libstdcxx - Class basic_istream<char, char_traits<char> > Function Interfaces.....	67
7-119	Primary vtable for basic_istream<wchar_t, char_traits<wchar_t> >.....	68
7-120	Secondary vtable for basic_istream<wchar_t, char_traits<wchar_t> > .....	68
7-121	VTT for basic_istream<wchar_t, char_traits<wchar_t> > .....	68
7-122	libstdcxx - Class basic_istream<wchar_t, char_traits<wchar_t> > Function Interfaces.....	68

7-123 Primary vtable for basic_ostream<char, char_traits<char> >.....	69
7-124 Secondary vtable for basic_ostream<char, char_traits<char> >.....	70
7-125 VTT for basic_ostream<char, char_traits<char> >.....	70
7-126 libstdcxx - Class basic_ostream<char, char_traits<char> > Function	
Interfaces.....	70
7-127 Primary vtable for basic_ostream<wchar_t, char_traits<wchar_t> >.....	71
7-128 Secondary vtable for basic_ostream<wchar_t, char_traits<wchar_t> >.....	71
7-129 VTT for basic_ostream<wchar_t, char_traits<wchar_t> >.....	71
7-130 libstdcxx - Class basic_ostream<wchar_t, char_traits<wchar_t> > Function	
Interfaces.....	72
7-131 Primary vtable for basic_fstream<char, char_traits<char> >.....	72
7-132 Secondary vtable for basic_fstream<char, char_traits<char> >.....	72
7-133 Secondary vtable for basic_fstream<char, char_traits<char> >.....	73
7-134 VTT for basic_fstream<char, char_traits<char> >.....	73
7-135 libstdcxx - Class basic_fstream<char, char_traits<char> > Function	
Interfaces.....	73
7-136 Primary vtable for basic_fstream<wchar_t, char_traits<wchar_t> >.....	73
7-137 Secondary vtable for basic_fstream<wchar_t, char_traits<wchar_t> >.....	74
7-138 Secondary vtable for basic_fstream<wchar_t, char_traits<wchar_t> >.....	74
7-139 VTT for basic_fstream<wchar_t, char_traits<wchar_t> >.....	74
7-140 libstdcxx - Class basic_fstream<wchar_t, char_traits<wchar_t> > Function	
Interfaces.....	75
7-141 Primary vtable for basic_ifstream<char, char_traits<char> >.....	75
7-142 Secondary vtable for basic_ifstream<char, char_traits<char> >.....	75
7-143 VTT for basic_ifstream<char, char_traits<char> >.....	76
7-144 libstdcxx - Class basic_ifstream<char, char_traits<char> > Function	
Interfaces.....	76
7-145 Primary vtable for basic_ifstream<wchar_t, char_traits<wchar_t> >.....	76
7-146 Secondary vtable for basic_ifstream<wchar_t, char_traits<wchar_t> >.....	77
7-147 VTT for basic_ifstream<wchar_t, char_traits<wchar_t> >.....	77
7-148 libstdcxx - Class basic_ifstream<wchar_t, char_traits<wchar_t> >	
Function Interfaces.....	77
7-149 Primary vtable for basic_ofstream<char, char_traits<char> >.....	77
7-150 Secondary vtable for basic_ofstream<char, char_traits<char> >.....	78
7-151 VTT for basic_ofstream<char, char_traits<char> >.....	78
7-152 libstdcxx - Class basic_ofstream<char, char_traits<char> > Function	
Interfaces.....	78
7-153 Primary vtable for basic_ofstream<wchar_t, char_traits<wchar_t> >.....	79
7-154 Secondary vtable for basic_ofstream<wchar_t, char_traits<wchar_t> >.....	79
7-155 VTT for basic_ofstream<wchar_t, char_traits<wchar_t> >.....	79
7-156 libstdcxx - Class basic_ofstream<wchar_t, char_traits<wchar_t> >	
Function Interfaces.....	80
7-157 Primary vtable for basic_streambuf<char, char_traits<char> >.....	80
7-158 typeid for basic_streambuf<char, char_traits<char> >.....	81
7-159 libstdcxx - Class basic_streambuf<char, char_traits<char> > Function	
Interfaces.....	81
7-160 Primary vtable for basic_streambuf<wchar_t, char_traits<wchar_t> >.....	82
7-161 typeid for basic_streambuf<wchar_t, char_traits<wchar_t> >.....	83
7-162 libstdcxx - Class basic_streambuf<wchar_t, char_traits<wchar_t> >	
Function Interfaces.....	83
7-163 Primary vtable for basic_filebuf<char, char_traits<char> >.....	84
7-164 typeid for basic_filebuf<char, char_traits<char> >.....	85
7-165 libstdcxx - Class basic_filebuf<char, char_traits<char> > Function	
Interfaces.....	85

7-166	Primary vtable for <code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;</code> .....	85
7-167	<code>typeinfo</code> for <code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;</code> .....	87
7-168	<code>libstdcxx</code> - Class <code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;</code> Function Interfaces .....	87
7-169	<code>typeinfo</code> for <code>ios_base</code> .....	88
7-170	Primary vtable for <code>basic_ios&lt;char, char_traits&lt;char&gt; &gt;</code> .....	88
7-171	Primary vtable for <code>ios_base::failure</code> .....	88
7-172	<code>typeinfo</code> for <code>ios_base::failure</code> .....	89
7-173	Primary vtable for <code>__timepunct&lt;char&gt;</code> .....	89
7-174	<code>typeinfo</code> for <code>__timepunct&lt;char&gt;</code> .....	89
7-175	<code>libstdcxx</code> - Class <code>__timepunct&lt;char&gt;</code> Function Interfaces .....	89
7-176	Primary vtable for <code>__timepunct&lt;wchar_t&gt;</code> .....	90
7-177	<code>typeinfo</code> for <code>__timepunct&lt;wchar_t&gt;</code> .....	90
7-178	<code>libstdcxx</code> - Class <code>__timepunct&lt;wchar_t&gt;</code> Function Interfaces .....	90
7-179	<code>typeinfo</code> for <code>messages_base</code> .....	91
7-180	Primary vtable for <code>messages&lt;char&gt;</code> .....	91
7-181	<code>libstdcxx</code> - Class <code>messages&lt;char&gt;</code> Function Interfaces .....	92
7-182	Primary vtable for <code>messages&lt;wchar_t&gt;</code> .....	92
7-183	<code>libstdcxx</code> - Class <code>messages&lt;wchar_t&gt;</code> Function Interfaces .....	93
7-184	Primary vtable for <code>messages_byname&lt;char&gt;</code> .....	93
7-185	<code>typeinfo</code> for <code>messages_byname&lt;char&gt;</code> .....	93
7-186	<code>libstdcxx</code> - Class <code>messages_byname&lt;char&gt;</code> Function Interfaces .....	94
7-187	Primary vtable for <code>messages_byname&lt;wchar_t&gt;</code> .....	94
7-188	<code>typeinfo</code> for <code>messages_byname&lt;wchar_t&gt;</code> .....	94
7-189	<code>libstdcxx</code> - Class <code>messages_byname&lt;wchar_t&gt;</code> Function Interfaces .....	95
7-190	Primary vtable for <code>numpunct&lt;char&gt;</code> .....	95
7-191	<code>typeinfo</code> for <code>numpunct&lt;char&gt;</code> .....	95
7-192	<code>libstdcxx</code> - Class <code>numpunct&lt;char&gt;</code> Function Interfaces .....	96
7-193	Primary vtable for <code>numpunct&lt;wchar_t&gt;</code> .....	96
7-194	<code>typeinfo</code> for <code>numpunct&lt;wchar_t&gt;</code> .....	97
7-195	<code>libstdcxx</code> - Class <code>numpunct&lt;wchar_t&gt;</code> Function Interfaces .....	97
7-196	Primary vtable for <code>numpunct_byname&lt;char&gt;</code> .....	97
7-197	<code>typeinfo</code> for <code>numpunct_byname&lt;char&gt;</code> .....	98
7-198	<code>libstdcxx</code> - Class <code>numpunct_byname&lt;char&gt;</code> Function Interfaces .....	98
7-199	Primary vtable for <code>numpunct_byname&lt;wchar_t&gt;</code> .....	98
7-200	<code>typeinfo</code> for <code>numpunct_byname&lt;wchar_t&gt;</code> .....	99
7-201	<code>libstdcxx</code> - Class <code>numpunct_byname&lt;wchar_t&gt;</code> Function Interfaces .....	99
7-202	Primary vtable for <code>__codecvt_abstract_base&lt;wchar_t, char, __mbstate_t&gt;</code> .....	99
7-203	<code>typeinfo</code> for <code>codecvt_base</code> .....	100
7-204	Primary vtable for <code>codecvt&lt;char, char, __mbstate_t&gt;</code> .....	100
7-205	<code>typeinfo</code> for <code>codecvt&lt;char, char, __mbstate_t&gt;</code> .....	101
7-206	Primary vtable for <code>__codecvt_abstract_base&lt;char, char, __mbstate_t&gt;</code> .....	101
7-207	<code>libstdcxx</code> - Class <code>codecvt&lt;char, char, __mbstate_t&gt;</code> Function Interfaces .....	102
7-208	Primary vtable for <code>codecvt&lt;wchar_t, char, __mbstate_t&gt;</code> .....	102
7-209	<code>typeinfo</code> for <code>codecvt&lt;wchar_t, char, __mbstate_t&gt;</code> .....	103
7-210	<code>libstdcxx</code> - Class <code>codecvt&lt;wchar_t, char, __mbstate_t&gt;</code> Function Interfaces .....	103
7-211	Primary vtable for <code>codecvt_byname&lt;char, char, __mbstate_t&gt;</code> .....	104
7-212	<code>typeinfo</code> for <code>codecvt_byname&lt;char, char, __mbstate_t&gt;</code> .....	105
7-213	<code>libstdcxx</code> - Class <code>codecvt_byname&lt;char, char, __mbstate_t&gt;</code> Function Interfaces .....	105
7-214	Primary vtable for <code>codecvt_byname&lt;wchar_t, char, __mbstate_t&gt;</code> .....	105
7-215	<code>typeinfo</code> for <code>codecvt_byname&lt;wchar_t, char, __mbstate_t&gt;</code> .....	106
7-216	Primary vtable for <code>collate_byname&lt;wchar_t&gt;</code> .....	106
7-217	<code>typeinfo</code> for <code>collate_byname&lt;wchar_t&gt;</code> .....	107



7-218 libstdcxx - Class codecvt_byname<wchar_t, char, __mbstate_t> Function Interfaces.....	107
7-219 Primary vtable for collate<char> .....	107
7-220 typeinfo for collate<char> .....	108
7-221 libstdcxx - Class collate<char> Function Interfaces .....	108
7-222 Primary vtable for collate<wchar_t> .....	108
7-223 typeinfo for collate<wchar_t>.....	109
7-224 libstdcxx - Class collate<wchar_t> Function Interfaces.....	109
7-225 Primary vtable for collate_byname<char> .....	109
7-226 typeinfo for collate_byname<char> .....	110
7-227 libstdcxx - Class collate_byname<char> Function Interfaces .....	110
7-228 typeinfo for time_base.....	110
7-229 Primary vtable for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > > .....	111
7-230 typeinfo for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > > .....	112
7-231 libstdcxx - Class time_get_byname<char, istreambuf_iterator<char, char_traits<char> > > Function Interfaces .....	112
7-232 Primary vtable for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > .....	113
7-233 typeinfo for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > .....	114
7-234 libstdcxx - Class time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces.....	114
7-235 Primary vtable for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > > .....	115
7-236 typeinfo for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > > .....	115
7-237 libstdcxx - Class time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > > Function Interfaces .....	116
7-238 Primary vtable for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > .....	116
7-239 typeinfo for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > .....	117
7-240 libstdcxx - Class time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces.....	117
7-241 Primary vtable for time_get<char, istreambuf_iterator<char, char_traits<char> > > .....	117
7-242 libstdcxx - Class time_get<char, istreambuf_iterator<char, char_traits<char> > > Function Interfaces .....	119
7-243 Primary vtable for time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > .....	119
7-244 libstdcxx - Class time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces.....	121
7-245 libstdcxx - Class time_put<char, ostreambuf_iterator<char, char_traits<char> > > Function Interfaces .....	121
7-246 libstdcxx - Class time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces.....	121
7-247 Primary vtable for moneypunct<char, false> .....	122
7-248 libstdcxx - Class moneypunct<char, false> Function Interfaces .....	122
7-249 Primary vtable for moneypunct<char, true> .....	123
7-250 libstdcxx - Class moneypunct<char, true> Function Interfaces .....	124

7-251	Primary vtable for <code>moneypunct&lt;wchar_t, false&gt;</code> .....	124
7-252	<code>libstdcxx</code> - Class <code>moneypunct&lt;wchar_t, false&gt;</code> Function Interfaces.....	125
7-253	Primary vtable for <code>moneypunct&lt;wchar_t, true&gt;</code> .....	125
7-254	<code>libstdcxx</code> - Class <code>moneypunct&lt;wchar_t, true&gt;</code> Function Interfaces.....	126
7-255	Primary vtable for <code>moneypunct_byname&lt;char, false&gt;</code> .....	127
7-256	<code>typeinfo</code> for <code>moneypunct_byname&lt;char, false&gt;</code> .....	127
7-257	<code>libstdcxx</code> - Class <code>moneypunct_byname&lt;char, false&gt;</code> Function Interfaces.....	128
7-258	Primary vtable for <code>moneypunct_byname&lt;char, true&gt;</code> .....	128
7-259	<code>typeinfo</code> for <code>moneypunct_byname&lt;char, true&gt;</code> .....	129
7-260	<code>libstdcxx</code> - Class <code>moneypunct_byname&lt;char, true&gt;</code> Function Interfaces.....	129
7-261	Primary vtable for <code>moneypunct_byname&lt;wchar_t, false&gt;</code> .....	129
7-262	<code>typeinfo</code> for <code>moneypunct_byname&lt;wchar_t, false&gt;</code> .....	130
7-263	<code>libstdcxx</code> - Class <code>moneypunct_byname&lt;wchar_t, false&gt;</code> Function Interfaces.....	130
7-264	Primary vtable for <code>moneypunct_byname&lt;wchar_t, true&gt;</code> .....	130
7-265	<code>typeinfo</code> for <code>moneypunct_byname&lt;wchar_t, true&gt;</code> .....	131
7-266	<code>libstdcxx</code> - Class <code>moneypunct_byname&lt;wchar_t, true&gt;</code> Function Interfaces.....	131
7-267	<code>typeinfo</code> for <code>money_base</code> .....	132
7-268	Primary vtable for <code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;&gt;</code> .....	132
7-269	<code>typeinfo</code> for <code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;&gt;</code> .....	133
7-270	<code>libstdcxx</code> - Class <code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;&gt;</code> Function Interfaces.....	133
7-271	Primary vtable for <code>money_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;&gt;</code> .....	133
7-272	<code>typeinfo</code> for <code>money_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;&gt;</code> .....	134
7-273	<code>libstdcxx</code> - Class <code>money_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;&gt;</code> Function Interfaces.....	135
7-274	Primary vtable for <code>money_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;&gt;</code> .....	135
7-275	<code>typeinfo</code> for <code>money_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;&gt;</code> .....	136
7-276	<code>libstdcxx</code> - Class <code>money_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;&gt;</code> Function Interfaces.....	136
7-277	Primary vtable for <code>money_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;&gt;</code> .....	136
7-278	<code>typeinfo</code> for <code>money_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;&gt;</code> .....	137
7-279	<code>libstdcxx</code> - Class <code>money_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;&gt;</code> Function Interfaces.....	137
7-280	<code>libstdcxx</code> - Class <code>locale</code> Function Interfaces.....	138
7-281	Primary vtable for <code>locale::facet</code> .....	138
7-282	<code>typeinfo</code> for <code>locale::facet</code> .....	138
7-283	.....	139
7-284	<code>libstdcxx</code> - Class <code>num_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;&gt;</code> Function Interfaces.....	139
7-285	<code>libstdcxx</code> - Class <code>num_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;&gt;</code> Function Interfaces.....	139
7-286	<code>libstdcxx</code> - Class <code>num_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;&gt;</code> Function Interfaces.....	140
7-287	<code>libstdcxx</code> - Class <code>num_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;&gt;</code> Function Interfaces.....	140

7-288 libstdcxx - Class <code>gslice</code> Function Interfaces .....	141
7-289 libstdcxx - Class <code>__basic_file&lt;char&gt;</code> Function Interfaces .....	141
7-290 libstdcxx - Class <code>valarray&lt;unsigned int&gt;</code> Function Interfaces .....	142

## Foreword

This is version 3.1 of the Linux Standard Base C++ Specification for IA32. This specification is part of a family of specifications under the general title "Linux Standard Base". Developers of applications or implementations interested in using the LSB trademark should see the Free Standards Group Certification Policy for details.

## Introduction

The LSB defines a binary interface for application programs that are compiled and packaged for LSB-conforming implementations on many different hardware architectures. Since a binary specification shall include information specific to the computer processor architecture for which it is intended, it is not possible for a single document to specify the interface for all possible LSB-conforming implementations. Therefore, the LSB is a family of specifications, rather than a single one.

This document should be used in conjunction with the documents it references. This document enumerates the system components it includes, but descriptions of those components may be included entirely or partly in this document, partly in other documents, or entirely in other reference documents. For example, the section that describes system service routines includes a list of the system routines supported in this interface, formal declarations of the data structures they use that are visible to applications, and a pointer to the underlying referenced specification for information about the syntax and semantics of each call. Only those routines not described in standards referenced by this document, or extensions to those standards, are described in the detail. Information referenced in this way is as much a part of this document as is the information explicitly included here.

The specification carries a version number of either the form  $x.y$  or  $x.y.z$ . This version number carries the following meaning:

- The first number ( $x$ ) is the major version number. All versions with the same major version number should share binary compatibility. Any addition or deletion of a new library results in a new version number. Interfaces marked as `deprecated` may be removed from the specification at a major version change.
- The second number ( $y$ ) is the minor version number. Individual interfaces may be added if all certified implementations already had that (previously undocumented) interface. Interfaces may be marked as `deprecated` at a minor version change. Other minor changes may be permitted at the discretion of the LSB workgroup.
- The third number ( $z$ ), if present, is the editorial level. Only editorial changes should be included in such versions.

Since this specification is a descriptive Application Binary Interface, and not a source level API specification, it is not possible to make a guarantee of 100% backward compatibility between major releases. However, it is the intent that those parts of the binary interface that are visible in the source level API will remain backward compatible from version to version, except where a feature marked as "Deprecated" in one release may be removed from a future release.

Implementors are strongly encouraged to make use of symbol versioning to permit simultaneous support of applications conforming to different releases of this specification.



# I Introductory Elements





# 1 Scope

## 1.1 General

The Linux Standard Base (LSB) defines a system interface for compiled applications and a minimal environment for support of installation scripts. Its purpose is to enable a uniform industry standard environment for high-volume applications conforming to the LSB.

These specifications are composed of two basic parts: A common specification ("LSB-generic" or "generic LSB"), ISO/IEC 23360 Part 1, describing those parts of the interface that remain constant across all implementations of the LSB, and an architecture-specific part ("LSB-arch" or "archLSB") describing the parts of the interface that vary by processor architecture. Together, the LSB-generic and the relevant architecture-specific part of ISO/IEC 23360 for a single hardware architecture provide a complete interface specification for compiled application programs on systems that share a common hardware architecture.

ISO/IEC 23360 Part 1, the LSB-generic document, should be used in conjunction with an architecture-specific part. Whenever a section of the LSB-generic specification is supplemented by architecture-specific information, the LSB-generic document includes a reference to the architecture part. Architecture-specific parts of ISO/IEC 23360 may also contain additional information that is not referenced in the LSB-generic document.

The LSB contains both a set of Application Program Interfaces (APIs) and Application Binary Interfaces (ABIs). APIs may appear in the source code of portable applications, while the compiled binary of that application may use the larger set of ABIs. A conforming implementation provides all of the ABIs listed here. The compilation system may replace (e.g. by macro definition) certain APIs with calls to one or more of the underlying binary interfaces, and may insert calls to binary interfaces as needed.

The LSB is primarily a binary interface definition. Not all of the source level APIs available to applications may be contained in this specification.

## 1.2 Module Specific Scope

This is the C++ module of the Linux Standards Base (LSB). This module supplements the core interfaces by providing system interfaces, libraries, and a runtime environment for applications built using the C++ programming language. These interfaces provide low-level support for the core constructs of the language, and implement the standard base C++ libraries.

Interfaces described in this module are presented in terms of C++; the binary interfaces will use encoded or mangled versions of the names.

## 2 Normative References

The specifications listed below are referenced in whole or in part by this module of the Linux Standard Base. In this specification, where only a particular section of one of these references is identified, then the normative reference is to that section alone, and the rest of the referenced document is informative.

**Table 2-1 Normative References**

Name	Title	URL
ISO/IEC 23360 Part 1	ISO/IEC 23360:2005 Linux Standard Base - Part 1 Generic Specification	<a href="http://www.linuxbase.org/spec/">http://www.linuxbase.org/spec/</a>
ISO C (1999)	ISO/IEC 9899: 1999, Programming Languages --C	
ISO POSIX (2003)	ISO/IEC 9945-1:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions  ISO/IEC 9945-2:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces  ISO/IEC 9945-3:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 3: Shell and Utilities  ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale  Including Technical Cor. 1: 2004	<a href="http://www.unix.org/version3/">http://www.unix.org/ version3/</a>
ISO/IEC 14882: 2003 C++ Language	ISO/IEC 14882: 2003 Programming languages --C++	
Itanium™ C++ ABI	Itanium™ C++ ABI (Revision 1.83)	<a href="http://refspecs.freestdards.org/cxxabi-1.83.html">http://refspecs.freestdards.org/cxxabi-1.83.html</a>

## 3 Requirements

### 3.1 Relevant Libraries

The libraries listed in Table 3-1 shall be available on a Linux Standard Base system, with the specified runtime names.

**Table 3-1 Standard Library Names**

Library	Runtime Name
libstdcxx	libstdc++.so.6

These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

### 3.2 LSB Implementation Conformance

An implementation shall satisfy the following requirements:

- The implementation shall implement fully the architecture described in the hardware manual for the target processor architecture.
- The implementation shall be capable of executing compiled applications having the format and using the system interfaces described in this document.
- The implementation shall provide libraries containing the interfaces specified by this document, and shall provide a dynamic linking mechanism that allows these interfaces to be attached to applications at runtime. All the interfaces shall behave as specified in this document.
- The map of virtual memory provided by the implementation shall conform to the requirements of this document.
- The implementation's low-level behavior with respect to function call linkage, system traps, signals, and other such activities shall conform to the formats described in this document.
- The implementation shall provide all of the mandatory interfaces in their entirety.
- The implementation may provide one or more of the optional interfaces. Each optional interface that is provided shall be provided in its entirety. The product documentation shall state which optional interfaces are provided.
- The implementation shall provide all files and utilities specified as part of this document in the format defined here and in other referenced documents. All commands and utilities shall behave as required by this document. The implementation shall also provide all mandatory components of an application's runtime environment that are included or referenced in this document.
- The implementation, when provided with standard data formats and values at a named interface, shall provide the behavior defined for those values and data formats at that interface. However, a conforming implementation may consist of components which are separately packaged and/or sold. For example, a vendor of a conforming implementation might sell the hardware, operating system, and windowing system as separately packaged items.

- The implementation may provide additional interfaces with different names. It may also provide additional behavior corresponding to data values outside the standard ranges, for standard named interfaces.

### 3.3 LSB Application Conformance

An application shall satisfy the following requirements:

- Its executable files are either shell scripts or object files in the format defined for the Object File Format system interface.
- Its object files participate in dynamic linking as defined in the Program Loading and Linking System interface.
- It employs only the instructions, traps, and other low-level facilities defined in the Low-Level System interface as being for use by applications.
- If it requires any optional interface defined in this document in order to be installed or to execute successfully, the requirement for that optional interface is stated in the application's documentation.
- It does not use any interface or data format that is not required to be provided by a conforming implementation, unless:
  - If such an interface or data format is supplied by another application through direct invocation of that application during execution, that application is in turn an LSB conforming application.
  - The use of that interface or data format, as well as its source, is identified in the documentation of the application.
- It shall not use any values for a named interface that are reserved for vendor extensions.

A strictly conforming application does not require or use any interface, facility, or implementation-defined extension that is not defined in this document in order to be installed or to execute successfully.

## 4 Definitions

For the purposes of this document, the following definitions, as specified in the *ISO/IEC Directives, Part 2, 2001, 4th Edition*, apply:

can

be able to; there is a possibility of; it is possible to

cannot

be unable to; there is no possibility of; it is not possible to

may

is permitted; is allowed; is permissible

need not

it is not required that; no...is required

shall

is to; is required to; it is required that; has to; only...is permitted; it is necessary

shall not

is not allowed [permitted] [acceptable] [permissible]; is required to be not; is required that...be not; is not to be

should

it is recommended that; ought to

should not

it is not recommended that; ought not to

## 5 Terminology

For the purposes of this document, the following terms apply:

### archLSB

The architectural part of the LSB Specification which describes the specific parts of the interface that are platform specific. The archLSB is complementary to the gLSB.

### Binary Standard

The total set of interfaces that are available to be used in the compiled binary code of a conforming application.

### gLSB

The common part of the LSB Specification that describes those parts of the interface that remain constant across all hardware implementations of the LSB.

### implementation-defined

Describes a value or behavior that is not defined by this document but is selected by an implementor. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations. The implementor shall document such a value or behavior so that it can be used correctly by an application.

### Shell Script

A file that is read by an interpreter (e.g., awk). The first line of the shell script includes a reference to its interpreter binary.

### Source Standard

The set of interfaces that are available to be used in the source code of a conforming application.

### undefined

Describes the nature of a value or behavior not defined by this document which results from use of an invalid program construct or invalid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

### unspecified

Describes the nature of a value or behavior not specified by this document which results from use of a valid program construct or valid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

Other terms and definitions used in this document shall have the same meaning as defined in Chapter 3 of the Base Definitions volume of ISO POSIX (2003).

## 6 Documentation Conventions

Throughout this document, the following typographic conventions are used:

`function()`

the name of a function

**command**

the name of a command or utility

CONSTANT

a constant value

*parameter*

a parameter

variable

a variable

Throughout this specification, several tables of interfaces are presented. Each entry in these tables has the following format:

name

the name of the interface

(symver)

An optional symbol version identifier, if required.

[*refno*]

A reference number indexing the table of referenced specifications that follows this table.

For example,

<code>forkpty(GLIBC_2.0) [SUSv3]</code>
---

refers to the interface named `forkpty()` with symbol version `GLIBC_2.0` that is defined in the `SUSv3` reference.

**Note:** Symbol versions are defined in the architecture specific parts of ISO/IEC 23360 only.



## II Base Libraries

## 7 Libraries

An LSB-conforming implementation shall support some base libraries which provide interfaces for accessing the operating system, processor and other hardware in the system.

Interfaces that are unique to the IA32 platform are defined here. This section should be used in conjunction with the corresponding section in the Linux Standard Base Specification.

### 7.1 Interfaces for libstdc++

Table 7-1 defines the library name and shared object name for the libstdc++ library

**Table 7-1 libstdc++ Definition**

Library:	libstdc++
SONAME:	libstdc++.so.6

The behavior of the interfaces in this library is specified by the following specifications:

[CXXABI] Itanium™ C++ ABI

[ISOCXX] ISO/IEC 14882: 2003 C++ Language

[LSB] ISO/IEC 23360 Part 1

#### 7.1.1 C++ Runtime Support

##### 7.1.1.1 Interfaces for C++ Runtime Support

An LSB conforming implementation shall provide the architecture specific methods for C++ Runtime Support specified in Table 7-2, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-2 libstdc++ - C++ Runtime Support Function Interfaces**

<code>operator new[](unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>operator new[](unsigned int, nothrow_t const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
<code>operator new(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>operator new(unsigned int, nothrow_t const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>

#### 7.1.2 C++ type descriptors for built-in types

##### 7.1.2.1 Interfaces for C++ type descriptors for built-in types

No external methods are defined for libstdc++ - C++ type descriptors for built-in types in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

#### 7.1.3 C++ `_Rb_tree`

##### 7.1.3.1 Interfaces for C++ `_Rb_tree`

No external methods are defined for libstdc++ - C++ `_Rb_tree` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

## 7.1.4 Class `type_info`

### 7.1.4.1 Class data for `type_info`

The virtual table for the `std::type_info` class is described by Table 7-3

**Table 7-3 Primary vtable for `type_info`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>type_info</code>
<code>vfunc[0]:</code>	<code>type_info::~~type_info()</code>
<code>vfunc[1]:</code>	<code>type_info::~~type_info()</code>
<code>vfunc[2]:</code>	<code>type_info::__is_pointer_p() const</code>
<code>vfunc[3]:</code>	<code>type_info::__is_function_p() const</code>
<code>vfunc[4]:</code>	<code>type_info::__do_catch(type_info const*, void**, unsigned int) const</code>
<code>vfunc[5]:</code>	<code>type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const</code>

The Run Time Type Information for the `std::type_info` class is described by Table 7-4

**Table 7-4 typeinfo for `type_info`**

Base Vtable	vtable for <code>__cxxabiv1::__class_type_info</code>
Name	typeinfo name for <code>type_info</code>

### 7.1.4.2 Interfaces for Class `type_info`

No external methods are defined for `libstdc++` - Class `std::type_info` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

## 7.1.5 Class `__cxxabiv1::__enum_type_info`

### 7.1.5.1 Class data for `__cxxabiv1::__enum_type_info`

The virtual table for the `__cxxabiv1::__enum_type_info` class is described by Table 7-5

**Table 7-5 Primary vtable for `__cxxabiv1::__enum_type_info`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>__cxxabiv1::__enum_type_info</code>
<code>vfunc[0]:</code>	<code>__cxxabiv1::__enum_type_info::~~enum_type_info()</code>
<code>vfunc[1]:</code>	<code>__cxxabiv1::__enum_type_info::~~e</code>

	num_type_info()
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::_class_type_info const*, void**) const

The Run Time Type Information for the `__cxxabiv1::_enum_type_info` class is described by Table 7-6

**Table 7-6 typeid for `__cxxabiv1::_enum_type_info`**

Base Vtable	vtable for <code>__cxxabiv1::_si_class_type_info</code>
Name	typeid name for <code>__cxxabiv1::_enum_type_info</code>

### 7.1.5.2 Interfaces for Class `__cxxabiv1::_enum_type_info`

No external methods are defined for `libstdcxx` - Class `__cxxabiv1::_enum_type_info` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.6 Class `__cxxabiv1::_array_type_info`

#### 7.1.6.1 Class data for `__cxxabiv1::_array_type_info`

The virtual table for the `__cxxabiv1::_array_type_info` class is described by Table 7-7

**Table 7-7 Primary vtable for `__cxxabiv1::_array_type_info`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>__cxxabiv1::_array_type_info</code>
vfunc[0]:	<code>__cxxabiv1::_array_type_info::~~array_type_info()</code>
vfunc[1]:	<code>__cxxabiv1::_array_type_info::~~array_type_info()</code>
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::_class_type_info const*, void**) const

The Run Time Type Information for the `__cxxabiv1::__array_type_info` class is described by Table 7-8

**Table 7-8 typeid for `__cxxabiv1::__array_type_info`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>__cxxabiv1::__array_type_info</code>

### 7.1.6.2 Interfaces for Class `__cxxabiv1::__array_type_info`

No external methods are defined for `libstdcxx` - Class `__cxxabiv1::__array_type_info` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.7 Class `__cxxabiv1::__class_type_info`

#### 7.1.7.1 Class data for `__cxxabiv1::__class_type_info`

The virtual table for the `__cxxabiv1::__class_type_info` class is described by Table 7-9

**Table 7-9 Primary vtable for `__cxxabiv1::__class_type_info`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>__cxxabiv1::__class_type_info</code>
vfunc[0]:	<code>__cxxabiv1::__class_type_info::~__class_type_info()</code>
vfunc[1]:	<code>__cxxabiv1::__class_type_info::~__class_type_info()</code>
vfunc[2]:	<code>type_info::__is_pointer_p() const</code>
vfunc[3]:	<code>type_info::__is_function_p() const</code>
vfunc[4]:	<code>__cxxabiv1::__class_type_info::__do_catch(type_info const*, void**, unsigned int) const</code>
vfunc[5]:	<code>__cxxabiv1::__class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const</code>
vfunc[6]:	<code>__cxxabiv1::__class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__upcast_result&amp;) const</code>
vfunc[7]:	<code>__cxxabiv1::__class_type_info::__do_dynccast(int, __cxxabiv1::__class_type_info::__sub_kind, __cxxabiv1::__class_type_info</code>

	const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dyn cast_result&) const
vfunc[8]:	__cxxabiv1::__class_type_info::__do_ find_public_src(int, void const*, __cxxabiv1::__class_type_info const*, void const*) const

The Run Time Type Information for the \_\_cxxabiv1::\_\_class\_type\_info class is described by Table 7-10

**Table 7-10 typeinfo for \_\_cxxabiv1::\_\_class\_type\_info**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__class_type_info

### 7.1.7.2 Interfaces for Class \_\_cxxabiv1::\_\_class\_type\_info

An LSB conforming implementation shall provide the architecture specific methods for Class \_\_cxxabiv1::\_\_class\_type\_info specified in Table 7-11, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-11 libstdc++ - Class \_\_cxxabiv1::\_\_class\_type\_info Function Interfaces**

__cxxabiv1::__class_type_info::__do_dyncast(int, __cxxabiv1::__class_type_info::__sub_kind, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dyncast_result&) const(CXXABI_1.3) [CXXABI]
__cxxabiv1::__class_type_info::__do_find_public_src(int, void const*, __cxxabiv1::__class_type_info const*, void const*) const(CXXABI_1.3) [CXXABI]

### 7.1.8 Class \_\_cxxabiv1::\_\_pbase\_type\_info

#### 7.1.8.1 Class data for \_\_cxxabiv1::\_\_pbase\_type\_info

The virtual table for the \_\_cxxabiv1::\_\_pbase\_type\_info class is described by Table 7-12

**Table 7-12 Primary vtable for \_\_cxxabiv1::\_\_pbase\_type\_info**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__pbase_type_info
vfunc[0]:	__cxxabiv1::__pbase_type_info::~~p base_type_info()

vfunc[1]:	<code>__cxxabiv1::__pbase_type_info::~~__pbase_type_info()</code>
vfunc[2]:	<code>type_info::__is_pointer_p() const</code>
vfunc[3]:	<code>type_info::__is_function_p() const</code>
vfunc[4]:	<code>__cxxabiv1::__pbase_type_info::__do_catch(type_info const*, void**, unsigned int) const</code>
vfunc[5]:	<code>type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const</code>
vfunc[6]:	<code>__cxxabiv1::__pbase_type_info::__pointer_catch(__cxxabiv1::__pbase_type_info const*, void**, unsigned int) const</code>

The Run Time Type Information for the `__cxxabiv1::__pbase_type_info` class is described by Table 7-13

**Table 7-13 typeid for `__cxxabiv1::__pbase_type_info`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>__cxxabiv1::__pbase_type_info</code>

### 7.1.8.2 Interfaces for Class `__cxxabiv1::__pbase_type_info`

No external methods are defined for `libstdcxx` - Class `__cxxabiv1::__pbase_type_info` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.9 Class `__cxxabiv1::__pointer_type_info`

#### 7.1.9.1 Class data for `__cxxabiv1::__pointer_type_info`

The virtual table for the `__cxxabiv1::__pointer_type_info` class is described by Table 7-14

**Table 7-14 Primary vtable for `__cxxabiv1::__pointer_type_info`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>__cxxabiv1::__pointer_type_info</code>
vfunc[0]:	<code>__cxxabiv1::__pointer_type_info::~~__pointer_type_info()</code>
vfunc[1]:	<code>__cxxabiv1::__pointer_type_info::~~__pointer_type_info()</code>
vfunc[2]:	<code>__cxxabiv1::__pointer_type_info::__is_pointer_p() const</code>

vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	__cxxabiv1::__pbase_type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const
vfunc[6]:	__cxxabiv1::__pointer_type_info::__pointer_catch(__cxxabiv1::__pbase_type_info const*, void**, unsigned int) const

The Run Time Type Information for the `__cxxabiv1::__pointer_type_info` class is described by Table 7-15

**Table 7-15 typeinfo for `__cxxabiv1::__pointer_type_info`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>__cxxabiv1::__pointer_type_info</code>

### 7.1.9.2 Interfaces for Class `__cxxabiv1::__pointer_type_info`

No external methods are defined for `libstdcxx` - Class `__cxxabiv1::__pointer_type_info` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.10 Class `__cxxabiv1::__function_type_info`

#### 7.1.10.1 Class data for `__cxxabiv1::__function_type_info`

The virtual table for the `__cxxabiv1::__function_type_info` class is described by Table 7-16

**Table 7-16 Primary vtable for `__cxxabiv1::__function_type_info`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>__cxxabiv1::__function_type_info</code>
vfunc[0]:	<code>__cxxabiv1::__function_type_info::~~_function_type_info()</code>
vfunc[1]:	<code>__cxxabiv1::__function_type_info::~~_function_type_info()</code>
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	<code>__cxxabiv1::__function_type_info::__is_function_p()</code> const
vfunc[4]:	type_info::__do_catch(type_info const*, void**, unsigned int) const



vfunc[5]:	type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const
-----------	--

The Run Time Type Information for the `__cxxabiv1::__function_type_info` class is described by Table 7-17

**Table 7-17 typeinfo for `__cxxabiv1::__function_type_info`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>__cxxabiv1::__function_type_info</code>

### 7.1.10.2 Interfaces for Class `__cxxabiv1::__function_type_info`

No external methods are defined for `libstdcxx` - Class `__cxxabiv1::__function_type_info` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.11 Class `__cxxabiv1::__si_class_type_info`

#### 7.1.11.1 Class data for `__cxxabiv1::__si_class_type_info`

The virtual table for the `__cxxabiv1::__si_class_type_info` class is described by Table 7-18

**Table 7-18 Primary vtable for `__cxxabiv1::__si_class_type_info`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>__cxxabiv1::__si_class_type_info</code>
vfunc[0]:	<code>__cxxabiv1::__si_class_type_info::~~__si_class_type_info()</code>
vfunc[1]:	<code>__cxxabiv1::__si_class_type_info::~~__si_class_type_info()</code>
vfunc[2]:	<code>type_info::__is_pointer_p() const</code>
vfunc[3]:	<code>type_info::__is_function_p() const</code>
vfunc[4]:	<code>__cxxabiv1::__class_type_info::__do_catch(type_info const*, void**, unsigned int) const</code>
vfunc[5]:	<code>__cxxabiv1::__class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const</code>
vfunc[6]:	<code>__cxxabiv1::__si_class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__upcast_result&amp;) const</code>

vfunc[7]:	<code>__cxxabiv1::__si_class_type_info::__do_dynccast(int, __cxxabiv1::__class_type_info::__sub_kind, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dynccast_result&amp;) const</code>
vfunc[8]:	<code>__cxxabiv1::__si_class_type_info::__do_find_public_src(int, void const*, __cxxabiv1::__class_type_info const*, void const*) const</code>

The Run Time Type Information for the `__cxxabiv1::__si_class_type_info` class is described by Table 7-19

**Table 7-19 typeinfo for `__cxxabiv1::__si_class_type_info`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>__cxxabiv1::__si_class_type_info</code>

### 7.1.11.2 Interfaces for Class `__cxxabiv1::__si_class_type_info`

An LSB conforming implementation shall provide the architecture specific methods for Class `__cxxabiv1::__si_class_type_info` specified in Table 7-20, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-20 libstdc++ - Class `__cxxabiv1::__si_class_type_info` Function Interfaces**

<code>__cxxabiv1::__si_class_type_info::__do_dynccast(int, __cxxabiv1::__class_type_info::__sub_kind, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dynccast_result&amp;) const(CXXABI_1.3)</code> [CXXABI]
<code>__cxxabiv1::__si_class_type_info::__do_find_public_src(int, void const*, __cxxabiv1::__class_type_info const*, void const*) const(CXXABI_1.3)</code> [CXXABI]

### 7.1.12 Class `__cxxabiv1::__vmi_class_type_info`

#### 7.1.12.1 Class data for `__cxxabiv1::__vmi_class_type_info`

The virtual table for the `__cxxabiv1::__vmi_class_type_info` class is described by Table 7-21

**Table 7-21 Primary vtable for `__cxxabiv1::__vmi_class_type_info`**

Base Offset	0
Virtual Base Offset	0

RTTI	typeinfo for __cxxabiv1::__vmi_class_type_info
vfunc[0]:	__cxxabiv1::__vmi_class_type_info::~~ __vmi_class_type_info()
vfunc[1]:	__cxxabiv1::__vmi_class_type_info::~~ __vmi_class_type_info()
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	__cxxabiv1::__class_type_info::__do_ catch(type_info const*, void**, unsigned int) const
vfunc[5]:	__cxxabiv1::__class_type_info::__do_ upcast(__cxxabiv1::__class_type_info const*, void**) const
vfunc[6]:	__cxxabiv1::__vmi_class_type_info::__ do_upcast(__cxxabiv1::__class_type _info const*, void const*, __cxxabiv1::__class_type_info::__upc ast_result&) const
vfunc[7]:	__cxxabiv1::__vmi_class_type_info::__ do_dyncast(int, __cxxabiv1::__class_type_info::__sub _kind, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dyn cast_result&) const
vfunc[8]:	__cxxabiv1::__vmi_class_type_info::__ do_find_public_src(int, void const*, __cxxabiv1::__class_type_info const*, void const*) const

The Run Time Type Information for the \_\_cxxabiv1::\_\_vmi\_class\_type\_info class is described by Table 7-22

**Table 7-22 typeinfo for \_\_cxxabiv1::\_\_vmi\_class\_type\_info**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__vmi_class_type_info

### 7.1.12.2 Interfaces for Class \_\_cxxabiv1::\_\_vmi\_class\_type\_info

An LSB conforming implementation shall provide the architecture specific methods for Class \_\_cxxabiv1::\_\_vmi\_class\_type\_info specified in Table 7-23, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-23 libstdcxx - Class `__cxxabiv1::__vmi_class_type_info` Function Interfaces**

<code>__cxxabiv1::__vmi_class_type_info::__do_dyncast(int, __cxxabiv1::__class_type_info::__sub_kind, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dyncast_result&amp;) const(CXXABI_1.3)</code> [CXXABI]
<code>__cxxabiv1::__vmi_class_type_info::__do_find_public_src(int, void const*, __cxxabiv1::__class_type_info const*, void const*) const(CXXABI_1.3)</code> [CXXABI]

**7.1.13 Class `__cxxabiv1::__fundamental_type_info`****7.1.13.1 Class data for `__cxxabiv1::__fundamental_type_info`**

The virtual table for the `__cxxabiv1::__fundamental_type_info` class is described by Table 7-24

**Table 7-24 Primary vtable for `__cxxabiv1::__fundamental_type_info`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>__cxxabiv1::__fundamental_type_info</code>
<code>vfunc[0]:</code>	<code>__cxxabiv1::__fundamental_type_info::~~__fundamental_type_info()</code>
<code>vfunc[1]:</code>	<code>__cxxabiv1::__fundamental_type_info::~~__fundamental_type_info()</code>
<code>vfunc[2]:</code>	<code>type_info::__is_pointer_p() const</code>
<code>vfunc[3]:</code>	<code>type_info::__is_function_p() const</code>
<code>vfunc[4]:</code>	<code>type_info::__do_catch(type_info const*, void**, unsigned int) const</code>
<code>vfunc[5]:</code>	<code>type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const</code>

The Run Time Type Information for the `__cxxabiv1::__fundamental_type_info` class is described by Table 7-25

**Table 7-25 typeinfo for `__cxxabiv1::__fundamental_type_info`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>__cxxabiv1::__fundamental_type_info</code>

### 7.1.13.2 Interfaces for Class `__cxxabiv1::__fundamental_type_info`

No external methods are defined for `libstdc++` - Class `__cxxabiv1::__fundamental_type_info` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.14 Class `__cxxabiv1::__pointer_to_member_type_info`

#### 7.1.14.1 Class data for `__cxxabiv1::__pointer_to_member_type_info`

The virtual table for the `__cxxabiv1::__pointer_to_member_type_info` class is described by Table 7-26

**Table 7-26 Primary vtable for `__cxxabiv1::__pointer_to_member_type_info`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>__cxxabiv1::__pointer_to_member_type_info</code>
<code>vfunc[0]:</code>	<code>__cxxabiv1::__pointer_to_member_type_info::~~__pointer_to_member_type_info()</code>
<code>vfunc[1]:</code>	<code>__cxxabiv1::__pointer_to_member_type_info::~~__pointer_to_member_type_info()</code>
<code>vfunc[2]:</code>	<code>type_info::__is_pointer_p() const</code>
<code>vfunc[3]:</code>	<code>type_info::__is_function_p() const</code>
<code>vfunc[4]:</code>	<code>__cxxabiv1::__pbase_type_info::__do_catch(type_info const*, void**, unsigned int) const</code>
<code>vfunc[5]:</code>	<code>type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const</code>
<code>vfunc[6]:</code>	<code>__cxxabiv1::__pointer_to_member_type_info::__pointer_catch(__cxxabiv1::__pbase_type_info const*, void**, unsigned int) const</code>

The Run Time Type Information for the `__cxxabiv1::__pointer_to_member_type_info` class is described by Table 7-27

**Table 7-27 typeinfo for `__cxxabiv1::__pointer_to_member_type_info`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>__cxxabiv1::__pointer_to_member_type_info</code>

**7.1.14.2 Interfaces for Class****\_\_cxxabiv1::\_\_pointer\_to\_member\_type\_info**

No external methods are defined for libstdcxx - Class `__cxxabiv1::__pointer_to_member_type_info` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.15 Class `__gnu_cxx::stdio_filebuf<char, char_traits<char>>`****7.1.15.1 Class data for `__gnu_cxx::stdio_filebuf<char, char_traits<char>>`**

The virtual table for the `__gnu_cxx::stdio_filebuf<char, std::char_traits<char>>` class is described by Table 7-28

**Table 7-28 Primary vtable for `__gnu_cxx::stdio_sync_filebuf<char, char_traits<char>>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>__gnu_cxx::stdio_sync_filebuf&lt;char, char_traits&lt;char&gt;&gt;</code>
vfunc[0]:	Unspecified
vfunc[1]:	Unspecified
vfunc[2]:	<code>basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;::imbue(locale const&amp;)</code>
vfunc[3]:	<code>basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;::setbuf(char*, int)</code>
vfunc[4]:	Unspecified
vfunc[5]:	Unspecified
vfunc[6]:	Unspecified
vfunc[7]:	<code>basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;::showmanyc()</code>
vfunc[8]:	Unspecified
vfunc[9]:	Unspecified
vfunc[10]:	Unspecified
vfunc[11]:	Unspecified
vfunc[12]:	Unspecified
vfunc[13]:	Unspecified

### 7.1.15.2 Interfaces for Class `__gnu_cxx::stdio_filebuf<char, char_traits<char>>`

No external methods are defined for `libstdcxx` - Class `__gnu_cxx::stdio_filebuf<char, std::char_traits<char>>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.16 Class `__gnu_cxx::stdio_filebuf<wchar_t, char_traits<wchar_t>>`

#### 7.1.16.1 Class data for `__gnu_cxx::stdio_filebuf<wchar_t, char_traits<wchar_t>>`

The virtual table for the `__gnu_cxx::stdio_filebuf<wchar_t, std::char_traits<wchar_t>>` class is described by Table 7-29

Table 7-29 Primary vtable for `__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t>>`

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>__gnu_cxx::stdio_sync_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;</code>
vfunc[0]:	Unspecified
vfunc[1]:	Unspecified
vfunc[2]:	<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::imbue(locale const&amp;)</code>
vfunc[3]:	<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::setbuf(wchar_t*, int)</code>
vfunc[4]:	Unspecified
vfunc[5]:	Unspecified
vfunc[6]:	Unspecified
vfunc[7]:	<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::showmanyc()</code>
vfunc[8]:	Unspecified
vfunc[9]:	Unspecified
vfunc[10]:	Unspecified
vfunc[11]:	Unspecified
vfunc[12]:	Unspecified
vfunc[13]:	Unspecified

### 7.1.16.2 Interfaces for Class `__gnu_cxx::stdio_filebuf<wchar_t, char_traits<wchar_t>>`

No external methods are defined for `libstdcxx` - Class `__gnu_cxx::stdio_filebuf<wchar_t, std::char_traits<wchar_t>>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.17 Class `__gnu_cxx::__pool_alloc_base`

#### 7.1.17.1 Interfaces for Class `__gnu_cxx::__pool_alloc_base`

An LSB conforming implementation shall provide the architecture specific methods for Class `__gnu_cxx::__pool_alloc_base` specified in Table 7-30, with the full mandatory functionality as described in the referenced underlying specification.

Table 7-30 `libstdcxx` - Class `__gnu_cxx::__pool_alloc_base` Function Interfaces

<code>__gnu_cxx::__pool_alloc_base::M_get_free_list(unsigned int)(GLIBCXX_3.4.2) [LSB]</code>
<code>__gnu_cxx::__pool_alloc_base::M_refill(unsigned int)(GLIBCXX_3.4.2) [LSB]</code>

### 7.1.18 Class `__gnu_cxx::stdio_sync_filebuf<char, char_traits<char>>`

#### 7.1.18.1 Interfaces for Class `__gnu_cxx::stdio_sync_filebuf<char, char_traits<char>>`

No external methods are defined for `libstdcxx` - Class `__gnu_cxx::stdio_sync_filebuf<char, std::char_traits<char>>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.19 Class `__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t>>`

#### 7.1.19.1 Interfaces for Class `__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t>>`

No external methods are defined for `libstdcxx` - Class `__gnu_cxx::stdio_sync_filebuf<wchar_t, std::char_traits<wchar_t>>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.20 Class exception

#### 7.1.20.1 Class data for exception

The virtual table for the `std::exception` class is described by Table 7-31

Table 7-31 Primary vtable for exception

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for exception
<code>vfunc[0]:</code>	<code>exception::~~exception()</code>
<code>vfunc[1]:</code>	<code>exception::~~exception()</code>



vfunc[2]:	exception::what() const
-----------	-------------------------

The Run Time Type Information for the `std::exception` class is described by Table 7-32

**Table 7-32 typeinfo for exception**

Base Vtable	vtable for <code>__cxxabiv1::__class_type_info</code>
Name	typeinfo name for exception

### 7.1.20.2 Interfaces for Class `exception`

No external methods are defined for `libstdc++` - Class `std::exception` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.21 Class `bad_typeid`

#### 7.1.21.1 Class data for `bad_typeid`

The virtual table for the `std::bad_typeid` class is described by Table 7-33

**Table 7-33 Primary vtable for `bad_typeid`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>bad_typeid</code>
vfunc[0]:	<code>bad_typeid::~~bad_typeid()</code>
vfunc[1]:	<code>bad_typeid::~~bad_typeid()</code>
vfunc[2]:	<code>exception::what() const</code>

The Run Time Type Information for the `std::bad_typeid` class is described by Table 7-34

**Table 7-34 typeinfo for `bad_typeid`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>bad_typeid</code>

#### 7.1.21.2 Interfaces for Class `bad_typeid`

No external methods are defined for `libstdc++` - Class `std::bad_typeid` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.22 Class `logic_error`

#### 7.1.22.1 Class data for `logic_error`

The virtual table for the `std::logic_error` class is described by Table 7-35

**Table 7-35 Primary vtable for logic\_error**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for logic_error
vfunc[0]:	logic_error::~~logic_error()
vfunc[1]:	logic_error::~~logic_error()
vfunc[2]:	logic_error::what() const

The Run Time Type Information for the std::logic\_error class is described by Table 7-36

**Table 7-36 typeid for logic\_error**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for logic_error

### 7.1.22.2 Interfaces for Class logic\_error

No external methods are defined for libstdc++ - Class std::logic\_error in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.23 Class range\_error

#### 7.1.23.1 Class data for range\_error

The virtual table for the std::range\_error class is described by Table 7-37

**Table 7-37 Primary vtable for range\_error**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for range_error
vfunc[0]:	range_error::~~range_error()
vfunc[1]:	range_error::~~range_error()
vfunc[2]:	runtime_error::what() const

The Run Time Type Information for the std::range\_error class is described by Table 7-38

**Table 7-38 typeid for range\_error**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for range_error

### 7.1.23.2 Interfaces for Class `range_error`

No external methods are defined for `libstdc++` - Class `std::range_error` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.24 Class `domain_error`

#### 7.1.24.1 Class data for `domain_error`

The virtual table for the `std::domain_error` class is described by Table 7-39

**Table 7-39 Primary vtable for `domain_error`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeid</code> for <code>domain_error</code>
<code>vfunc[0]:</code>	<code>domain_error::~~domain_error()</code>
<code>vfunc[1]:</code>	<code>domain_error::~~domain_error()</code>
<code>vfunc[2]:</code>	<code>logic_error::what() const</code>

The Run Time Type Information for the `std::domain_error` class is described by Table 7-40

**Table 7-40 `typeid` for `domain_error`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	<code>typeid</code> name for <code>domain_error</code>

#### 7.1.24.2 Interfaces for Class `domain_error`

No external methods are defined for `libstdc++` - Class `std::domain_error` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.25 Class `length_error`

#### 7.1.25.1 Class data for `length_error`

The virtual table for the `std::length_error` class is described by Table 7-41

**Table 7-41 Primary vtable for `length_error`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeid</code> for <code>length_error</code>
<code>vfunc[0]:</code>	<code>length_error::~~length_error()</code>
<code>vfunc[1]:</code>	<code>length_error::~~length_error()</code>
<code>vfunc[2]:</code>	<code>logic_error::what() const</code>

The Run Time Type Information for the `std::length_error` class is described by Table 7-42

**Table 7-42 typeid for `length_error`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>length_error</code>

### 7.1.25.2 Interfaces for Class `length_error`

No external methods are defined for `libstdc++` - Class `std::length_error` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.26 Class `out_of_range`

#### 7.1.26.1 Class data for `out_of_range`

The virtual table for the `std::out_of_range` class is described by Table 7-43

**Table 7-43 Primary vtable for `out_of_range`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>out_of_range</code>
<code>vfunc[0]:</code>	<code>out_of_range::~out_of_range()</code>
<code>vfunc[1]:</code>	<code>out_of_range::~out_of_range()</code>
<code>vfunc[2]:</code>	<code>logic_error::what() const</code>

The Run Time Type Information for the `std::out_of_range` class is described by Table 7-44

**Table 7-44 typeid for `out_of_range`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>out_of_range</code>

#### 7.1.26.2 Interfaces for Class `out_of_range`

No external methods are defined for `libstdc++` - Class `std::out_of_range` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.27 Class `bad_exception`

#### 7.1.27.1 Class data for `bad_exception`

The virtual table for the `std::bad_exception` class is described by Table 7-45

**Table 7-45 Primary vtable for `bad_exception`**

Base Offset	0
-------------	---

Virtual Base Offset	0
RTTI	typeid for bad_exception
vfunc[0]:	bad_exception::~bad_exception()
vfunc[1]:	bad_exception::~bad_exception()
vfunc[2]:	exception::what() const

The Run Time Type Information for the std::bad\_exception class is described by Table 7-46

**Table 7-46 typeid for bad\_exception**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for bad_exception

### 7.1.27.2 Interfaces for Class bad\_exception

No external methods are defined for libstdc++ - Class std::bad\_exception in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.28 Class runtime\_error

#### 7.1.28.1 Class data for runtime\_error

The virtual table for the std::runtime\_error class is described by Table 7-47

**Table 7-47 Primary vtable for runtime\_error**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for runtime_error
vfunc[0]:	runtime_error::~runtime_error()
vfunc[1]:	runtime_error::~runtime_error()
vfunc[2]:	runtime_error::what() const

The Run Time Type Information for the std::runtime\_error class is described by Table 7-48

**Table 7-48 typeid for runtime\_error**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for runtime_error

### 7.1.28.2 Interfaces for Class runtime\_error

No external methods are defined for libstdc++ - Class std::runtime\_error in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

## 7.1.29 Class overflow\_error

### 7.1.29.1 Class data for overflow\_error

The virtual table for the `std::overflow_error` class is described by Table 7-49

**Table 7-49 Primary vtable for overflow\_error**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for overflow_error
vfunc[0]:	overflow_error::~~overflow_error()
vfunc[1]:	overflow_error::~~overflow_error()
vfunc[2]:	runtime_error::what() const

The Run Time Type Information for the `std::overflow_error` class is described by Table 7-50

**Table 7-50 typeinfo for overflow\_error**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for overflow_error

### 7.1.29.2 Interfaces for Class overflow\_error

No external methods are defined for `libstdc++` - Class `std::overflow_error` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

## 7.1.30 Class underflow\_error

### 7.1.30.1 Class data for underflow\_error

The virtual table for the `std::underflow_error` class is described by Table 7-51

**Table 7-51 Primary vtable for underflow\_error**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for underflow_error
vfunc[0]:	underflow_error::~~underflow_error()
vfunc[1]:	underflow_error::~~underflow_error()
vfunc[2]:	runtime_error::what() const

The Run Time Type Information for the `std::underflow_error` class is described by Table 7-52

**Table 7-52 typeinfo for underflow\_error**

Base Vtable	vtable for
-------------	------------

	__cxxabiv1::__si_class_type_info
Name	typeinfo name for underflow_error

### 7.1.30.2 Interfaces for Class underflow\_error

No external methods are defined for libstdcxx - Class std::underflow\_error in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.31 Class invalid\_argument

#### 7.1.31.1 Class data for invalid\_argument

The virtual table for the std::invalid\_argument class is described by Table 7-53

**Table 7-53 Primary vtable for invalid\_argument**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for invalid_argument
vfunc[0]:	invalid_argument::~~invalid_argument()
vfunc[1]:	invalid_argument::~~invalid_argument()
vfunc[2]:	logic_error::what() const

The Run Time Type Information for the std::invalid\_argument class is described by Table 7-54

**Table 7-54 typeinfo for invalid\_argument**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for invalid_argument

### 7.1.31.2 Interfaces for Class invalid\_argument

No external methods are defined for libstdcxx - Class std::invalid\_argument in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.32 Class bad\_cast

#### 7.1.32.1 Class data for bad\_cast

The virtual table for the std::bad\_cast class is described by Table 7-55

**Table 7-55 Primary vtable for bad\_cast**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for bad_cast

vfunc[0]:	bad_cast::~~bad_cast()
vfunc[1]:	bad_cast::~~bad_cast()
vfunc[2]:	exception::what() const

The Run Time Type Information for the std::bad\_cast class is described by Table 7-56

**Table 7-56 typeid for bad\_cast**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for bad_cast

### 7.1.32.2 Interfaces for Class bad\_cast

No external methods are defined for libstdc++ - Class std::bad\_cast in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.33 Class bad\_alloc

#### 7.1.33.1 Class data for bad\_alloc

The virtual table for the std::bad\_alloc class is described by Table 7-57

**Table 7-57 Primary vtable for bad\_alloc**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for bad_alloc
vfunc[0]:	bad_alloc::~~bad_alloc()
vfunc[1]:	bad_alloc::~~bad_alloc()
vfunc[2]:	exception::what() const

The Run Time Type Information for the std::bad\_alloc class is described by Table 7-58

**Table 7-58 typeid for bad\_alloc**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for bad_alloc

#### 7.1.33.2 Interfaces for Class bad\_alloc

No external methods are defined for libstdc++ - Class std::bad\_alloc in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.



**7.1.34 struct `__numeric_limits_base`****7.1.34.1 Interfaces for struct `__numeric_limits_base`**

No external methods are defined for `libstdcxx` - struct `__numeric_limits_base` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.35 struct `numeric_limits<long double>`****7.1.35.1 Interfaces for struct `numeric_limits<long double>`**

No external methods are defined for `libstdcxx` - struct `numeric_limits<long double>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.36 struct `numeric_limits<long long>`****7.1.36.1 Interfaces for struct `numeric_limits<long long>`**

No external methods are defined for `libstdcxx` - struct `numeric_limits<long long>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.37 struct `numeric_limits<unsigned long long>`****7.1.37.1 Interfaces for struct `numeric_limits<unsigned long long>`**

No external methods are defined for `libstdcxx` - struct `numeric_limits<unsigned long long>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.38 struct `numeric_limits<float>`****7.1.38.1 Interfaces for struct `numeric_limits<float>`**

No external methods are defined for `libstdcxx` - struct `numeric_limits<float>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.39 struct `numeric_limits<double>`****7.1.39.1 Interfaces for struct `numeric_limits<double>`**

No external methods are defined for `libstdcxx` - struct `numeric_limits<double>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.40 struct `numeric_limits<short>`****7.1.40.1 Interfaces for struct `numeric_limits<short>`**

No external methods are defined for `libstdcxx` - struct `numeric_limits<short>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### **7.1.41 struct numeric\_limits<unsigned short>**

#### **7.1.41.1 Interfaces for struct numeric\_limits<unsigned short>**

No external methods are defined for libstdcxx - struct numeric\_limits<unsigned short> in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### **7.1.42 struct numeric\_limits<int>**

#### **7.1.42.1 Interfaces for struct numeric\_limits<int>**

No external methods are defined for libstdcxx - struct numeric\_limits<int> in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### **7.1.43 struct numeric\_limits<unsigned int>**

#### **7.1.43.1 Interfaces for struct numeric\_limits<unsigned int>**

No external methods are defined for libstdcxx - struct numeric\_limits<unsigned int> in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### **7.1.44 struct numeric\_limits<long>**

#### **7.1.44.1 Interfaces for struct numeric\_limits<long>**

No external methods are defined for libstdcxx - struct numeric\_limits<long> in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### **7.1.45 struct numeric\_limits<unsigned long>**

#### **7.1.45.1 Interfaces for struct numeric\_limits<unsigned long>**

No external methods are defined for libstdcxx - struct numeric\_limits<unsigned long> in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### **7.1.46 struct numeric\_limits<wchar\_t>**

#### **7.1.46.1 Interfaces for struct numeric\_limits<wchar\_t>**

No external methods are defined for libstdcxx - struct numeric\_limits<wchar\_t> in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### **7.1.47 struct numeric\_limits<unsigned char>**

#### **7.1.47.1 Interfaces for struct numeric\_limits<unsigned char>**

No external methods are defined for libstdcxx - struct numeric\_limits<unsigned char> in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.48 struct numeric\_limits<signed char>****7.1.48.1 Interfaces for struct numeric\_limits<signed char>**

No external methods are defined for libstdcxx - struct numeric\_limits<signed char> in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.49 struct numeric\_limits<char>****7.1.49.1 Interfaces for struct numeric\_limits<char>**

No external methods are defined for libstdcxx - struct numeric\_limits<char> in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.50 struct numeric\_limits<bool>****7.1.50.1 Interfaces for struct numeric\_limits<bool>**

No external methods are defined for libstdcxx - struct numeric\_limits<bool> in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.51 Class ctype\_base****7.1.51.1 Class data for ctype\_base**

The Run Time Type Information for the std::ctype\_base class is described by Table 7-59

**Table 7-59 typeinfo for ctype\_base**

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for ctype_base

**7.1.51.2 Interfaces for Class ctype\_base**

No external methods are defined for libstdcxx - Class std::ctype\_base in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.52 Class \_\_ctype\_abstract\_base<char>****7.1.52.1 Class data for \_\_ctype\_abstract\_base<char>**

The virtual table for the std::\_\_ctype\_abstract\_base<char> class is described by Table 7-60

**Table 7-60 Primary vtable for \_\_ctype\_abstract\_base<char>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __ctype_abstract_base<char>
vfunc[0]:	

vfunc[1]:	
vfunc[2]:	__cxa_pure_virtual
vfunc[3]:	__cxa_pure_virtual
vfunc[4]:	__cxa_pure_virtual
vfunc[5]:	__cxa_pure_virtual
vfunc[6]:	__cxa_pure_virtual
vfunc[7]:	__cxa_pure_virtual
vfunc[8]:	__cxa_pure_virtual
vfunc[9]:	__cxa_pure_virtual
vfunc[10]:	__cxa_pure_virtual
vfunc[11]:	__cxa_pure_virtual
vfunc[12]:	__cxa_pure_virtual
vfunc[13]:	__cxa_pure_virtual

### 7.1.52.2 Interfaces for Class `__ctype_abstract_base<char>`

No external methods are defined for `libstdcxx` - Class `std::__ctype_abstract_base<char>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.53 Class `__ctype_abstract_base<wchar_t>`

#### 7.1.53.1 Class data for `__ctype_abstract_base<wchar_t>`

The virtual table for the `std::__ctype_abstract_base<wchar_t>` class is described by Table 7-61

**Table 7-61 Primary vtable for `__ctype_abstract_base<wchar_t>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>__ctype_abstract_base&lt;wchar_t&gt;</code>
vfunc[0]:	
vfunc[1]:	
vfunc[2]:	__cxa_pure_virtual
vfunc[3]:	__cxa_pure_virtual
vfunc[4]:	__cxa_pure_virtual
vfunc[5]:	__cxa_pure_virtual
vfunc[6]:	__cxa_pure_virtual
vfunc[7]:	__cxa_pure_virtual
vfunc[8]:	__cxa_pure_virtual

vfunc[9]:	__cxa_pure_virtual
vfunc[10]:	__cxa_pure_virtual
vfunc[11]:	__cxa_pure_virtual
vfunc[12]:	__cxa_pure_virtual
vfunc[13]:	__cxa_pure_virtual

### 7.1.53.2 Interfaces for Class `__ctype_abstract_base<wchar_t>`

No external methods are defined for `libstdcxx` - Class `std::__ctype_abstract_base<wchar_t>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.54 Class `ctype<char>`

#### 7.1.54.1 Class data for `ctype<char>`

The virtual table for the `std::ctype<char>` class is described by Table 7-62

Table 7-62 Primary vtable for `ctype<char>`

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>ctype&lt;char&gt;</code>
vfunc[0]:	<code>ctype&lt;char&gt;::~~ctype()</code>
vfunc[1]:	<code>ctype&lt;char&gt;::~~ctype()</code>
vfunc[2]:	<code>ctype&lt;char&gt;::do_toupper(char) const</code>
vfunc[3]:	<code>ctype&lt;char&gt;::do_toupper(char*, char const*) const</code>
vfunc[4]:	<code>ctype&lt;char&gt;::do_tolower(char) const</code>
vfunc[5]:	<code>ctype&lt;char&gt;::do_tolower(char*, char const*) const</code>
vfunc[6]:	<code>ctype&lt;char&gt;::do_widen(char) const</code>
vfunc[7]:	<code>ctype&lt;char&gt;::do_widen(char const*, char const*, char*) const</code>
vfunc[8]:	<code>ctype&lt;char&gt;::do_narrow(char, char) const</code>
vfunc[9]:	<code>ctype&lt;char&gt;::do_narrow(char const*, char const*, char, char*) const</code>

#### 7.1.54.2 Interfaces for Class `ctype<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::ctype<char>` specified in Table 7-63, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-63 libstdcxx - Class `ctype<char>` Function Interfaces**

<code>ctype&lt;char&gt;::ctype(__locale_struct*, unsigned short const*, bool, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>ctype&lt;char&gt;::ctype(unsigned short const*, bool, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>ctype&lt;char&gt;::ctype(__locale_struct*, unsigned short const*, bool, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>ctype&lt;char&gt;::ctype(unsigned short const*, bool, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

**7.1.55 Class `ctype<wchar_t>`****7.1.55.1 Class data for `ctype<wchar_t>`**

The virtual table for the `std::ctype<wchar_t>` class is described by Table 7-64

**Table 7-64 Primary vtable for `ctype<wchar_t>`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeinfo for ctype&lt;wchar_t&gt;</code>
<code>vfunc[0]:</code>	<code>ctype&lt;wchar_t&gt;::~~ctype()</code>
<code>vfunc[1]:</code>	<code>ctype&lt;wchar_t&gt;::~~ctype()</code>
<code>vfunc[2]:</code>	<code>ctype&lt;wchar_t&gt;::do_is(unsigned short, wchar_t) const</code>
<code>vfunc[3]:</code>	<code>ctype&lt;wchar_t&gt;::do_is(wchar_t const*, wchar_t const*, unsigned short*) const</code>
<code>vfunc[4]:</code>	<code>ctype&lt;wchar_t&gt;::do_scan_is(unsigned short, wchar_t const*, wchar_t const*) const</code>
<code>vfunc[5]:</code>	<code>ctype&lt;wchar_t&gt;::do_scan_not(unsigned short, wchar_t const*, wchar_t const*) const</code>
<code>vfunc[6]:</code>	<code>ctype&lt;wchar_t&gt;::do_toupper(wchar_t) const</code>
<code>vfunc[7]:</code>	<code>ctype&lt;wchar_t&gt;::do_toupper(wchar_t*, wchar_t const*) const</code>
<code>vfunc[8]:</code>	<code>ctype&lt;wchar_t&gt;::do_tolower(wchar_t) const</code>
<code>vfunc[9]:</code>	<code>ctype&lt;wchar_t&gt;::do_tolower(wchar_t*, wchar_t const*) const</code>
<code>vfunc[10]:</code>	<code>ctype&lt;wchar_t&gt;::do_widen(char) const</code>
<code>vfunc[11]:</code>	<code>ctype&lt;wchar_t&gt;::do_widen(char</code>

	const*, char const*, wchar_t*) const
vfunc[12]:	ctype<wchar_t>::do_narrow(wchar_t, char) const
vfunc[13]:	ctype<wchar_t>::do_narrow(wchar_t const*, wchar_t const*, char, char*) const

The Run Time Type Information for the std::ctype<wchar\_t> class is described by Table 7-65

**Table 7-65 typeid for ctype<wchar\_t>**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for ctype<wchar_t>

### 7.1.55.2 Interfaces for Class ctype<wchar\_t>

An LSB conforming implementation shall provide the architecture specific methods for Class std::ctype<wchar\_t> specified in Table 7-66, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-66 libstdc++ - Class ctype<wchar\_t> Function Interfaces**

ctype<wchar_t>::ctype(__locale_struct*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::ctype(unsigned int)(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::ctype(__locale_struct*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::ctype(unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.56 Class ctype\_byname<char>

#### 7.1.56.1 Class data for ctype\_byname<char>

The virtual table for the std::ctype\_byname<char> class is described by Table 7-67

**Table 7-67 Primary vtable for ctype\_byname<char>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for ctype_byname<char>
vfunc[0]:	ctype_byname<char>::~~ctype_byname()
vfunc[1]:	ctype_byname<char>::~~ctype_byname()
vfunc[2]:	ctype<char>::do_toupper(char) const
vfunc[3]:	ctype<char>::do_toupper(char*, char

	const*) const
vfunc[4]:	ctype<char>::do_tolower(char) const
vfunc[5]:	ctype<char>::do_tolower(char*, char const*) const
vfunc[6]:	ctype<char>::do_widen(char) const
vfunc[7]:	ctype<char>::do_widen(char const*, char const*, char*) const
vfunc[8]:	ctype<char>::do_narrow(char, char) const
vfunc[9]:	ctype<char>::do_narrow(char const*, char const*, char, char*) const

The Run Time Type Information for the `std::ctype_byname<char>` class is described by Table 7-68

**Table 7-68 typeinfo for `ctype_byname<char>`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>ctype_byname&lt;char&gt;</code>

### 7.1.56.2 Interfaces for Class `ctype_byname<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::ctype_byname<char>` specified in Table 7-69, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-69 libstdcxx - Class `ctype_byname<char>` Function Interfaces**

<code>ctype_byname&lt;char&gt;::ctype_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>ctype_byname&lt;char&gt;::ctype_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.57 Class `ctype_byname<wchar_t>`

#### 7.1.57.1 Interfaces for Class `ctype_byname<wchar_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::ctype_byname<wchar_t>` specified in Table 7-70, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-70 libstdcxx - Class `ctype_byname<wchar_t>` Function Interfaces**

<code>ctype_byname&lt;wchar_t&gt;::ctype_byname(char const*, unsigned int)(GLIBCXX_3.4) [CXXABI]</code>
<code>ctype_byname&lt;wchar_t&gt;::ctype_byname(char const*, unsigned int)(GLIBCXX_3.4) [CXXABI]</code>



## 7.1.58 Class `basic_string<char, char_traits<char>, allocator<char> >`

### 7.1.58.1 Interfaces for Class `basic_string<char, char_traits<char>, allocator<char> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_string<char, std::char_traits<char>, std::allocator<char> >` specified in Table 7-71, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-71 `libstdcxx` - Class `basic_string<char, char_traits<char>, allocator<char> >` Function Interfaces**

<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_last_of(char const*, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_last_of(char const*, unsigned int, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_last_of(basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_last_of(char, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_first_of(char const*, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_first_of(char const*, unsigned int, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_first_of(basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_first_of(char, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_last_not_of(char const*, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_last_not_of(char const*, unsigned int, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_last_not_of(basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_last_not_of(char, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_first_not_of(char const*, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_first_not_of(char const*, unsigned int, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;</code>

<code>&gt;::find_first_not_of(basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find_first_not_of(char, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::at(unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::copy(char*, unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find(char const*, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find(char const*, unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find(basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::find(char, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::rfind(char const*, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::rfind(char const*, unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::rfind(basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::rfind(char, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::substr(unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::compare(unsigned int, unsigned int, char const*) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::compare(unsigned int, unsigned int, char const*, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::compare(unsigned int, unsigned int, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::compare(unsigned int, unsigned int, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::_M_check(unsigned int, char const*) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::_M_limit(unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::operator[](unsigned</code>

int) const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::_S_construct(unsigned int, char, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::_M_replace_aux(unsigned int, unsigned int, unsigned int, char)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::_M_replace_safe(unsigned int, unsigned int, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::at(unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::_Rep::_M_clone(allocator<char> const&, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::_Rep::_S_create(unsigned int, unsigned int, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::erase(unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::append(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::append(basic_string<char, char_traits<char>, allocator<char>> const&, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::append(unsigned int, char)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::assign(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::assign(basic_string<char, char_traits<char>, allocator<char>> const&, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::assign(unsigned int, char)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::insert(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char>>>, unsigned int, char)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::insert(unsigned int, char const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::insert(unsigned int, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>>::insert(unsigned int, basic_string<char, char_traits<char>, allocator<char>> >

<code>const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::insert(unsigned int, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::insert(unsigned int, unsigned int, char)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::resize(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::resize(unsigned int, char)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::replace(__gnu_cxx::__normal_iterator&lt;char*, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; &gt;, __gnu_cxx::__normal_iterator&lt;char*, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; &gt;, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::replace(__gnu_cxx::__normal_iterator&lt;char*, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; &gt;, __gnu_cxx::__normal_iterator&lt;char*, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; &gt;, unsigned int, char)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::replace(unsigned int, unsigned int, char const*)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::replace(unsigned int, unsigned int, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::replace(unsigned int, unsigned int, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::replace(unsigned int, unsigned int, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::replace(unsigned int, unsigned int, unsigned int, char)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::reserve(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::_M_mutate(unsigned int, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::basic_string(char const*, unsigned int, allocator&lt;char&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::basic_string(basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::basic_string(basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, unsigned int, unsigned int, allocator&lt;char&gt; const&amp;)(GLIBCXX_3.4)</code>

[ISOCXX]
basic_string<char, char_traits<char>, allocator<char>> >::basic_string(unsigned int, char, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>> >::basic_string(char const*, unsigned int, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>> >::basic_string(basic_string<char, char_traits<char>, allocator<char>> > const&, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>> >::basic_string(basic_string<char, char_traits<char>, allocator<char>> > const&, unsigned int, unsigned int, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>> >::basic_string(unsigned int, char, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char>> >::operator[](unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.59 Class basic\_string<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t>>

#### 7.1.59.1 Interfaces for Class basic\_string<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t>>

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_string<wchar\_t, std::char\_traits<wchar\_t>, std::allocator<wchar\_t>> specified in Table 7-72, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-72 libstdcxx - Class basic\_string<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t>> Function Interfaces**

basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> >::find_last_of(wchar_t const*, unsigned int) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> >::find_last_of(wchar_t const*, unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> >::find_last_of(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> > const&, unsigned int) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> >::find_last_of(wchar_t, unsigned int) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> >::find_first_of(wchar_t const*, unsigned int) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> >::find_first_of(wchar_t const*, unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]

<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find_first_of(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find_first_of(wchar_t, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find_last_not_of(wchar_t const*, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find_last_not_of(wchar_t const*, unsigned int, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find_last_not_of(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find_last_not_of(wchar_t, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find_first_not_of(wchar_t const*, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find_first_not_of(wchar_t const*, unsigned int, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find_first_not_of(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find_first_not_of(wchar_t, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::at(unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::copy(wchar_t*, unsigned int, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find(wchar_t const*, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find(wchar_t const*, unsigned int, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::find(wchar_t, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::rfind(wchar_t const*, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;</code>

<code>&gt;::rfind(wchar_t const*, unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::rfind(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::rfind(wchar_t, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::substr(unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::compare(unsigned int, unsigned int, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::compare(unsigned int, unsigned int, wchar_t const*, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::compare(unsigned int, unsigned int, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::compare(unsigned int, unsigned int, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;, unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::_M_check(unsigned int, char const*) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::_M_limit(unsigned int, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::operator[](unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::_S_construct(unsigned int, wchar_t, allocator&lt;wchar_t&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::_M_replace_aux(unsigned int, unsigned int, unsigned int, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::_M_replace_safe(unsigned int, unsigned int, wchar_t const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::at(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::_Rep::_M_clone(allocator&lt;wchar_t&gt; const&amp;, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; &gt;::_Rep::_S_create(unsigned int, unsigned int, allocator&lt;wchar_t&gt;</code>

<code>const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::erase(unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::append(wchar_t const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::append(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::append(unsigned int, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::assign(wchar_t const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::assign(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::assign(unsigned int, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::insert(__gnu_cxx::__normal_iterator&lt;wchar_t*, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;&gt;, unsigned int, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::insert(unsigned int, wchar_t const*)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::insert(unsigned int, wchar_t const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::insert(unsigned int, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::insert(unsigned int, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt; const&amp;, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::insert(unsigned int, unsigned int, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::resize(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::resize(unsigned int, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::replace(__gnu_cxx::__normal_iterator&lt;wchar_t*, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;&gt;, __gnu_cxx::__normal_iterator&lt;wchar_t*, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;&gt;, wchar_t const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>



<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::replace(__gnu_cxx::__normal_iterator&lt;wchar_t*, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt; &gt;, __gnu_cxx::__normal_iterator&lt;wchar_t*, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt; &gt;, unsigned int, wchar_t)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::replace(unsigned int, unsigned int, wchar_t const*)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::replace(unsigned int, unsigned int, wchar_t const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::replace(unsigned int, unsigned int, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::replace(unsigned int, unsigned int, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt; const&amp;, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::replace(unsigned int, unsigned int, unsigned int, wchar_t)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::reserve(unsigned int)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::_M_mutate(unsigned int, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::basic_string(wchar_t const*, unsigned int, allocator&lt;wchar_t&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::basic_string(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt; const&amp;, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::basic_string(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt; const&amp;, unsigned int, unsigned int, allocator&lt;wchar_t&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::basic_string(unsigned int, wchar_t, allocator&lt;wchar_t&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::basic_string(wchar_t const*, unsigned int, allocator&lt;wchar_t&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;</pre>

<code>&gt;::basic_string(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt; const&amp;, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::basic_string(basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt; const&amp;, unsigned int, unsigned int, allocator&lt;wchar_t&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::basic_string(unsigned int, wchar_t, allocator&lt;wchar_t&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::operator[](unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.60 Class `basic_stringstream<char, char_traits<char>, allocator<char> >`

#### 7.1.60.1 Class data for `basic_stringstream<char, char_traits<char>, allocator<char> >`

The virtual table for the `std::basic_stringstream<char, std::char_traits<char>, std::allocator<char> >` class is described by Table 7-73

**Table 7-73 Primary vtable for `basic_stringstream<char, char_traits<char>, allocator<char> >`**

Base Offset	0
Virtual Base Offset	52
RTTI	<code>typeid for basic_stringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;</code>
<code>vfunc[0]:</code>	<code>basic_stringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~~basic_stringstream()</code>
<code>vfunc[1]:</code>	<code>basic_stringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~~basic_stringstream()</code>

**Table 7-74 Secondary vtable for `basic_stringstream<char, char_traits<char>, allocator<char> >`**

Base Offset	-8
Virtual Base Offset	44
RTTI	<code>typeid for basic_stringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;</code>
<code>vfunc[0]:</code>	<code>non-virtual thunk to basic_stringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~~basic_stringstream()</code>
<code>vfunc[1]:</code>	<code>non-virtual thunk to basic_stringstream&lt;char,</code>

	char_traits<char>, allocator<char> >::~~basic_stringstream()
--	---

**Table 7-75 Secondary vtable for basic\_stringstream<char, char\_traits<char>, allocator<char> >**

Base Offset	-52
Virtual Base Offset	-52
RTTI	typeinfo for basic_stringstream<char, char_traits<char>, allocator<char> >
vfunc[0]:	virtual thunk to basic_stringstream<char, char_traits<char>, allocator<char> >::~~basic_stringstream()
vfunc[1]:	virtual thunk to basic_stringstream<char, char_traits<char>, allocator<char> >::~~basic_stringstream()

The VTT for the std::basic\_stringstream<char, std::char\_traits<char>, std::allocator<char> > class is described by Table 7-76

**Table 7-76 VTT for basic\_stringstream<char, char\_traits<char>, allocator<char> >**

VTT Name	_ZTTSt18basic_stringstreamIcSt11char_traitsIcESaIcEE
Number of Entries	10

### 7.1.60.2 Interfaces for Class basic\_stringstream<char, char\_traits<char>, allocator<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_stringstream<char, std::char\_traits<char>, std::allocator<char> > specified in Table 7-77, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-77 libstdcxx - Class basic\_stringstream<char, char\_traits<char>, allocator<char> > Function Interfaces**

non-virtual thunk to basic_stringstream<char, char_traits<char>, allocator<char> >::~~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]
non-virtual thunk to basic_stringstream<char, char_traits<char>, allocator<char> >::~~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_stringstream<char, char_traits<char>, allocator<char> >::~~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_stringstream<char, char_traits<char>, allocator<char> >::~~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]

### 7.1.61 Class `basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

#### 7.1.61.1 Class data for `basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

The virtual table for the `std::basic_stringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >` class is described by Table 7-78

**Table 7-78 Primary vtable for `basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`**

Base Offset	0
Virtual Base Offset	52
RTTI	typeid for <code>basic_stringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;</code>
vfunc[0]:	<code>basic_stringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::~~basic_stringstream()</code>
vfunc[1]:	<code>basic_stringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::~~basic_stringstream()</code>

**Table 7-79 Secondary vtable for `basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`**

Base Offset	-8
Virtual Base Offset	44
RTTI	typeid for <code>basic_stringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;</code>
vfunc[0]:	non-virtual thunk to <code>basic_stringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::~~basic_stringstream()</code>
vfunc[1]:	non-virtual thunk to <code>basic_stringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::~~basic_stringstream()</code>

**Table 7-80 Secondary vtable for basic\_stringstream<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> >**

Base Offset	-52
Virtual Base Offset	-52
RTTI	typeid for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >
vfunc[0]:	virtual thunk to basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()
vfunc[1]:	virtual thunk to basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()

The VTT for the std::basic\_stringstream<wchar\_t, std::char\_traits<wchar\_t>, std::allocator<wchar\_t> > class is described by Table 7-81

**Table 7-81 VTT for basic\_stringstream<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> >**

VTT Name	_ZTTSt18basic_stringstreamIwSt11char_traitsIwESaIwEE
Number of Entries	10

### 7.1.61.2 Interfaces for Class basic\_stringstream<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_stringstream<wchar\_t, std::char\_traits<wchar\_t>, std::allocator<wchar\_t> > specified in Table 7-82, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-82 libstdcxx - Class basic\_stringstream<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> > Function Interfaces**

non-virtual thunk to basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]
non-virtual thunk to basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]

### 7.1.62 Class `basic_istringstream<char, char_traits<char>, allocator<char> >`

#### 7.1.62.1 Class data for `basic_istringstream<char, char_traits<char>, allocator<char> >`

The virtual table for the `std::basic_istringstream<char, std::char_traits<char>, std::allocator<char> >` class is described by Table 7-83

**Table 7-83 Primary vtable for `basic_istringstream<char, char_traits<char>, allocator<char> >`**

Base Offset	0
Virtual Base Offset	48
RTTI	typeid for <code>basic_istringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;</code>
vfunc[0]:	<code>basic_istringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~~basic_istringstream()</code>
vfunc[1]:	<code>basic_istringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~~basic_istringstream()</code>

**Table 7-84 Secondary vtable for `basic_istringstream<char, char_traits<char>, allocator<char> >`**

Base Offset	-48
Virtual Base Offset	-48
RTTI	typeid for <code>basic_istringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;</code>
vfunc[0]:	virtual thunk to <code>basic_istringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~~basic_istringstream()</code>
vfunc[1]:	virtual thunk to <code>basic_istringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~~basic_istringstream()</code>

The VTT for the `std::basic_istringstream<char, std::char_traits<char>, std::allocator<char> >` class is described by Table 7-85

**Table 7-85 VTT for `basic_istringstream<char, char_traits<char>, allocator<char> >`**

VTT Name	<code>_ZTTSt19basic_istringstreamIcSt11char_traitsIcESaIcEE</code>
Number of Entries	4

### 7.1.62.2 Interfaces for Class `basic_istream<char, char_traits<char>, allocator<char>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_istream<char, std::char_traits<char>, std::allocator<char>>` specified in Table 7-86, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-86 libstdc++ - Class `basic_istream<char, char_traits<char>, allocator<char>>` Function Interfaces**

virtual thunk to <code>basic_istream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt;&gt;::~basic_istream()</code> (GLIBCXX_3.4) [ISOCXX]
virtual thunk to <code>basic_istream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt;&gt;::~basic_istream()</code> (GLIBCXX_3.4) [ISOCXX]

### 7.1.63 Class `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`

#### 7.1.63.1 Class data for `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`

The virtual table for the `std::basic_istream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>>` class is described by Table 7-87

**Table 7-87 Primary vtable for `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`**

Base Offset	0
Virtual Base Offset	48
RTTI	typeid for <code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;</code>
vfunc[0]:	<code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::~basic_istream()</code>
vfunc[1]:	<code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::~basic_istream()</code>

**Table 7-88 Secondary vtable for `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`**

Base Offset	-48
Virtual Base Offset	-48
RTTI	typeid for <code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;</code>

	allocator<wchar_t> >
vfunc[0]:	virtual thunk to basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_istream()
vfunc[1]:	virtual thunk to basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_istream()

The VTT for the `std::basic_istream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >` class is described by Table 7-89

**Table 7-89 VTT for `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`**

VTT Name	<code>_ZTTSt19basic_istreamlwSt11char_traitslwESaIwEE</code>
Number of Entries	4

### 7.1.63.2 Interfaces for Class `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_istream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >` specified in Table 7-90, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-90 `libstdcxx` - Class `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >` Function Interfaces**

virtual thunk to <code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::~basic_istream()(GLIBCXX_3.4) [ISOCXX]</code>
virtual thunk to <code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;::~basic_istream()(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.64 Class `basic_ostringstream<char, char_traits<char>, allocator<char> >`

#### 7.1.64.1 Class data for `basic_ostringstream<char, char_traits<char>, allocator<char> >`

The virtual table for the `std::basic_ostringstream<char, std::char_traits<char>, std::allocator<char> >` class is described by Table 7-91

**Table 7-91 Primary vtable for `basic_ostringstream<char, char_traits<char>, allocator<char> >`**

Base Offset	0
Virtual Base Offset	44
RTTI	typeid for



	<code>basic_ostringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;</code>
<code>vfunc[0]:</code>	<code>basic_ostringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~basic_ostringstream()</code>
<code>vfunc[1]:</code>	<code>basic_ostringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~basic_ostringstream()</code>

**Table 7-92 Secondary vtable for `basic_ostringstream<char, char_traits<char>, allocator<char> >`**

Base Offset	-44
Virtual Base Offset	-44
RTTI	typeinfo for <code>basic_ostringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;</code>
<code>vfunc[0]:</code>	virtual thunk to <code>basic_ostringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~basic_ostringstream()</code>
<code>vfunc[1]:</code>	virtual thunk to <code>basic_ostringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~basic_ostringstream()</code>

The VTT for the `std::basic_ostringstream<char, std::char_traits<char>, std::allocator<char> >` class is described by Table 7-93

**Table 7-93 VTT for `basic_ostringstream<char, char_traits<char>, allocator<char> >`**

VTT Name	<code>_ZTTSt19basic_ostringstreamIcSt11char_traitsIcESaIcEE</code>
Number of Entries	4

### 7.1.64.2 Interfaces for Class `basic_ostringstream<char, char_traits<char>, allocator<char> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_ostringstream<char, std::char_traits<char>, std::allocator<char> >` specified in Table 7-94, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-94 `libstdcxx` - Class `basic_ostringstream<char, char_traits<char>, allocator<char> >` Function Interfaces**

<code>virtual thunk to <code>basic_ostringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~basic_ostringstream()(GLIBCXX_3.4)</code> [ISOCXX]</code>
<code>virtual thunk to <code>basic_ostringstream&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;::~basic_ostringstream()(GLIBCXX_3.4)</code> [ISOCXX]</code>

### 7.1.65 Class `basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`

#### 7.1.65.1 Class data for `basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`

The virtual table for the `std::basic_ostringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>>` class is described by Table 7-95

**Table 7-95 Primary vtable for `basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`**

Base Offset	0
Virtual Base Offset	44
RTTI	typeid for <code>basic_ostringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;</code>
vfunc[0]:	<code>basic_ostringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::~~basic_ostringstream()</code>
vfunc[1]:	<code>basic_ostringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::~~basic_ostringstream()</code>

**Table 7-96 Secondary vtable for `basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`**

Base Offset	-44
Virtual Base Offset	-44
RTTI	typeid for <code>basic_ostringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;</code>
vfunc[0]:	virtual thunk to <code>basic_ostringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::~~basic_ostringstream()</code>
vfunc[1]:	virtual thunk to <code>basic_ostringstream&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt;&gt;::~~basic_ostringstream()</code>

The VTT for the `std::basic_ostringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>>` class is described by Table 7-97

**Table 7-97 VTT for basic\_ostringstream<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> >**

VTT Name	_ZTTSt19basic_ostringstreamIwSt11char_traitsIwESaIwEE
Number of Entries	4

### 7.1.65.2 Interfaces for Class basic\_ostringstream<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_ostringstream<wchar\_t, std::char\_traits<wchar\_t>, std::allocator<wchar\_t> > specified in Table 7-98, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-98 libstdcxx - Class basic\_ostringstream<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> > Function Interfaces**

virtual thunk to basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~~basic_ostringstream()(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~~basic_ostringstream()(GLIBCXX_3.4) [ISOCXX]

### 7.1.66 Class basic\_stringbuf<char, char\_traits<char>, allocator<char> >

#### 7.1.66.1 Class data for basic\_stringbuf<char, char\_traits<char>, allocator<char> >

The virtual table for the std::basic\_stringbuf<char, std::char\_traits<char>, std::allocator<char> > class is described by Table 7-99

**Table 7-99 Primary vtable for basic\_stringbuf<char, char\_traits<char>, allocator<char> >**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for basic_stringbuf<char, char_traits<char>, allocator<char> >
vfunc[0]:	basic_stringbuf<char, char_traits<char>, allocator<char> >::~~basic_stringbuf()
vfunc[1]:	basic_stringbuf<char, char_traits<char>, allocator<char> >::~~basic_stringbuf()
vfunc[2]:	basic_streambuf<char, char_traits<char> >::imbue(locale const&)
vfunc[3]:	basic_stringbuf<char, char_traits<char>, allocator<char> >::~setbuf(char*, int)

vfunc[4]:	basic_stringbuf<char, char_traits<char>, allocator<char>>::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)
vfunc[5]:	basic_stringbuf<char, char_traits<char>, allocator<char>>::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	basic_streambuf<char, char_traits<char>>::sync()
vfunc[7]:	basic_streambuf<char, char_traits<char>>::showmanyc()
vfunc[8]:	basic_streambuf<char, char_traits<char>>::xsgetn(char*, int)
vfunc[9]:	basic_stringbuf<char, char_traits<char>, allocator<char>>::underflow()
vfunc[10]:	basic_streambuf<char, char_traits<char>>::uflow()
vfunc[11]:	basic_stringbuf<char, char_traits<char>, allocator<char>>::pbackfail(int)
vfunc[12]:	basic_streambuf<char, char_traits<char>>::xsputn(char const*, int)
vfunc[13]:	basic_stringbuf<char, char_traits<char>, allocator<char>>::overflow(int)

The Run Time Type Information for the `std::basic_stringbuf<char, std::char_traits<char>, std::allocator<char>>` class is described by Table 7-100

**Table 7-100 typeid for basic\_stringbuf<char, char\_traits<char>, allocator<char>>**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>basic_stringbuf&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt;&gt;</code>

### 7.1.66.2 Interfaces for Class `basic_stringbuf<char, char_traits<char>, allocator<char>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_stringbuf<char, std::char_traits<char>, std::allocator<char>>` specified in Table 7-101, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-101 libstdcxx - Class basic\_stringbuf<char, char\_traits<char>, allocator<char> > Function Interfaces**

basic_stringbuf<char, char_traits<char>, allocator<char> >::setbuf(char*, int)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::_M_sync(char*, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]

**7.1.67 Class basic\_stringbuf<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> >****7.1.67.1 Class data for basic\_stringbuf<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> >**

The virtual table for the std::basic\_stringbuf<wchar\_t, std::char\_traits<wchar\_t>, std::allocator<wchar\_t> > class is described by Table 7-102

**Table 7-102 Primary vtable for basic\_stringbuf<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> >**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >
vfunc[0]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~~basic_stringbuf()
vfunc[1]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~~basic_stringbuf()
vfunc[2]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::imbue(locale const&)
vfunc[3]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::setbuf(wchar_t*, int)
vfunc[4]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)
vfunc[5]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)

	allocator<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::sync()
vfunc[7]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::showmanyc()
vfunc[8]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::xsgetn(wchar_t*, int)
vfunc[9]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::underflow()
vfunc[10]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::uflow()
vfunc[11]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::pbackfail(unsigned int)
vfunc[12]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::xsputn(wchar_t const*, int)
vfunc[13]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::overflow(unsigned int)

The Run Time Type Information for the `std::basic_stringbuf<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >` class is described by Table 7-103

**Table 7-103** typeid for `basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

### 7.1.67.2 Interfaces for Class `basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_stringbuf<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >` specified in Table 7-104, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-104 libstdcxx - Class basic\_stringbuf<wchar\_t, char\_traits<wchar\_t>, allocator<wchar\_t> > Function Interfaces**

basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::setbuf(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::_M_sync(wchar_t*, unsigned int, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]

**7.1.68 Class basic\_istream<char, char\_traits<char> >****7.1.68.1 Class data for basic\_istream<char, char\_traits<char> >**

The virtual table for the std::basic\_istream<char, std::char\_traits<char> > class is described by Table 7-105

**Table 7-105 Primary vtable for basic\_istream<char, char\_traits<char> >**

Base Offset	0
Virtual Base Offset	12
RTTI	typeid for basic_istream<char, char_traits<char> >
vfunc[0]:	basic_istream<char, char_traits<char>>::~~basic_istream()
vfunc[1]:	basic_istream<char, char_traits<char>>::~~basic_istream()

**Table 7-106 Secondary vtable for basic\_istream<char, char\_traits<char> >**

Base Offset	-8
Virtual Base Offset	4
RTTI	typeid for basic_istream<char, char_traits<char> >
vfunc[0]:	non-virtual thunk to basic_istream<char, char_traits<char>>::~~basic_istream()
vfunc[1]:	non-virtual thunk to basic_istream<char, char_traits<char>>::~~basic_istream()

**Table 7-107 Secondary vtable for basic\_istream<char, char\_traits<char> >**

Base Offset	-12
-------------	-----

Virtual Base Offset	-12
RTTI	typeid for basic_ostream<char, char_traits<char> >
vfunc[0]:	virtual thunk to basic_ostream<char, char_traits<char> >::~basic_ostream()
vfunc[1]:	virtual thunk to basic_ostream<char, char_traits<char> >::~basic_ostream()

The VTT for the std::basic\_ostream<char, std::char\_traits<char> > class is described by Table 7-108

**Table 7-108 VTT for basic\_ostream<char, char\_traits<char> >**

VTT Name	_ZTTSd
Number of Entries	7

### 7.1.68.2 Interfaces for Class basic\_ostream<char, char\_traits<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_ostream<char, std::char\_traits<char> > specified in Table 7-109, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-109 libstdc++ - Class basic\_ostream<char, char\_traits<char> > Function Interfaces**

non-virtual thunk to basic_ostream<char, char_traits<char> >::~basic_ostream()(GLIBCXX_3.4) [CXXABI]
non-virtual thunk to basic_ostream<char, char_traits<char> >::~basic_ostream()(GLIBCXX_3.4) [CXXABI]
virtual thunk to basic_ostream<char, char_traits<char> >::~basic_ostream()(GLIBCXX_3.4) [CXXABI]
virtual thunk to basic_ostream<char, char_traits<char> >::~basic_ostream()(GLIBCXX_3.4) [CXXABI]

### 7.1.69 Class basic\_ostream<wchar\_t, char\_traits<wchar\_t> >

#### 7.1.69.1 Class data for basic\_ostream<wchar\_t, char\_traits<wchar\_t> >

The virtual table for the std::basic\_ostream<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-110

**Table 7-110 Primary vtable for basic\_ostream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	0
Virtual Base Offset	12



RTTI	typeid for basic_istream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_istream<wchar_t, char_traits<wchar_t> >::~~basic_istream()
vfunc[1]:	basic_istream<wchar_t, char_traits<wchar_t> >::~~basic_istream()

**Table 7-111 Secondary vtable for basic\_istream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	-8
Virtual Base Offset	4
RTTI	typeid for basic_istream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	non-virtual thunk to basic_istream<wchar_t, char_traits<wchar_t> >::~~basic_istream()
vfunc[1]:	non-virtual thunk to basic_istream<wchar_t, char_traits<wchar_t> >::~~basic_istream()

**Table 7-112 Secondary vtable for basic\_istream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	-12
Virtual Base Offset	-12
RTTI	typeid for basic_istream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	virtual thunk to basic_istream<wchar_t, char_traits<wchar_t> >::~~basic_istream()
vfunc[1]:	virtual thunk to basic_istream<wchar_t, char_traits<wchar_t> >::~~basic_istream()

The VTT for the std::basic\_istream<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-113

**Table 7-113 VTT for basic\_istream<wchar\_t, char\_traits<wchar\_t> >**

VTT Name	_ZTTSt14basic_istreamIwSt11char_traitsIwEE
----------	--

Number of Entries	7
-------------------	---

### 7.1.69.2 Interfaces for Class `basic_iostream<wchar_t, char_traits<wchar_t>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_iostream<wchar_t, std::char_traits<wchar_t>>` specified in Table 7-114, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-114 libstdcxx - Class `basic_iostream<wchar_t, char_traits<wchar_t>>` Function Interfaces**

non-virtual thunk to <code>basic_iostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_iostream()</code> (GLIBCXX_3.4) [ISOCXX]
non-virtual thunk to <code>basic_iostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_iostream()</code> (GLIBCXX_3.4) [ISOCXX]
virtual thunk to <code>basic_iostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_iostream()</code> (GLIBCXX_3.4) [ISOCXX]
virtual thunk to <code>basic_iostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_iostream()</code> (GLIBCXX_3.4) [ISOCXX]

### 7.1.70 Class `basic_istream<char, char_traits<char>>`

#### 7.1.70.1 Class data for `basic_istream<char, char_traits<char>>`

The virtual table for the `std::basic_istream<char, std::char_traits<char>>` class is described by Table 7-115

**Table 7-115 Primary vtable for `basic_istream<char, char_traits<char>>`**

Base Offset	0
Virtual Base Offset	8
RTTI	<code>typeinfo for basic_istream&lt;char, char_traits&lt;char&gt;&gt;</code>
<code>vfunc[0]:</code>	<code>basic_istream&lt;char, char_traits&lt;char&gt;&gt;::~basic_istream()</code>
<code>vfunc[1]:</code>	<code>basic_istream&lt;char, char_traits&lt;char&gt;&gt;::~basic_istream()</code>

**Table 7-116 Secondary vtable for `basic_istream<char, char_traits<char>>`**

Base Offset	-8
Virtual Base Offset	-8
RTTI	<code>typeinfo for basic_istream&lt;char, char_traits&lt;char&gt;&gt;</code>
<code>vfunc[0]:</code>	<code>virtual thunk to basic_istream&lt;char, char_traits&lt;char&gt;&gt;::~basic_istream()</code>
<code>vfunc[1]:</code>	<code>virtual thunk to basic_istream&lt;char,</code>

	char_traits<char> >::~basic_istream()
--	---------------------------------------

The VTT for the std::basic\_istream<char, std::char\_traits<char> > class is described by Table 7-117

**Table 7-117 VTT for basic\_istream<char, char\_traits<char> >**

VTT Name	_ZTTSi
Number of Entries	2

### 7.1.70.2 Interfaces for Class basic\_istream<char, char\_traits<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_istream<char, std::char\_traits<char> > specified in Table 7-118, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-118 libstdcxx - Class basic\_istream<char, char\_traits<char> > Function Interfaces**

basic_istream<char, char_traits<char> >::get(char*, int)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::get(char*, int, char)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::read(char*, int)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::seekg(long long, _Ios_Seekdir)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::ignore(int, int)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::getline(char*, int)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::getline(char*, int, char)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::readsome(char*, int)(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_istream<char, char_traits<char> >::~basic_istream()(GLIBCXX_3.4) [CXXABI]
virtual thunk to basic_istream<char, char_traits<char> >::~basic_istream()(GLIBCXX_3.4) [CXXABI]

### 7.1.71 Class basic\_istream<wchar\_t, char\_traits<wchar\_t> >

#### 7.1.71.1 Class data for basic\_istream<wchar\_t, char\_traits<wchar\_t> >

The virtual table for the std::basic\_istream<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-119

**Table 7-119 Primary vtable for basic\_istream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	0
Virtual Base Offset	8
RTTI	typeid for basic_istream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_istream<wchar_t, char_traits<wchar_t> >::~~basic_istream()
vfunc[1]:	basic_istream<wchar_t, char_traits<wchar_t> >::~~basic_istream()

**Table 7-120 Secondary vtable for basic\_istream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	-8
Virtual Base Offset	-8
RTTI	typeid for basic_istream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	virtual thunk to basic_istream<wchar_t, char_traits<wchar_t> >::~~basic_istream()
vfunc[1]:	virtual thunk to basic_istream<wchar_t, char_traits<wchar_t> >::~~basic_istream()

The VTT for the std::basic\_istream<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-121

**Table 7-121 VTT for basic\_istream<wchar\_t, char\_traits<wchar\_t> >**

VTT Name	_ZTTSt13basic_istreamIwSt11char_traitsIwEE
Number of Entries	2

### 7.1.71.2 Interfaces for Class basic\_istream<wchar\_t, char\_traits<wchar\_t> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_istream<wchar\_t, std::char\_traits<wchar\_t> > specified in Table 7-122, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-122 libstdcxx - Class basic\_istream<wchar\_t, char\_traits<wchar\_t> > Function Interfaces**

basic_istream<wchar_t, char_traits<wchar_t> >::get(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]
---

<code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::get(wchar_t*, int, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::read(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::ignore(int, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::getline(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::getline(wchar_t*, int, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::readsome(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>virtual thunk to basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::~basic_istream()(GLIBCXX_3.4) [ISOCXX]</code>
<code>virtual thunk to basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::~basic_istream()(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.72 Class `istreambuf_iterator<wchar_t, char_traits<wchar_t> >`

#### 7.1.72.1 Interfaces for Class `istreambuf_iterator<wchar_t, char_traits<wchar_t> >`

No external methods are defined for `libstdc++` - Class `std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> >` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.73 Class `istreambuf_iterator<char, char_traits<char> >`

#### 7.1.73.1 Interfaces for Class `istreambuf_iterator<char, char_traits<char> >`

No external methods are defined for `libstdc++` - Class `std::istreambuf_iterator<char, std::char_traits<char> >` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.74 Class `basic_ostream<char, char_traits<char> >`

#### 7.1.74.1 Class data for `basic_ostream<char, char_traits<char> >`

The virtual table for the `std::basic_ostream<char, std::char_traits<char> >` class is described by Table 7-123

Table 7-123 Primary vtable for `basic_ostream<char, char_traits<char> >`

Base Offset	0
Virtual Base Offset	4
RTTI	<code>typeid for basic_ostream&lt;char, char_traits&lt;char&gt; &gt;</code>
<code>vfunc[0]:</code>	<code>basic_ostream&lt;char, char_traits&lt;char&gt;</code>

	>::~~basic_ostream()
vfunc[1]:	basic_ostream<char, char_traits<char> >::~~basic_ostream()

**Table 7-124 Secondary vtable for basic\_ostream<char, char\_traits<char> >**

Base Offset	-4
Virtual Base Offset	-4
RTTI	typeid for basic_ostream<char, char_traits<char> >
vfunc[0]:	virtual thunk to basic_ostream<char, char_traits<char> >::~~basic_ostream()
vfunc[1]:	virtual thunk to basic_ostream<char, char_traits<char> >::~~basic_ostream()

The VTT for the std::basic\_ostream<char, std::char\_traits<char> > class is described by Table 7-125

**Table 7-125 VTT for basic\_ostream<char, char\_traits<char> >**

VTT Name	_ZTTS0
Number of Entries	2

### 7.1.74.2 Interfaces for Class basic\_ostream<char, char\_traits<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_ostream<char, std::char\_traits<char> > specified in Table 7-126, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-126 libstdcxx - Class basic\_ostream<char, char\_traits<char> > Function Interfaces**

basic_ostream<char, char_traits<char> >::seekp(long long, _Ios_Seekdir)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::write(char const*, int)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::_M_write(char const*, int)(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_ostream<char, char_traits<char> >::~~basic_ostream()(GLIBCXX_3.4) [CXXABI]
virtual thunk to basic_ostream<char, char_traits<char> >::~~basic_ostream()(GLIBCXX_3.4) [CXXABI]

### 7.1.75 Class `basic_ostream<wchar_t, char_traits<wchar_t>>`

#### 7.1.75.1 Class data for `basic_ostream<wchar_t, char_traits<wchar_t>>`

The virtual table for the `std::basic_ostream<wchar_t, std::char_traits<wchar_t>>` class is described by Table 7-127

**Table 7-127 Primary vtable for `basic_ostream<wchar_t, char_traits<wchar_t>>`**

Base Offset	0
Virtual Base Offset	4
RTTI	typeid for <code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;</code>
vfunc[0]:	<code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~~basic_ostream()</code>
vfunc[1]:	<code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~~basic_ostream()</code>

**Table 7-128 Secondary vtable for `basic_ostream<wchar_t, char_traits<wchar_t>>`**

Base Offset	-4
Virtual Base Offset	-4
RTTI	typeid for <code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;</code>
vfunc[0]:	virtual thunk to <code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~~basic_ostream()</code>
vfunc[1]:	virtual thunk to <code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~~basic_ostream()</code>

The VTT for the `std::basic_ostream<wchar_t, std::char_traits<wchar_t>>` class is described by Table 7-129

**Table 7-129 VTT for `basic_ostream<wchar_t, char_traits<wchar_t>>`**

VTT Name	<code>_ZTTSt13basic_ostreamIwSt11char_traitsIwEE</code>
Number of Entries	2

### 7.1.75.2 Interfaces for Class `basic_ostream<wchar_t, char_traits<wchar_t>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_ostream<wchar_t, std::char_traits<wchar_t>>` specified in Table 7-130, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-130 libstdcxx - Class `basic_ostream<wchar_t, char_traits<wchar_t>>` Function Interfaces**

<code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::write(wchar_t const*, int)(GLIBCXX_3.4) [ISOCXX]</code>
virtual thunk to <code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_ostream()(GLIBCXX_3.4) [ISOCXX]</code>
virtual thunk to <code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_ostream()(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.76 Class `basic_fstream<char, char_traits<char>>`

#### 7.1.76.1 Class data for `basic_fstream<char, char_traits<char>>`

The virtual table for the `std::basic_fstream<char, std::char_traits<char>>` class is described by Table 7-131

**Table 7-131 Primary vtable for `basic_fstream<char, char_traits<char>>`**

Base Offset	0
Virtual Base Offset	148
RTTI	<code>typeinfo for basic_fstream&lt;char, char_traits&lt;char&gt;&gt;</code>
<code>vfunc[0]:</code>	<code>basic_fstream&lt;char, char_traits&lt;char&gt;&gt;::~basic_fstream()</code>
<code>vfunc[1]:</code>	<code>basic_fstream&lt;char, char_traits&lt;char&gt;&gt;::~basic_fstream()</code>

**Table 7-132 Secondary vtable for `basic_fstream<char, char_traits<char>>`**

Base Offset	-8
Virtual Base Offset	140
RTTI	<code>typeinfo for basic_fstream&lt;char, char_traits&lt;char&gt;&gt;</code>
<code>vfunc[0]:</code>	non-virtual thunk to <code>basic_fstream&lt;char, char_traits&lt;char&gt;&gt;::~basic_fstream()</code>
<code>vfunc[1]:</code>	non-virtual thunk to <code>basic_fstream&lt;char, char_traits&lt;char&gt;&gt;::~basic_fstream()</code>



**Table 7-133 Secondary vtable for basic\_fstream<char, char\_traits<char> >**

Base Offset	-148
Virtual Base Offset	-148
RTTI	typeid for basic_fstream<char, char_traits<char> >
vfunc[0]:	virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()
vfunc[1]:	virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()

The VTT for the std::basic\_fstream<char, std::char\_traits<char> > class is described by Table 7-134

**Table 7-134 VTT for basic\_fstream<char, char\_traits<char> >**

VTT Name	_ZTTSt13basic_fstreamIcSt11char_traitsIcEE
Number of Entries	10

### 7.1.76.2 Interfaces for Class basic\_fstream<char, char\_traits<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_fstream<char, std::char\_traits<char> > specified in Table 7-135, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-135 libstdcxx - Class basic\_fstream<char, char\_traits<char> > Function Interfaces**

non-virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()(GLIBCXX_3.4) [ISOCXX]
non-virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()(GLIBCXX_3.4) [ISOCXX]

### 7.1.77 Class basic\_fstream<wchar\_t, char\_traits<wchar\_t> >

#### 7.1.77.1 Class data for basic\_fstream<wchar\_t, char\_traits<wchar\_t> >

The virtual table for the std::basic\_fstream<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-136

**Table 7-136 Primary vtable for basic\_fstream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	0
-------------	---

Virtual Base Offset	152
RTTI	typeid for basic_fstream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_fstream<wchar_t, char_traits<wchar_t> >::~~basic_fstream()
vfunc[1]:	basic_fstream<wchar_t, char_traits<wchar_t> >::~~basic_fstream()

**Table 7-137 Secondary vtable for basic\_fstream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	-8
Virtual Base Offset	144
RTTI	typeid for basic_fstream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	non-virtual thunk to basic_fstream<wchar_t, char_traits<wchar_t> >::~~basic_fstream()
vfunc[1]:	non-virtual thunk to basic_fstream<wchar_t, char_traits<wchar_t> >::~~basic_fstream()

**Table 7-138 Secondary vtable for basic\_fstream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	-152
Virtual Base Offset	-152
RTTI	typeid for basic_fstream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	virtual thunk to basic_fstream<wchar_t, char_traits<wchar_t> >::~~basic_fstream()
vfunc[1]:	virtual thunk to basic_fstream<wchar_t, char_traits<wchar_t> >::~~basic_fstream()

The VTT for the std::basic\_fstream<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-139

**Table 7-139 VTT for basic\_fstream<wchar\_t, char\_traits<wchar\_t> >**

VTT Name	_ZTTSt13basic_fstreamIwSt11char_tr
----------	------------------------------------

	aitsIwEE
Number of Entries	10

### 7.1.77.2 Interfaces for Class `basic_fstream<wchar_t, char_traits<wchar_t>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_fstream<wchar_t, std::char_traits<wchar_t>>` specified in Table 7-140, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-140 libstdcxx - Class `basic_fstream<wchar_t, char_traits<wchar_t>>` Function Interfaces**

non-virtual thunk to <code>basic_fstream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_fstream()</code> (GLIBCXX_3.4) [ISOCXX]
non-virtual thunk to <code>basic_fstream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_fstream()</code> (GLIBCXX_3.4) [ISOCXX]
virtual thunk to <code>basic_fstream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_fstream()</code> (GLIBCXX_3.4) [ISOCXX]
virtual thunk to <code>basic_fstream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_fstream()</code> (GLIBCXX_3.4) [ISOCXX]

### 7.1.78 Class `basic_ifstream<char, char_traits<char>>`

#### 7.1.78.1 Class data for `basic_ifstream<char, char_traits<char>>`

The virtual table for the `std::basic_ifstream<char, std::char_traits<char>>` class is described by Table 7-141

**Table 7-141 Primary vtable for `basic_ifstream<char, char_traits<char>>`**

Base Offset	0
Virtual Base Offset	144
RTTI	typeinfo for <code>basic_ifstream&lt;char, char_traits&lt;char&gt;&gt;</code>
vfunc[0]:	<code>basic_ifstream&lt;char, char_traits&lt;char&gt;&gt;::~basic_ifstream()</code>
vfunc[1]:	<code>basic_ifstream&lt;char, char_traits&lt;char&gt;&gt;::~basic_ifstream()</code>

**Table 7-142 Secondary vtable for `basic_ifstream<char, char_traits<char>>`**

Base Offset	-144
Virtual Base Offset	-144
RTTI	typeinfo for <code>basic_ifstream&lt;char, char_traits&lt;char&gt;&gt;</code>
vfunc[0]:	virtual thunk to <code>basic_ifstream&lt;char,</code>

	char_traits<char> >::~~basic_ifstream()
vfunc[1]:	virtual thunk to basic_ifstream<char, char_traits<char> >::~~basic_ifstream()

The VTT for the std::basic\_ifstream<char, std::char\_traits<char> > class is described by Table 7-143

**Table 7-143 VTT for basic\_ifstream<char, char\_traits<char> >**

VTT Name	_ZTTSt14basic_ifstreamIcSt11char_traitsIcEE
Number of Entries	4

### 7.1.78.2 Interfaces for Class basic\_ifstream<char, char\_traits<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_ifstream<char, std::char\_traits<char> > specified in Table 7-144, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-144 libstdc++ - Class basic\_ifstream<char, char\_traits<char> > Function Interfaces**

virtual thunk to basic_ifstream<char, char_traits<char> >::~~basic_ifstream()(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_ifstream<char, char_traits<char> >::~~basic_ifstream()(GLIBCXX_3.4) [ISOCXX]

### 7.1.79 Class basic\_ifstream<wchar\_t, char\_traits<wchar\_t> >

#### 7.1.79.1 Class data for basic\_ifstream<wchar\_t, char\_traits<wchar\_t> >

The virtual table for the std::basic\_ifstream<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-145

**Table 7-145 Primary vtable for basic\_ifstream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	0
Virtual Base Offset	148
RTTI	typeid for basic_ifstream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_ifstream<wchar_t, char_traits<wchar_t> >::~~basic_ifstream()
vfunc[1]:	basic_ifstream<wchar_t, char_traits<wchar_t> >::~~basic_ifstream()

**Table 7-146 Secondary vtable for basic\_ifstream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	-148
Virtual Base Offset	-148
RTTI	typeid for basic_ifstream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	virtual thunk to basic_ifstream<wchar_t, char_traits<wchar_t> >::~basic_ifstream()
vfunc[1]:	virtual thunk to basic_ifstream<wchar_t, char_traits<wchar_t> >::~basic_ifstream()

The VTT for the std::basic\_ifstream<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-147

**Table 7-147 VTT for basic\_ifstream<wchar\_t, char\_traits<wchar\_t> >**

VTT Name	_ZTTSt14basic_ifstreamlwSt11char_traitslwEE
Number of Entries	4

### 7.1.79.2 Interfaces for Class basic\_ifstream<wchar\_t, char\_traits<wchar\_t> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_ifstream<wchar\_t, std::char\_traits<wchar\_t> > specified in Table 7-148, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-148 libstdcxx - Class basic\_ifstream<wchar\_t, char\_traits<wchar\_t> > Function Interfaces**

virtual thunk to basic_ifstream<wchar_t, char_traits<wchar_t> >::~basic_ifstream()(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_ifstream<wchar_t, char_traits<wchar_t> >::~basic_ifstream()(GLIBCXX_3.4) [ISOCXX]

### 7.1.80 Class basic\_ofstream<char, char\_traits<char> >

#### 7.1.80.1 Class data for basic\_ofstream<char, char\_traits<char> >

The virtual table for the std::basic\_ofstream<char, std::char\_traits<char> > class is described by Table 7-149

**Table 7-149 Primary vtable for basic\_ofstream<char, char\_traits<char> >**

Base Offset	0
Virtual Base Offset	140

RTTI	typeid for basic_ofstream<char, char_traits<char> >
vfunc[0]:	basic_ofstream<char, char_traits<char> >::~basic_ofstream()
vfunc[1]:	basic_ofstream<char, char_traits<char> >::~basic_ofstream()

**Table 7-150 Secondary vtable for basic\_ofstream<char, char\_traits<char> >**

Base Offset	-140
Virtual Base Offset	-140
RTTI	typeid for basic_ofstream<char, char_traits<char> >
vfunc[0]:	virtual thunk to basic_ofstream<char, char_traits<char> >::~basic_ofstream()
vfunc[1]:	virtual thunk to basic_ofstream<char, char_traits<char> >::~basic_ofstream()

The VTT for the std::basic\_ofstream<char, std::char\_traits<char> > class is described by Table 7-151

**Table 7-151 VTT for basic\_ofstream<char, char\_traits<char> >**

VTT Name	_ZTTSt14basic_ofstreamIcSt11char_traitsIcEE
Number of Entries	4

### 7.1.80.2 Interfaces for Class basic\_ofstream<char, char\_traits<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_ofstream<char, std::char\_traits<char> > specified in Table 7-152, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-152 libstdcxx - Class basic\_ofstream<char, char\_traits<char> > Function Interfaces**

virtual thunk to basic_ofstream<char, char_traits<char> >::~basic_ofstream()(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_ofstream<char, char_traits<char> >::~basic_ofstream()(GLIBCXX_3.4) [ISOCXX]

**7.1.81 Class basic\_ofstream<wchar\_t, char\_traits<wchar\_t> >****7.1.81.1 Class data for basic\_ofstream<wchar\_t, char\_traits<wchar\_t> >**

The virtual table for the std::basic\_ofstream<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-153

**Table 7-153 Primary vtable for basic\_ofstream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	0
Virtual Base Offset	144
RTTI	typeid for basic_ofstream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_ofstream<wchar_t, char_traits<wchar_t> >::~~basic_ofstream()
vfunc[1]:	basic_ofstream<wchar_t, char_traits<wchar_t> >::~~basic_ofstream()

**Table 7-154 Secondary vtable for basic\_ofstream<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	-144
Virtual Base Offset	-144
RTTI	typeid for basic_ofstream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	virtual thunk to basic_ofstream<wchar_t, char_traits<wchar_t> >::~~basic_ofstream()
vfunc[1]:	virtual thunk to basic_ofstream<wchar_t, char_traits<wchar_t> >::~~basic_ofstream()

The VTT for the std::basic\_ofstream<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-155

**Table 7-155 VTT for basic\_ofstream<wchar\_t, char\_traits<wchar\_t> >**

VTT Name	_ZTTSt14basic_ofstreamIwSt11char_traitsIwEE
Number of Entries	4

### 7.1.81.2 Interfaces for Class `basic_ofstream<wchar_t, char_traits<wchar_t>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_ofstream<wchar_t, std::char_traits<wchar_t>>` specified in Table 7-156, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-156 libstdcxx - Class `basic_ofstream<wchar_t, char_traits<wchar_t>>` Function Interfaces**

virtual thunk to <code>basic_ofstream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_ofstream()</code> (GLIBCXX_3.4) [ISOCXX]
virtual thunk to <code>basic_ofstream&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;::~basic_ofstream()</code> (GLIBCXX_3.4) [ISOCXX]

### 7.1.82 Class `basic_streambuf<char, char_traits<char>>`

#### 7.1.82.1 Class data for `basic_streambuf<char, char_traits<char>>`

The virtual table for the `std::basic_streambuf<char, std::char_traits<char>>` class is described by Table 7-157

**Table 7-157 Primary vtable for `basic_streambuf<char, char_traits<char>>`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeinfo for basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;</code>
<code>vfunc[0]:</code>	<code>basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;::~basic_streambuf()</code>
<code>vfunc[1]:</code>	<code>basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;::~basic_streambuf()</code>
<code>vfunc[2]:</code>	<code>basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;::imbue(locale const&amp;)</code>
<code>vfunc[3]:</code>	<code>basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;::setbuf(char*, int)</code>
<code>vfunc[4]:</code>	<code>basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)</code>
<code>vfunc[5]:</code>	<code>basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;::seekpos(fpos&lt;__mbstate_t&gt;, _Ios_Openmode)</code>
<code>vfunc[6]:</code>	<code>basic_streambuf&lt;char, char_traits&lt;char&gt;&gt;::sync()</code>
<code>vfunc[7]:</code>	<code>basic_streambuf&lt;char,</code>



	char_traits<char> >::showmanyc()
vfunc[8]:	basic_streambuf<char, char_traits<char> >::xsgetn(char*, int)
vfunc[9]:	basic_streambuf<char, char_traits<char> >::underflow()
vfunc[10]:	basic_streambuf<char, char_traits<char> >::uflow()
vfunc[11]:	basic_streambuf<char, char_traits<char> >::pbackfail(int)
vfunc[12]:	basic_streambuf<char, char_traits<char> >::xsputn(char const*, int)
vfunc[13]:	basic_streambuf<char, char_traits<char> >::overflow(int)

The Run Time Type Information for the std::basic\_streambuf<char, std::char\_traits<char> > class is described by Table 7-158

**Table 7-158 typeinfo for basic\_streambuf<char, char\_traits<char> >**

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for basic_streambuf<char, char_traits<char> >

### 7.1.82.2 Interfaces for Class basic\_streambuf<char, char\_traits<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic\_streambuf<char, std::char\_traits<char> > specified in Table 7-159, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-159 libstdcxx - Class basic\_streambuf<char, char\_traits<char> > Function Interfaces**

basic_streambuf<char, char_traits<char> >::pubseekoff(long long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::sgetn(char*, int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::sputn(char const*, int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::setbuf(char*, int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::xsgetn(char*, int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::xsputn(char const*,

int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::pubsetbuf(char*, int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.83 Class basic\_streambuf<wchar\_t, char\_traits<wchar\_t> >

#### 7.1.83.1 Class data for basic\_streambuf<wchar\_t, char\_traits<wchar\_t> >

The virtual table for the std::basic\_streambuf<wchar\_t, std::char\_traits<wchar\_t> > class is described by Table 7-160

**Table 7-160 Primary vtable for basic\_streambuf<wchar\_t, char\_traits<wchar\_t> >**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for basic_streambuf<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::~~basic_streambuf()
vfunc[1]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::~~basic_streambuf()
vfunc[2]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::imbue(locale const&)
vfunc[3]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::setbuf(wchar_t*, int)
vfunc[4]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)
vfunc[5]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::sync()
vfunc[7]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::showmanyc()

vfunc[8]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::xsgetn(wchar_t*, int)
vfunc[9]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::underflow()
vfunc[10]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::uflow()
vfunc[11]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::pbackfail(unsigned int)
vfunc[12]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::xsputn(wchar_t const*, int)
vfunc[13]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::overflow(unsigned int)

The Run Time Type Information for the `std::basic_streambuf<wchar_t, std::char_traits<wchar_t> >` class is described by Table 7-161

**Table 7-161** typeid for `basic_streambuf<wchar_t, char_traits<wchar_t> >`

Base Vtable	vtable for <code>__cxxabiv1::__class_type_info</code>
Name	typeid name for <code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;</code>

### 7.1.83.2 Interfaces for Class `basic_streambuf<wchar_t, char_traits<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_streambuf<wchar_t, std::char_traits<wchar_t> >` specified in Table 7-162, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-162** `libstdcxx` - Class `basic_streambuf<wchar_t, char_traits<wchar_t> >` Function Interfaces

<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::pubseekoff(long long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::sgetn(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::sputn(wchar_t const*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::setbuf(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::xsgetn(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]</code>

<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::xsputn(wchar_t const*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::pubsetbuf(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]</code>

## 7.1.84 Class `basic_filebuf<char, char_traits<char> >`

### 7.1.84.1 Class data for `basic_filebuf<char, char_traits<char> >`

The virtual table for the `std::basic_filebuf<char, std::char_traits<char> >` class is described by Table 7-163

**Table 7-163 Primary vtable for `basic_filebuf<char, char_traits<char> >`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;</code>
<code>vfunc[0]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::~~basic_filebuf()</code>
<code>vfunc[1]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::~~basic_filebuf()</code>
<code>vfunc[2]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::imbue(locale const&amp;)</code>
<code>vfunc[3]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::setbuf(char*, int)</code>
<code>vfunc[4]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)</code>
<code>vfunc[5]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::seekpos(fpos&lt;_mbstate_t&gt;, _Ios_Openmode)</code>
<code>vfunc[6]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::sync()</code>
<code>vfunc[7]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::showmanyc()</code>
<code>vfunc[8]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::xsgetn(char*, int)</code>
<code>vfunc[9]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::underflow()</code>
<code>vfunc[10]:</code>	<code>basic_streambuf&lt;char, char_traits&lt;char&gt; &gt;::uflow()</code>
<code>vfunc[11]:</code>	<code>basic_filebuf&lt;char, char_traits&lt;char&gt; &gt;::pbackfail(int)</code>

vfunc[12]:	basic_filebuf<char, char_traits<char>>::xsputn(char const*, int)
vfunc[13]:	basic_filebuf<char, char_traits<char>>::overflow(int)

The Run Time Type Information for the `std::basic_filebuf<char, std::char_traits<char>>` class is described by Table 7-164

**Table 7-164 typeid for basic\_filebuf<char, char\_traits<char>>**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>basic_filebuf&lt;char, char_traits&lt;char&gt;&gt;</code>

### 7.1.84.2 Interfaces for Class `basic_filebuf<char, char_traits<char>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_filebuf<char, std::char_traits<char>>` specified in Table 7-165, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-165 libstdc++ - Class `basic_filebuf<char, char_traits<char>>` Function Interfaces**

<code>basic_filebuf&lt;char, char_traits&lt;char&gt;&gt;::M_set_buffer(int)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_filebuf&lt;char, char_traits&lt;char&gt;&gt;::M_convert_to_external(char*, int)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_filebuf&lt;char, char_traits&lt;char&gt;&gt;::setbuf(char*, int)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_filebuf&lt;char, char_traits&lt;char&gt;&gt;::xsgetn(char*, int)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_filebuf&lt;char, char_traits&lt;char&gt;&gt;::xsputn(char const*, int)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_filebuf&lt;char, char_traits&lt;char&gt;&gt;::M_seek(long long, _Ios_Seekdir, __mbstate_t)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_filebuf&lt;char, char_traits&lt;char&gt;&gt;::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4)</code> [ISOCXX]

### 7.1.85 Class `basic_filebuf<wchar_t, char_traits<wchar_t>>`

#### 7.1.85.1 Class data for `basic_filebuf<wchar_t, char_traits<wchar_t>>`

The virtual table for the `std::basic_filebuf<wchar_t, std::char_traits<wchar_t>>` class is described by Table 7-166

**Table 7-166 Primary vtable for `basic_filebuf<wchar_t, char_traits<wchar_t>>`**

Base Offset	0
-------------	---

Virtual Base Offset	0
RTTI	typeid for basic_filebuf<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::~~basic_filebuf()
vfunc[1]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::~~basic_filebuf()
vfunc[2]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::imbue(locale const&)
vfunc[3]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::setbuf(wchar_t*, int)
vfunc[4]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)
vfunc[5]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::sync()
vfunc[7]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::showmanyc()
vfunc[8]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::xsgetn(wchar_t*, int)
vfunc[9]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::underflow()
vfunc[10]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::uflow()
vfunc[11]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::pbackfail(unsigned int)
vfunc[12]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::xsputn(wchar_t const*, int)
vfunc[13]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::overflow(unsigned int)

The Run Time Type Information for the `std::basic_filebuf<wchar_t, std::char_traits<wchar_t> >` class is described by Table 7-167

**Table 7-167 typeid for `basic_filebuf<wchar_t, char_traits<wchar_t> >`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;</code>

### 7.1.85.2 Interfaces for Class `basic_filebuf<wchar_t, char_traits<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_filebuf<wchar_t, std::char_traits<wchar_t> >` specified in Table 7-168, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-168 libstdc++ - Class `basic_filebuf<wchar_t, char_traits<wchar_t> >` Function Interfaces**

<code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::_M_set_buffer(int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::_M_convert_to_external(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::setbuf(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::xsgetn(wchar_t*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::xsputn(wchar_t const*, int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::_M_seek(long long, _Ios_Seekdir, __mbstate_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_filebuf&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::seekoff(long long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_istream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::seekg(long long, _Ios_Seekdir)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::seekp(long long, _Ios_Seekdir)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_ostream&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;::_M_write(wchar_t const*, int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.86 Class `ios_base`

#### 7.1.86.1 Class data for `ios_base`

The Run Time Type Information for the `std::ios_base` class is described by Table 7-169

**Table 7-169 typeinfo for ios\_base**

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for ios_base

**7.1.86.2 Interfaces for Class ios\_base**

No external methods are defined for libstdc++ - Class std::ios\_base in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.87 Class basic\_ios<char, char\_traits<char> >****7.1.87.1 Class data for basic\_ios<char, char\_traits<char> >**

The virtual table for the std::basic\_ios<char, std::char\_traits<char> > class is described by Table 7-170

**Table 7-170 Primary vtable for basic\_ios<char, char\_traits<char> >**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for basic_ios<char, char_traits<char> >
vfunc[0]:	basic_ios<char, char_traits<char> >::~~basic_ios()
vfunc[1]:	basic_ios<char, char_traits<char> >::~~basic_ios()

**7.1.87.2 Interfaces for Class basic\_ios<char, char\_traits<char> >**

No external methods are defined for libstdc++ - Class std::basic\_ios<char, std::char\_traits<char> > in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.88 Class basic\_ios<wchar\_t, char\_traits<wchar\_t> >****7.1.88.1 Interfaces for Class basic\_ios<wchar\_t, char\_traits<wchar\_t> >**

No external methods are defined for libstdc++ - Class std::basic\_ios<wchar\_t, std::char\_traits<wchar\_t> > in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.89 Class ios\_base::failure****7.1.89.1 Class data for ios\_base::failure**

The virtual table for the std::ios\_base::failure class is described by Table 7-171

**Table 7-171 Primary vtable for ios\_base::failure**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for ios_base::failure



vfunc[0]:	ios_base::failure::~~failure()
vfunc[1]:	ios_base::failure::~~failure()
vfunc[2]:	ios_base::failure::what() const

The Run Time Type Information for the `std::ios_base::failure` class is described by Table 7-172

**Table 7-172 typeinfo for `ios_base::failure`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>ios_base::failure</code>

### 7.1.89.2 Interfaces for Class `ios_base::failure`

No external methods are defined for `libstdcxx` - Class `std::ios_base::failure` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

## 7.1.90 Class `__timepunct<char>`

### 7.1.90.1 Class data for `__timepunct<char>`

The virtual table for the `std::__timepunct<char>` class is described by Table 7-173

**Table 7-173 Primary vtable for `__timepunct<char>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>__timepunct&lt;char&gt;</code>
vfunc[0]:	<code>__timepunct&lt;char&gt;::~~__timepunct()</code>
vfunc[1]:	<code>__timepunct&lt;char&gt;::~~__timepunct()</code>

The Run Time Type Information for the `std::__timepunct<char>` class is described by Table 7-174

**Table 7-174 typeinfo for `__timepunct<char>`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>__timepunct&lt;char&gt;</code>

### 7.1.90.2 Interfaces for Class `__timepunct<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::__timepunct<char>` specified in Table 7-175, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-175 `libstdcxx` - Class `__timepunct<char>` Function Interfaces**

<code>__timepunct&lt;char&gt;::_M_put(char*, unsigned int, char const*, tm const*)</code>
---

<code>const(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;char&gt;::__timepunct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;char&gt;::__timepunct(__timepunct_cache&lt;char&gt;*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;char&gt;::__timepunct(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;char&gt;::__timepunct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;char&gt;::__timepunct(__timepunct_cache&lt;char&gt;*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;char&gt;::__timepunct(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.91 Class `__timepunct<wchar_t>`

#### 7.1.91.1 Class data for `__timepunct<wchar_t>`

The virtual table for the `std::__timepunct<wchar_t>` class is described by Table 7-176

**Table 7-176 Primary vtable for `__timepunct<wchar_t>`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeinfo for __timepunct&lt;wchar_t&gt;</code>
<code>vfunc[0]:</code>	<code>__timepunct&lt;wchar_t&gt;::~~__timepunct()</code>
<code>vfunc[1]:</code>	<code>__timepunct&lt;wchar_t&gt;::~~__timepunct()</code>

The Run Time Type Information for the `std::__timepunct<wchar_t>` class is described by Table 7-177

**Table 7-177 `typeinfo` for `__timepunct<wchar_t>`**

Base Vtable	<code>vtable for __cxxabiv1::__si_class_type_info</code>
Name	<code>typeinfo name for __timepunct&lt;wchar_t&gt;</code>

#### 7.1.91.2 Interfaces for Class `__timepunct<wchar_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::__timepunct<wchar_t>` specified in Table 7-178, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-178 `libstdcxx` - Class `__timepunct<wchar_t>` Function Interfaces**

<code>__timepunct&lt;wchar_t&gt;::__M_put(wchar_t*, unsigned int, wchar_t const*, tm const*) const(GLIBCXX_3.4) [ISOCXX]</code>
---

<code>__timepunct&lt;wchar_t&gt;::__timepunct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;wchar_t&gt;::__timepunct(__timepunct_cache&lt;wchar_t&gt;*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;wchar_t&gt;::__timepunct(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;wchar_t&gt;::__timepunct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;wchar_t&gt;::__timepunct(__timepunct_cache&lt;wchar_t&gt;*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>__timepunct&lt;wchar_t&gt;::__timepunct(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

## 7.1.92 Class messages\_base

### 7.1.92.1 Class data for messages\_base

The Run Time Type Information for the `std::messages_base` class is described by Table 7-179

**Table 7-179** typeinfo for messages\_base

Base Vtable	vtable for <code>__cxxabiv1::__class_type_info</code>
Name	typeinfo name for messages_base

### 7.1.92.2 Interfaces for Class messages\_base

No external methods are defined for `libstdc++` - Class `std::messages_base` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

## 7.1.93 Class messages<char>

### 7.1.93.1 Class data for messages<char>

The virtual table for the `std::messages<char>` class is described by Table 7-180

**Table 7-180** Primary vtable for messages<char>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for messages<char>
<code>vfunc[0]:</code>	<code>messages&lt;char&gt;::~~messages()</code>
<code>vfunc[1]:</code>	<code>messages&lt;char&gt;::~~messages()</code>
<code>vfunc[2]:</code>	<code>messages&lt;char&gt;::do_open(basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;, locale const&amp;) const</code>
<code>vfunc[3]:</code>	<code>messages&lt;char&gt;::do_get(int, int, int, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt; const&amp;) const</code>

vfunc[4]:	messages<char>::do_close(int) const
-----------	-------------------------------------

### 7.1.93.2 Interfaces for Class messages<char>

An LSB conforming implementation shall provide the architecture specific methods for Class std::messages<char> specified in Table 7-181, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-181 libstdcxx - Class messages<char> Function Interfaces**

messages<char>::messages(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
messages<char>::messages(unsigned int)(GLIBCXX_3.4) [ISOCXX]
messages<char>::messages(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
messages<char>::messages(unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.94 Class messages<wchar\_t>

#### 7.1.94.1 Class data for messages<wchar\_t>

The virtual table for the std::messages<wchar\_t> class is described by Table 7-182

**Table 7-182 Primary vtable for messages<wchar\_t>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for messages<wchar_t>
vfunc[0]:	messages<wchar_t>::~~messages()
vfunc[1]:	messages<wchar_t>::~~messages()
vfunc[2]:	messages<wchar_t>::do_open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const
vfunc[3]:	messages<wchar_t>::do_get(int, int, int, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const
vfunc[4]:	messages<wchar_t>::do_close(int) const

#### 7.1.94.2 Interfaces for Class messages<wchar\_t>

An LSB conforming implementation shall provide the architecture specific methods for Class std::messages<wchar\_t> specified in Table 7-183, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-183 libstdcxx - Class messages<wchar\_t> Function Interfaces**

messages<wchar_t>::messages(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::messages(unsigned int)(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::messages(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::messages(unsigned int)(GLIBCXX_3.4) [ISOCXX]

**7.1.95 Class messages\_byname<char>****7.1.95.1 Class data for messages\_byname<char>**

The virtual table for the std::messages\_byname<char> class is described by Table 7-184

**Table 7-184 Primary vtable for messages\_byname<char>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for messages_byname<char>
vfunc[0]:	messages_byname<char>::~~messages_byname()
vfunc[1]:	messages_byname<char>::~~messages_byname()
vfunc[2]:	messages<char>::do_open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const
vfunc[3]:	messages<char>::do_get(int, int, int, basic_string<char, char_traits<char>, allocator<char> > const&) const
vfunc[4]:	messages<char>::do_close(int) const

The Run Time Type Information for the std::messages\_byname<char> class is described by Table 7-185

**Table 7-185 typeid for messages\_byname<char>**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for messages_byname<char>

**7.1.95.2 Interfaces for Class messages\_byname<char>**

An LSB conforming implementation shall provide the architecture specific methods for Class std::messages\_byname<char> specified in Table 7-186, with

the full mandatory functionality as described in the referenced underlying specification.

**Table 7-186 libstdcxx - Class messages\_byname<char> Function Interfaces**

messages_byname<char>::messages_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
messages_byname<char>::messages_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]

## 7.1.96 Class messages\_byname<wchar\_t>

### 7.1.96.1 Class data for messages\_byname<wchar\_t>

The virtual table for the std::messages\_byname<wchar\_t> class is described by Table 7-187

**Table 7-187 Primary vtable for messages\_byname<wchar\_t>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for messages_byname<wchar_t>
vfunc[0]:	messages_byname<wchar_t>::~messages_byname()
vfunc[1]:	messages_byname<wchar_t>::~messages_byname()
vfunc[2]:	messages<wchar_t>::do_open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const
vfunc[3]:	messages<wchar_t>::do_get(int, int, int, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const
vfunc[4]:	messages<wchar_t>::do_close(int) const

The Run Time Type Information for the std::messages\_byname<wchar\_t> class is described by Table 7-188

**Table 7-188 typeid for messages\_byname<wchar\_t>**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for messages_byname<wchar_t>

### 7.1.96.2 Interfaces for Class `messages_byname<wchar_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::messages_byname<wchar_t>` specified in Table 7-189, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-189 libstdcxx - Class `messages_byname<wchar_t>` Function Interfaces**

<code>messages_byname&lt;wchar_t&gt;::messages_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>messages_byname&lt;wchar_t&gt;::messages_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.97 Class `numpunct<char>`

#### 7.1.97.1 Class data for `numpunct<char>`

The virtual table for the `std::numpunct<char>` class is described by Table 7-190

**Table 7-190 Primary vtable for `numpunct<char>`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeinfo for numpunct&lt;char&gt;</code>
<code>vfunc[0]:</code>	<code>numpunct&lt;char&gt;::~~numpunct()</code>
<code>vfunc[1]:</code>	<code>numpunct&lt;char&gt;::~~numpunct()</code>
<code>vfunc[2]:</code>	<code>numpunct&lt;char&gt;::do_decimal_point( ) const</code>
<code>vfunc[3]:</code>	<code>numpunct&lt;char&gt;::do_thousands_sep( ) const</code>
<code>vfunc[4]:</code>	<code>numpunct&lt;char&gt;::do_grouping() const</code>
<code>vfunc[5]:</code>	<code>numpunct&lt;char&gt;::do_truename() const</code>
<code>vfunc[6]:</code>	<code>numpunct&lt;char&gt;::do_falsename() const</code>

The Run Time Type Information for the `std::numpunct<char>` class is described by Table 7-191

**Table 7-191 `typeinfo` for `numpunct<char>`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	<code>typeinfo name for numpunct&lt;char&gt;</code>

### 7.1.97.2 Interfaces for Class `numpunct<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::numpunct<char>` specified in Table 7-192, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-192 libstdcxx - Class `numpunct<char>` Function Interfaces**

<code>numpunct&lt;char&gt;::numpunct(__locale_struct*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct&lt;char&gt;::numpunct(__numpunct_cache&lt;char&gt;*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct&lt;char&gt;::numpunct(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct&lt;char&gt;::numpunct(__locale_struct*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct&lt;char&gt;::numpunct(__numpunct_cache&lt;char&gt;*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct&lt;char&gt;::numpunct(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.98 Class `numpunct<wchar_t>`

#### 7.1.98.1 Class data for `numpunct<wchar_t>`

The virtual table for the `std::numpunct<wchar_t>` class is described by Table 7-193

**Table 7-193 Primary vtable for `numpunct<wchar_t>`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeid for numpunct&lt;wchar_t&gt;</code>
<code>vfunc[0]:</code>	<code>numpunct&lt;wchar_t&gt;::~~numpunct()</code>
<code>vfunc[1]:</code>	<code>numpunct&lt;wchar_t&gt;::~~numpunct()</code>
<code>vfunc[2]:</code>	<code>numpunct&lt;wchar_t&gt;::do_decimal_point() const</code>
<code>vfunc[3]:</code>	<code>numpunct&lt;wchar_t&gt;::do_thousands_sep() const</code>
<code>vfunc[4]:</code>	<code>numpunct&lt;wchar_t&gt;::do_grouping() const</code>
<code>vfunc[5]:</code>	<code>numpunct&lt;wchar_t&gt;::do_truename() const</code>
<code>vfunc[6]:</code>	<code>numpunct&lt;wchar_t&gt;::do_falsename() const</code>

The Run Time Type Information for the `std::numpunct<wchar_t>` class is described by Table 7-194



**Table 7-194 typeid for numput<wchar\_t>**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for numput<wchar_t>

**7.1.98.2 Interfaces for Class numput<wchar\_t>**

An LSB conforming implementation shall provide the architecture specific methods for Class std::numput<wchar\_t> specified in Table 7-195, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-195 libstdc++ - Class numput<wchar\_t> Function Interfaces**

numput<wchar_t>::numput(__locale_struct*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
numput<wchar_t>::numput(unsigned int)(GLIBCXX_3.4) [ISOCXX]
numput<wchar_t>::numput(__locale_struct*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
numput<wchar_t>::numput(unsigned int)(GLIBCXX_3.4) [ISOCXX]

**7.1.99 Class numput\_byname<char>****7.1.99.1 Class data for numput\_byname<char>**

The virtual table for the std::numput\_byname<char> class is described by Table 7-196

**Table 7-196 Primary vtable for numput\_byname<char>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for numput_byname<char>
vfunc[0]:	numput_byname<char>::~~numput_byname()
vfunc[1]:	numput_byname<char>::~~numput_byname()
vfunc[2]:	numput<char>::do_decimal_point() const
vfunc[3]:	numput<char>::do_thousands_sep() const
vfunc[4]:	numput<char>::do_grouping() const
vfunc[5]:	numput<char>::do_truename() const
vfunc[6]:	numput<char>::do_falsename()

	const
--	-------

The Run Time Type Information for the `std::num_punct_byname<char>` class is described by Table 7-197

**Table 7-197 typeid for `num_punct_byname<char>`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>num_punct_byname&lt;char&gt;</code>

### 7.1.99.2 Interfaces for Class `num_punct_byname<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::num_punct_byname<char>` specified in Table 7-198, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-198 libstdc++ - Class `num_punct_byname<char>` Function Interfaces**

<code>num_punct_byname&lt;char&gt;::num_punct_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>num_punct_byname&lt;char&gt;::num_punct_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.100 Class `num_punct_byname<wchar_t>`

#### 7.1.100.1 Class data for `num_punct_byname<wchar_t>`

The virtual table for the `std::num_punct_byname<wchar_t>` class is described by Table 7-199

**Table 7-199 Primary vtable for `num_punct_byname<wchar_t>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>num_punct_byname&lt;wchar_t&gt;</code>
<code>vfunc[0]:</code>	<code>num_punct_byname&lt;wchar_t&gt;::~~num_punct_byname()</code>
<code>vfunc[1]:</code>	<code>num_punct_byname&lt;wchar_t&gt;::~~num_punct_byname()</code>
<code>vfunc[2]:</code>	<code>num_punct&lt;wchar_t&gt;::do_decimal_point() const</code>
<code>vfunc[3]:</code>	<code>num_punct&lt;wchar_t&gt;::do_thousands_sep() const</code>
<code>vfunc[4]:</code>	<code>num_punct&lt;wchar_t&gt;::do_grouping() const</code>
<code>vfunc[5]:</code>	<code>num_punct&lt;wchar_t&gt;::do_truename()</code>

	const
vfunc[6]:	numpunct<wchar_t>::do_falsename( ) const

The Run Time Type Information for the `std::numpunct_byname<wchar_t>` class is described by Table 7-200

**Table 7-200 typeinfo for `numpunct_byname<wchar_t>`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>numpunct_byname&lt;wchar_t&gt;</code>

### 7.1.100.2 Interfaces for Class `numpunct_byname<wchar_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::numpunct_byname<wchar_t>` specified in Table 7-201, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-201 `libstdcxx` - Class `numpunct_byname<wchar_t>` Function Interfaces**

<code>numpunct_byname&lt;wchar_t&gt;::numpunct_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct_byname&lt;wchar_t&gt;::numpunct_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.101 Class `__codecvt_abstract_base<char, char, __mbstate_t>`

#### 7.1.101.1 Interfaces for Class `__codecvt_abstract_base<char, char, __mbstate_t>`

No external methods are defined for `libstdcxx` - Class `std::__codecvt_abstract_base<char, char, __mbstate_t>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.102 Class `__codecvt_abstract_base<wchar_t, char, __mbstate_t>`

#### 7.1.102.1 Class data for `__codecvt_abstract_base<wchar_t, char, __mbstate_t>`

The virtual table for the `std::__codecvt_abstract_base<wchar_t, char, __mbstate_t>` class is described by Table 7-202

**Table 7-202 Primary vtable for `__codecvt_abstract_base<wchar_t, char, __mbstate_t>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for

	<code>__codecvt_abstract_base&lt;wchar_t, char, __mbstate_t&gt;</code>
<code>vfunc[0]:</code>	
<code>vfunc[1]:</code>	
<code>vfunc[2]:</code>	<code>__cxa_pure_virtual</code>
<code>vfunc[3]:</code>	<code>__cxa_pure_virtual</code>
<code>vfunc[4]:</code>	<code>__cxa_pure_virtual</code>
<code>vfunc[5]:</code>	<code>__cxa_pure_virtual</code>
<code>vfunc[6]:</code>	<code>__cxa_pure_virtual</code>
<code>vfunc[7]:</code>	<code>__cxa_pure_virtual</code>
<code>vfunc[8]:</code>	<code>__cxa_pure_virtual</code>

### 7.1.102.2 Interfaces for Class `__codecvt_abstract_base<wchar_t, char, __mbstate_t>`

No external methods are defined for `libstdcxx` - Class `std::__codecvt_abstract_base<wchar_t, char, __mbstate_t>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.103 Class `codecvt_base`

#### 7.1.103.1 Class data for `codecvt_base`

The Run Time Type Information for the `std::codecvt_base` class is described by Table 7-203

Table 7-203 `typeinfo` for `codecvt_base`

Base Vtable	vtable for <code>__cxxabiv1::__class_type_info</code>
Name	<code>typeinfo</code> name for <code>codecvt_base</code>

#### 7.1.103.2 Interfaces for Class `codecvt_base`

No external methods are defined for `libstdcxx` - Class `std::codecvt_base` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.104 Class `codecvt<char, char, __mbstate_t>`

#### 7.1.104.1 Class data for `codecvt<char, char, __mbstate_t>`

The virtual table for the `std::codecvt<char, char, __mbstate_t>` class is described by Table 7-204

Table 7-204 Primary vtable for `codecvt<char, char, __mbstate_t>`

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeinfo</code> for <code>codecvt&lt;char, char,</code>

	<code>__mbstate_t&gt;</code>
<code>vfunc[0]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::~~codecvt()</code>
<code>vfunc[1]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::~~codecvt()</code>
<code>vfunc[2]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_out(__mbstate_t&amp;, char const*, char const*, char const*&amp;, char*, char*, char*&amp;) const</code>
<code>vfunc[3]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_unshift(__mbstate_t&amp;, char*, char*, char*&amp;) const</code>
<code>vfunc[4]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_in(__mbstate_t&amp;, char const*, char const*, char const*&amp;, char*, char*, char*&amp;) const</code>
<code>vfunc[5]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_encoding() const</code>
<code>vfunc[6]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_always_noconv() const</code>
<code>vfunc[7]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_length(__mbstate_t&amp;, char const*, char const*, unsigned int) const</code>
<code>vfunc[8]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_max_length() const</code>

The Run Time Type Information for the `std::codecvt<char, char, __mbstate_t>` class is described by Table 7-205

**Table 7-205** `typeinfo` for `codecvt<char, char, __mbstate_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>codecvt&lt;char, char, __mbstate_t&gt;</code>

#### 7.1.104.2 Class data for `__codecvt_abstract_base<char, char, __mbstate_t>`

The virtual table for the `std::__codecvt_abstract_base<char, char, __mbstate_t>` class is described by Table 7-206

**Table 7-206** Primary vtable for `__codecvt_abstract_base<char, char, __mbstate_t>`

Base Offset	0
Virtual Base Offset	0

RTTI	typeid for __codecvt_abstract_base<char, char, __mbstate_t>
vfunc[0]:	
vfunc[1]:	
vfunc[2]:	__cxa_pure_virtual
vfunc[3]:	__cxa_pure_virtual
vfunc[4]:	__cxa_pure_virtual
vfunc[5]:	__cxa_pure_virtual
vfunc[6]:	__cxa_pure_virtual
vfunc[7]:	__cxa_pure_virtual
vfunc[8]:	__cxa_pure_virtual

### 7.1.104.3 Interfaces for Class `codecvt<char, char, __mbstate_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::codecvt<char, char, __mbstate_t>` specified in Table 7-207, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-207 libstdcxx - Class `codecvt<char, char, __mbstate_t>` Function Interfaces**

<code>codecvt&lt;char, char, __mbstate_t&gt;::do_length(__mbstate_t&amp;, char const*, char const*, unsigned int) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>codecvt&lt;char, char, __mbstate_t&gt;::codecvt(__locale_struct*, unsigned int)</code> (GLIBCXX_3.4) [ISOCXX]
<code>codecvt&lt;char, char, __mbstate_t&gt;::codecvt(unsigned int)</code> (GLIBCXX_3.4) [ISOCXX]
<code>codecvt&lt;char, char, __mbstate_t&gt;::codecvt(__locale_struct*, unsigned int)</code> (GLIBCXX_3.4) [ISOCXX]
<code>codecvt&lt;char, char, __mbstate_t&gt;::codecvt(unsigned int)</code> (GLIBCXX_3.4) [ISOCXX]

### 7.1.105 Class `codecvt<wchar_t, char, __mbstate_t>`

#### 7.1.105.1 Class data for `codecvt<wchar_t, char, __mbstate_t>`

The virtual table for the `std::codecvt<wchar_t, char, __mbstate_t>` class is described by Table 7-208

**Table 7-208 Primary vtable for `codecvt<wchar_t, char, __mbstate_t>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>codecvt&lt;wchar_t, char, __mbstate_t&gt;</code>

vfunc[0]:	codecvt<wchar_t, char, __mbstate_t>::~~codecvt()
vfunc[1]:	codecvt<wchar_t, char, __mbstate_t>::~~codecvt()
vfunc[2]:	codecvt<wchar_t, char, __mbstate_t>::~do_out(__mbstate_t&, wchar_t const*, wchar_t const*, wchar_t const*&, char*, char*, char*&) const
vfunc[3]:	codecvt<wchar_t, char, __mbstate_t>::~do_unshift(__mbstate_t&, char*, char*, char*&) const
vfunc[4]:	codecvt<wchar_t, char, __mbstate_t>::~do_in(__mbstate_t&, char const*, char const*, char const*&, wchar_t*, wchar_t*, wchar_t*&) const
vfunc[5]:	codecvt<wchar_t, char, __mbstate_t>::~do_encoding() const
vfunc[6]:	codecvt<wchar_t, char, __mbstate_t>::~do_always_noconv() const
vfunc[7]:	codecvt<wchar_t, char, __mbstate_t>::~do_length(__mbstate_t&, char const*, char const*, unsigned int) const
vfunc[8]:	codecvt<wchar_t, char, __mbstate_t>::~do_max_length() const

The Run Time Type Information for the `std::codecvt<wchar_t, char, __mbstate_t>` class is described by Table 7-209

**Table 7-209 typeid for `codecvt<wchar_t, char, __mbstate_t>`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>codecvt&lt;wchar_t, char, __mbstate_t&gt;</code>

### 7.1.105.2 Interfaces for Class `codecvt<wchar_t, char, __mbstate_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::codecvt<wchar_t, char, __mbstate_t>` specified in Table 7-210, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-210 libstdc++ - Class `codecvt<wchar_t, char, __mbstate_t>` Function Interfaces**

<code>codecvt&lt;wchar_t, char, __mbstate_t&gt;::~do_length(__mbstate_t&amp;, char const*,</code>
---

<code>char const*, unsigned int) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>codecvt&lt;wchar_t, char, __mbstate_t&gt;::codecvt(__locale_struct*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>codecvt&lt;wchar_t, char, __mbstate_t&gt;::codecvt(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>codecvt&lt;wchar_t, char, __mbstate_t&gt;::codecvt(__locale_struct*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>codecvt&lt;wchar_t, char, __mbstate_t&gt;::codecvt(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.106 Class `codecvt_byname<char, char, __mbstate_t>`

#### 7.1.106.1 Class data for `codecvt_byname<char, char, __mbstate_t>`

The virtual table for the `std::codecvt_byname<char, char, __mbstate_t>` class is described by Table 7-211

**Table 7-211 Primary vtable for `codecvt_byname<char, char, __mbstate_t>`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeinfo for codecvt_byname&lt;char, char, __mbstate_t&gt;</code>
<code>vfunc[0]:</code>	<code>codecvt_byname&lt;char, char, __mbstate_t&gt;::~~codecvt_byname()</code>
<code>vfunc[1]:</code>	<code>codecvt_byname&lt;char, char, __mbstate_t&gt;::~~codecvt_byname()</code>
<code>vfunc[2]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_out(__mbstate_t&amp;, char const*, char const*, char const*&amp;, char*, char*, char*&amp;) const</code>
<code>vfunc[3]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_unshift(__mbstate_t&amp;, char*, char*, char*&amp;) const</code>
<code>vfunc[4]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_in(__mbstate_t&amp;, char const*, char const*, char const*&amp;, char*, char*, char*&amp;) const</code>
<code>vfunc[5]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_encoding() const</code>
<code>vfunc[6]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_always_noconv() const</code>
<code>vfunc[7]:</code>	<code>codecvt&lt;char, char, __mbstate_t&gt;::do_length(__mbstate_t&amp;, char const*, char const*, unsigned int) const</code>



vfunc[8]:	codecvt<char, char, __mbstate_t>::do_max_length() const
-----------	--

The Run Time Type Information for the `std::codecvt_byname<char, char, __mbstate_t>` class is described by Table 7-212

**Table 7-212 typeid for `codecvt_byname<char, char, __mbstate_t>`**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for <code>codecvt_byname&lt;char, char, __mbstate_t&gt;</code>

### 7.1.106.2 Interfaces for Class `codecvt_byname<char, char, __mbstate_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::codecvt_byname<char, char, __mbstate_t>` specified in Table 7-213, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-213 libstdc++ - Class `codecvt_byname<char, char, __mbstate_t>` Function Interfaces**

<code>codecvt_byname&lt;char, char, __mbstate_t&gt;::codecvt_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>codecvt_byname&lt;char, char, __mbstate_t&gt;::codecvt_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.107 Class `codecvt_byname<wchar_t, char, __mbstate_t>`

#### 7.1.107.1 Class data for `codecvt_byname<wchar_t, char, __mbstate_t>`

The virtual table for the `std::codecvt_byname<wchar_t, char, __mbstate_t>` class is described by Table 7-214

**Table 7-214 Primary vtable for `codecvt_byname<wchar_t, char, __mbstate_t>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>codecvt_byname&lt;wchar_t, char, __mbstate_t&gt;</code>
vfunc[0]:	<code>codecvt_byname&lt;wchar_t, char, __mbstate_t&gt;::~~codecvt_byname()</code>
vfunc[1]:	<code>codecvt_byname&lt;wchar_t, char, __mbstate_t&gt;::~~codecvt_byname()</code>
vfunc[2]:	<code>codecvt&lt;wchar_t, char, __mbstate_t&gt;::do_out(__mbstate_t&amp;, wchar_t const*, wchar_t const*,</code>

	wchar_t const*&, char*, char*, char*&) const
vfunc[3]:	codecvt<wchar_t, char, __mbstate_t>::do_unshift(__mbstate_t&, char*, char*, char*&) const
vfunc[4]:	codecvt<wchar_t, char, __mbstate_t>::do_in(__mbstate_t&, char const*, char const*, char const*&, wchar_t*, wchar_t*, wchar_t*&) const
vfunc[5]:	codecvt<wchar_t, char, __mbstate_t>::do_encoding() const
vfunc[6]:	codecvt<wchar_t, char, __mbstate_t>::do_always_noconv() const
vfunc[7]:	codecvt<wchar_t, char, __mbstate_t>::do_length(__mbstate_t&, char const*, char const*, unsigned int) const
vfunc[8]:	codecvt<wchar_t, char, __mbstate_t>::do_max_length() const

The Run Time Type Information for the `std::codecvt_byname<wchar_t, char, __mbstate_t>` class is described by Table 7-215

**Table 7-215** typeid for `codecvt_byname<wchar_t, char, __mbstate_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>codecvt_byname&lt;wchar_t, char, __mbstate_t&gt;</code>

### 7.1.107.2 Class data for `collate_byname<wchar_t>`

The virtual table for the `std::collate_byname<wchar_t>` class is described by Table 7-216

**Table 7-216** Primary vtable for `collate_byname<wchar_t>`

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>collate_byname&lt;wchar_t&gt;</code>
vfunc[0]:	<code>collate_byname&lt;wchar_t&gt;::~~collate_byname()</code>
vfunc[1]:	<code>collate_byname&lt;wchar_t&gt;::~~collate_byname()</code>

vfunc[2]:	collate<wchar_t>::do_compare(wchar_t const*, wchar_t const*, wchar_t const*, wchar_t const*) const
vfunc[3]:	collate<wchar_t>::do_transform(wchar_t const*, wchar_t const*) const
vfunc[4]:	collate<wchar_t>::do_hash(wchar_t const*, wchar_t const*) const

The Run Time Type Information for the `std::collate_byname<wchar_t>` class is described by Table 7-217

**Table 7-217 typeinfo for collate\_byname<wchar\_t>**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>collate_byname&lt;wchar_t&gt;</code>

### 7.1.107.3 Interfaces for Class `codecvt_byname<wchar_t, char, __mbstate_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::codecvt_byname<wchar_t, char, __mbstate_t>` specified in Table 7-218, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-218 libstdcxx - Class `codecvt_byname<wchar_t, char, __mbstate_t>` Function Interfaces**

<code>codecvt_byname&lt;wchar_t, char, __mbstate_t&gt;::codecvt_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>codecvt_byname&lt;wchar_t, char, __mbstate_t&gt;::codecvt_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>collate_byname&lt;wchar_t&gt;::collate_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>collate_byname&lt;wchar_t&gt;::collate_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.108 Class `collate<char>`

#### 7.1.108.1 Class data for `collate<char>`

The virtual table for the `std::collate<char>` class is described by Table 7-219

**Table 7-219 Primary vtable for collate<char>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>collate&lt;char&gt;</code>
vfunc[0]:	<code>collate&lt;char&gt;::~~collate()</code>

vfunc[1]:	collate<char>::~~collate()
vfunc[2]:	collate<char>::~do_compare(char const*, char const*, char const*, char const*) const
vfunc[3]:	collate<char>::~do_transform(char const*, char const*) const
vfunc[4]:	collate<char>::~do_hash(char const*, char const*) const

The Run Time Type Information for the std::collate<char> class is described by Table 7-220

**Table 7-220 typeid for collate<char>**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for collate<char>

### 7.1.108.2 Interfaces for Class collate<char>

An LSB conforming implementation shall provide the architecture specific methods for Class std::collate<char> specified in Table 7-221, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-221 libstdc++ - Class collate<char> Function Interfaces**

collate<char>::_M_transform(char*, char const*, unsigned int) const(GLIBCXX_3.4) [ISOCXX]
collate<char>::collate(__locale_struct*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
collate<char>::collate(unsigned int)(GLIBCXX_3.4) [ISOCXX]
collate<char>::collate(__locale_struct*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
collate<char>::collate(unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.109 Class collate<wchar\_t>

#### 7.1.109.1 Class data for collate<wchar\_t>

The virtual table for the std::collate<wchar\_t> class is described by Table 7-222

**Table 7-222 Primary vtable for collate<wchar\_t>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for collate<wchar_t>
vfunc[0]:	collate<wchar_t>::~~collate()
vfunc[1]:	collate<wchar_t>::~~collate()
vfunc[2]:	collate<wchar_t>::~do_compare(wcha

	r_t const*, wchar_t const*, wchar_t const*, wchar_t const*) const
vfunc[3]:	collate<wchar_t>::do_transform(wchar_t const*, wchar_t const*) const
vfunc[4]:	collate<wchar_t>::do_hash(wchar_t const*, wchar_t const*) const

The Run Time Type Information for the std::collate<wchar\_t> class is described by Table 7-223

**Table 7-223 typeid for collate<wchar\_t>**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for collate<wchar_t>

### 7.1.109.2 Interfaces for Class collate<wchar\_t>

An LSB conforming implementation shall provide the architecture specific methods for Class std::collate<wchar\_t> specified in Table 7-224, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-224 libstdc++ - Class collate<wchar\_t> Function Interfaces**

collate<wchar_t>::_M_transform(wchar_t*, wchar_t const*, unsigned int) const (GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::collate(__locale_struct*, unsigned int) (GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::collate(unsigned int) (GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::collate(__locale_struct*, unsigned int) (GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::collate(unsigned int) (GLIBCXX_3.4) [ISOCXX]

### 7.1.110 Class collate\_byname<char>

#### 7.1.110.1 Class data for collate\_byname<char>

The virtual table for the std::collate\_byname<char> class is described by Table 7-225

**Table 7-225 Primary vtable for collate\_byname<char>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for collate_byname<char>
vfunc[0]:	collate_byname<char>::~~collate_byname()
vfunc[1]:	collate_byname<char>::~~collate_byname()

vfunc[2]:	collate<char>::do_compare(char const*, char const*, char const*, char const*) const
vfunc[3]:	collate<char>::do_transform(char const*, char const*) const
vfunc[4]:	collate<char>::do_hash(char const*, char const*) const

The Run Time Type Information for the `std::collate_byname<char>` class is described by Table 7-226

**Table 7-226 typeinfo for `collate_byname<char>`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>collate_byname&lt;char&gt;</code>

### 7.1.110.2 Interfaces for Class `collate_byname<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::collate_byname<char>` specified in Table 7-227, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-227 libstdcxx - Class `collate_byname<char>` Function Interfaces**

<code>collate_byname&lt;char&gt;::collate_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>collate_byname&lt;char&gt;::collate_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.111 Class `collate_byname<wchar_t>`

#### 7.1.111.1 Interfaces for Class `collate_byname<wchar_t>`

No external methods are defined for `libstdcxx` - Class `std::collate_byname<wchar_t>` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.112 Class `time_base`

#### 7.1.112.1 Class data for `time_base`

The Run Time Type Information for the `std::time_base` class is described by Table 7-228

**Table 7-228 typeinfo for `time_base`**

Base Vtable	vtable for <code>__cxxabiv1::__class_type_info</code>
Name	typeinfo name for <code>time_base</code>

### 7.1.112.2 Interfaces for Class `time_base`

No external methods are defined for `libstdc++` - Class `std::time_base` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.113 Class `time_get_byname<char, istreambuf_iterator<char, char_traits<char>>>`

#### 7.1.113.1 Class data for `time_get_byname<char, istreambuf_iterator<char, char_traits<char>>>`

The virtual table for the `std::time_get_byname<char, std::istreambuf_iterator<char, std::char_traits<char>>>` class is described by Table 7-229

Table 7-229 Primary vtable for `time_get_byname<char, istreambuf_iterator<char, char_traits<char>>>`

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeid</code> for <code>time_get_byname&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code>
<code>vfunc[0]:</code>	<code>time_get_byname&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::~time_get_byname()</code>
<code>vfunc[1]:</code>	<code>time_get_byname&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::~time_get_byname()</code>
<code>vfunc[2]:</code>	<code>time_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::do_date_order() const</code>
<code>vfunc[3]:</code>	<code>time_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::do_get_time(istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</code>
<code>vfunc[4]:</code>	<code>time_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::do_get_date(istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</code>

vfunc[5]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_weekday(istreambuf_iterat or<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[6]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_monthname(istreambuf_it erator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[7]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_year(istreambuf_iterator<c har, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const

The Run Time Type Information for the `std::time_get_byname<char, std::istreambuf_iterator<char, std::char_traits<char> > >` class is described by Table 7-230

**Table 7-230 typeid for `time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>time_get_byname&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;</code>

### 7.1.113.2 Interfaces for Class `time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_get_byname<char, std::istreambuf_iterator<char, std::char_traits<char> > >` specified in Table 7-231, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-231 libstdc++ - Class `time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >` Function Interfaces**

<code>time_get_byname&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;::time_get_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_get_byname&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;</code>



<code>&gt;::time_get_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
---

### 7.1.114 Class `time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

#### 7.1.114.1 Class data for `time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

The virtual table for the `std::time_get_byname<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t>>>` class is described by Table 7-232

**Table 7-232 Primary vtable for `time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>time_get_byname&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code>
vfunc[0]:	<code>time_get_byname&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::~time_get_byname()</code>
vfunc[1]:	<code>time_get_byname&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::~time_get_byname()</code>
vfunc[2]:	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::do_date_order() const</code>
vfunc[3]:	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::do_get_time(istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</code>
vfunc[4]:	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::do_get_date(istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</code>
vfunc[5]:	<code>time_get&lt;wchar_t,</code>

	<pre>istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::do_get_weekday(istreambuf_iterat or&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</pre>
<pre>vfunc[6]:</pre>	<pre>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::do_get_monthname(istreambuf_it erator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</pre>
<pre>vfunc[7]:</pre>	<pre>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::do_get_year(istreambuf_iterator&lt; wchar_t, char_traits&lt;wchar_t&gt; &gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</pre>

The Run Time Type Information for the `std::time_get_byname<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > >` class is described by Table 7-233

**Table 7-233** `typeinfo` for `time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>time_get_byname&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;</code>

#### 7.1.114.2 Interfaces for Class `time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_get_byname<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > >` specified in Table 7-234, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-234** `libstdcxx` - Class `time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >` Function Interfaces

<pre>time_get_byname&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::time_get_byname(char const*, unsigned</pre>
--

int)(GLIBCXX_3.4) [ISOCXX]
time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::time_get_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.115 Class `time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >`

#### 7.1.115.1 Class data for `time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >`

The virtual table for the `std::time_put_byname<char, std::ostreambuf_iterator<char, std::char_traits<char> > >` class is described by Table 7-235

**Table 7-235 Primary vtable for `time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeinfo</code> for <code>time_put_byname&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;</code>
<code>vfunc[0]:</code>	<code>time_put_byname&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;::~time_put_byname()</code>
<code>vfunc[1]:</code>	<code>time_put_byname&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;::~time_put_byname()</code>
<code>vfunc[2]:</code>	<code>time_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;::do_put(ostreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;, ios_base&amp;, char, tm const*, char, char) const</code>

The Run Time Type Information for the `std::time_put_byname<char, std::ostreambuf_iterator<char, std::char_traits<char> > >` class is described by Table 7-236

**Table 7-236 `typeinfo` for `time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	<code>typeinfo</code> name for <code>time_put_byname&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;</code>

### 7.1.115.2 Interfaces for Class `time_put_byname<char, ostreambuf_iterator<char, char_traits<char>>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_put_byname<char, std::ostreambuf_iterator<char, std::char_traits<char>>>` specified in Table 7-237, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-237 libstdcxx - Class `time_put_byname<char, ostreambuf_iterator<char, char_traits<char>>>` Function Interfaces**

<code>time_put_byname&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::time_put_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_put_byname&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::time_put_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.116 Class `time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

#### 7.1.116.1 Class data for `time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

The virtual table for the `std::time_put_byname<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t>>>` class is described by Table 7-238

**Table 7-238 Primary vtable for `time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>time_put_byname&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code>
vfunc[0]:	<code>time_put_byname&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::~time_put_byname()</code>
vfunc[1]:	<code>time_put_byname&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::~time_put_byname()</code>
vfunc[2]:	<code>time_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::do_put(ostreambuf_iterator&lt;wcha</code> <code>r_t, char_traits&lt;wchar_t&gt;&gt;,</code> <code>ios_base&amp;, wchar_t, tm const*, char,</code> <code>char) const</code>

The Run Time Type Information for the `std::time_put_byname<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > >` class is described by Table 7-239

**Table 7-239** typeinfo for `time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>time_put_byname&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;</code>

### 7.1.116.2 Interfaces for Class `time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_put_byname<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > >` specified in Table 7-240, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-240** `libstdcxx` - Class `time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >` Function Interfaces

<code>time_put_byname&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::time_put_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_put_byname&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::time_put_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.117 Class `time_get<char, istreambuf_iterator<char, char_traits<char> > >`

#### 7.1.117.1 Class data for `time_get<char, istreambuf_iterator<char, char_traits<char> > >`

The virtual table for the `std::time_get<char, std::istreambuf_iterator<char, std::char_traits<char> > >` class is described by Table 7-241

**Table 7-241** Primary vtable for `time_get<char, istreambuf_iterator<char, char_traits<char> > >`

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>time_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;</code>
<code>vfunc[0]:</code>	<code>time_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;::~time_get()</code>

vfunc[1]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::~time_get()
vfunc[2]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_date_order() const
vfunc[3]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_time(istreambuf_iterator<c har, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[4]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_date(istreambuf_iterator<c har, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[5]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_weekday(istreambuf_iterat or<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[6]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_monthname(istreambuf_it erator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[7]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_year(istreambuf_iterator<c har, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const

### 7.1.117.2 Interfaces for Class `time_get<char, istreambuf_iterator<char, char_traits<char>>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_get<char, std::istreambuf_iterator<char, std::char_traits<char>>>` specified in Table 7-242, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-242 libstdc++ - Class `time_get<char, istreambuf_iterator<char, char_traits<char>>>` Function Interfaces**

<code>time_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::M_extract_num(istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;, int&amp;, int, int, unsigned int, ios_base&amp;, _Ios_Iostate&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::M_extract_name(istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;, int&amp;, char const**, unsigned int, ios_base&amp;, _Ios_Iostate&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::time_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::time_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.118 Class `time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

#### 7.1.118.1 Class data for `time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

The virtual table for the `std::time_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t>>>` class is described by Table 7-243

**Table 7-243 Primary vtable for `time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeinfo for time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code>
<code>vfunc[0]:</code>	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::~~time_get()</code>
<code>vfunc[1]:</code>	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::~~time_get()</code>
<code>vfunc[2]:</code>	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t,</code>

	<code>char_traits&lt;wchar_t&gt;&gt; &gt;::do_date_order() const</code>
<code>vfunc[3]:</code>	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt; &gt;::do_get_time(istreambuf_iterator&lt; wchar_t, char_traits&lt;wchar_t&gt;&gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</code>
<code>vfunc[4]:</code>	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt; &gt;::do_get_date(istreambuf_iterator&lt; wchar_t, char_traits&lt;wchar_t&gt;&gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</code>
<code>vfunc[5]:</code>	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt; &gt;::do_get_weekday(istreambuf_iterat or&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</code>
<code>vfunc[6]:</code>	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt; &gt;::do_get_monthname(istreambuf_it erator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</code>
<code>vfunc[7]:</code>	<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt; &gt;::do_get_year(istreambuf_iterator&lt; wchar_t, char_traits&lt;wchar_t&gt;&gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, ios_base&amp;, _Ios_Iostate&amp;, tm*) const</code>

#### 7.1.118.2 Interfaces for Class `time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t>>>` specified in Table 7-244, with the full mandatory functionality as described in the referenced underlying specification.



**Table 7-244 libstdcxx - Class `time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>` Function Interfaces**

<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;::M_extract_num(istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, int&amp;, int, unsigned int, ios_base&amp;, _Ios_Iostate&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;::M_extract_name(istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;, int&amp;, wchar_t const**, unsigned int, ios_base&amp;, _Ios_Iostate&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;::time_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;::time_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

**7.1.119 Class `time_put<char, ostreambuf_iterator<char, char_traits<char>>>`****7.1.119.1 Interfaces for Class `time_put<char, ostreambuf_iterator<char, char_traits<char>>>`**

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_put<char, std::ostreambuf_iterator<char, std::char_traits<char>>>` specified in Table 7-245, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-245 libstdcxx - Class `time_put<char, ostreambuf_iterator<char, char_traits<char>>>` Function Interfaces**

<code>time_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;::time_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;::time_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

**7.1.120 Class `time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>`****7.1.120.1 Interfaces for Class `time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>`**

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t>>>` specified in Table 7-246, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-246 libstdcxx - Class `time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>` Function Interfaces**

<code>time_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;::time_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;::time_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

**7.1.121 Class moneypunct<char, false>****7.1.121.1 Class data for moneypunct<char, false>**

The virtual table for the `std::moneypunct<char, false>` class is described by Table 7-247

**Table 7-247 Primary vtable for moneypunct<char, false>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for moneypunct<char, false>
vfunc[0]:	moneypunct<char, false>::~~moneypunct()
vfunc[1]:	moneypunct<char, false>::~~moneypunct()
vfunc[2]:	moneypunct<char, false>::do_decimal_point() const
vfunc[3]:	moneypunct<char, false>::do_thousands_sep() const
vfunc[4]:	moneypunct<char, false>::do_grouping() const
vfunc[5]:	moneypunct<char, false>::do_curr_symbol() const
vfunc[6]:	moneypunct<char, false>::do_positive_sign() const
vfunc[7]:	moneypunct<char, false>::do_negative_sign() const
vfunc[8]:	moneypunct<char, false>::do_frac_digits() const
vfunc[9]:	moneypunct<char, false>::do_pos_format() const
vfunc[10]:	moneypunct<char, false>::do_neg_format() const

**7.1.121.2 Interfaces for Class moneypunct<char, false>**

An LSB conforming implementation shall provide the architecture specific methods for Class `std::moneypunct<char, false>` specified in Table 7-248, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-248 libstdcxx - Class moneypunct<char, false> Function Interfaces**

<code>moneypunct&lt;char, false&gt;::moneypunct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>moneypunct&lt;char, false&gt;::moneypunct(__moneypunct_cache&lt;char, false&gt;*,</code>

unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_punct<char, false>::money_punct(unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_punct<char, false>::money_punct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_punct<char, false>::money_punct(__money_punct_cache<char, false>*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_punct<char, false>::money_punct(unsigned int)(GLIBCXX_3.4) [ISOCXX]

## 7.1.122 Class money\_punct<char, true>

### 7.1.122.1 Class data for money\_punct<char, true>

The virtual table for the std::money\_punct<char, true> class is described by Table 7-249

**Table 7-249 Primary vtable for money\_punct<char, true>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for money_punct<char, true>
vfunc[0]:	money_punct<char, true>::~~money_punct()
vfunc[1]:	money_punct<char, true>::~~money_punct()
vfunc[2]:	money_punct<char, true>::do_decimal_point() const
vfunc[3]:	money_punct<char, true>::do_thousands_sep() const
vfunc[4]:	money_punct<char, true>::do_grouping() const
vfunc[5]:	money_punct<char, true>::do_curr_symbol() const
vfunc[6]:	money_punct<char, true>::do_positive_sign() const
vfunc[7]:	money_punct<char, true>::do_negative_sign() const
vfunc[8]:	money_punct<char, true>::do_frac_digits() const
vfunc[9]:	money_punct<char, true>::do_pos_format() const
vfunc[10]:	money_punct<char, true>::do_neg_format() const

### 7.1.122.2 Interfaces for Class `money_punct<char, true>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::money_punct<char, true>` specified in Table 7-250, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-250 libstdcxx - Class `money_punct<char, true>` Function Interfaces**

<code>money_punct&lt;char, true&gt;::money_punct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>money_punct&lt;char, true&gt;::money_punct(__money_punct_cache&lt;char, true&gt;*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>money_punct&lt;char, true&gt;::money_punct(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>money_punct&lt;char, true&gt;::money_punct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>money_punct&lt;char, true&gt;::money_punct(__money_punct_cache&lt;char, true&gt;*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>money_punct&lt;char, true&gt;::money_punct(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.123 Class `money_punct<wchar_t, false>`

#### 7.1.123.1 Class data for `money_punct<wchar_t, false>`

The virtual table for the `std::money_punct<wchar_t, false>` class is described by Table 7-251

**Table 7-251 Primary vtable for `money_punct<wchar_t, false>`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeid for money_punct&lt;wchar_t, false&gt;</code>
<code>vfunc[0]:</code>	<code>money_punct&lt;wchar_t, false&gt;::~~money_punct()</code>
<code>vfunc[1]:</code>	<code>money_punct&lt;wchar_t, false&gt;::~~money_punct()</code>
<code>vfunc[2]:</code>	<code>money_punct&lt;wchar_t, false&gt;::do_decimal_point() const</code>
<code>vfunc[3]:</code>	<code>money_punct&lt;wchar_t, false&gt;::do_thousands_sep() const</code>
<code>vfunc[4]:</code>	<code>money_punct&lt;wchar_t, false&gt;::do_grouping() const</code>
<code>vfunc[5]:</code>	<code>money_punct&lt;wchar_t, false&gt;::do_curr_symbol() const</code>
<code>vfunc[6]:</code>	<code>money_punct&lt;wchar_t,</code>

	false>::do_positive_sign() const
vfunc[7]:	money_punct<wchar_t, false>::do_negative_sign() const
vfunc[8]:	money_punct<wchar_t, false>::do_frac_digits() const
vfunc[9]:	money_punct<wchar_t, false>::do_pos_format() const
vfunc[10]:	money_punct<wchar_t, false>::do_neg_format() const

### 7.1.123.2 Interfaces for Class money\_punct<wchar\_t, false>

An LSB conforming implementation shall provide the architecture specific methods for Class std::money\_punct<wchar\_t, false> specified in Table 7-252, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-252 libstdc++ - Class money\_punct<wchar\_t, false> Function Interfaces**

money_punct<wchar_t, false>::money_punct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_punct<wchar_t, false>::money_punct(__money_punct_cache<wchar_t, false>*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_punct<wchar_t, false>::money_punct(unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_punct<wchar_t, false>::money_punct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_punct<wchar_t, false>::money_punct(__money_punct_cache<wchar_t, false>*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_punct<wchar_t, false>::money_punct(unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.124 Class money\_punct<wchar\_t, true>

#### 7.1.124.1 Class data for money\_punct<wchar\_t, true>

The virtual table for the std::money\_punct<wchar\_t, true> class is described by Table 7-253

**Table 7-253 Primary vtable for money\_punct<wchar\_t, true>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for money_punct<wchar_t, true>
vfunc[0]:	money_punct<wchar_t, true>::~~money_punct()
vfunc[1]:	money_punct<wchar_t,

	true>::~moneypunct()
vfunc[2]:	moneypunct<wchar_t, true>::do_decimal_point() const
vfunc[3]:	moneypunct<wchar_t, true>::do_thousands_sep() const
vfunc[4]:	moneypunct<wchar_t, true>::do_grouping() const
vfunc[5]:	moneypunct<wchar_t, true>::do_curr_symbol() const
vfunc[6]:	moneypunct<wchar_t, true>::do_positive_sign() const
vfunc[7]:	moneypunct<wchar_t, true>::do_negative_sign() const
vfunc[8]:	moneypunct<wchar_t, true>::do_frac_digits() const
vfunc[9]:	moneypunct<wchar_t, true>::do_pos_format() const
vfunc[10]:	moneypunct<wchar_t, true>::do_neg_format() const

### 7.1.124.2 Interfaces for Class `moneypunct<wchar_t, true>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::moneypunct<wchar_t, true>` specified in Table 7-254, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-254 libstdcxx - Class `moneypunct<wchar_t, true>` Function Interfaces**

<code>moneypunct&lt;wchar_t, true&gt;::moneypunct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>moneypunct&lt;wchar_t, true&gt;::moneypunct(__moneypunct_cache&lt;wchar_t, true&gt;*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>moneypunct&lt;wchar_t, true&gt;::moneypunct(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>moneypunct&lt;wchar_t, true&gt;::moneypunct(__locale_struct*, char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>moneypunct&lt;wchar_t, true&gt;::moneypunct(__moneypunct_cache&lt;wchar_t, true&gt;*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>moneypunct&lt;wchar_t, true&gt;::moneypunct(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.125 Class `moneypunct_byname<char, false>`

#### 7.1.125.1 Class data for `moneypunct_byname<char, false>`

The virtual table for the `std::moneypunct_byname<char, false>` class is described by Table 7-255

**Table 7-255 Primary vtable for `money_punct_byname<char, false>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>money_punct_byname&lt;char, false&gt;</code>
<code>vfunc[0]:</code>	<code>money_punct_byname&lt;char, false&gt;::~~money_punct_byname()</code>
<code>vfunc[1]:</code>	<code>money_punct_byname&lt;char, false&gt;::~~money_punct_byname()</code>
<code>vfunc[2]:</code>	<code>money_punct&lt;char, false&gt;::do_decimal_point() const</code>
<code>vfunc[3]:</code>	<code>money_punct&lt;char, false&gt;::do_thousands_sep() const</code>
<code>vfunc[4]:</code>	<code>money_punct&lt;char, false&gt;::do_grouping() const</code>
<code>vfunc[5]:</code>	<code>money_punct&lt;char, false&gt;::do_curr_symbol() const</code>
<code>vfunc[6]:</code>	<code>money_punct&lt;char, false&gt;::do_positive_sign() const</code>
<code>vfunc[7]:</code>	<code>money_punct&lt;char, false&gt;::do_negative_sign() const</code>
<code>vfunc[8]:</code>	<code>money_punct&lt;char, false&gt;::do_frac_digits() const</code>
<code>vfunc[9]:</code>	<code>money_punct&lt;char, false&gt;::do_pos_format() const</code>
<code>vfunc[10]:</code>	<code>money_punct&lt;char, false&gt;::do_neg_format() const</code>

The Run Time Type Information for the `std::money_punct_byname<char, false>` class is described by Table 7-256

**Table 7-256 typeid for `money_punct_byname<char, false>`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>money_punct_byname&lt;char, false&gt;</code>

### 7.1.125.2 Interfaces for Class `money_punct_byname<char, false>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::money_punct_byname<char, false>` specified in Table 7-257, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-257 libstdcxx - Class `money_punct_byname<char, false>` Function Interfaces**

<code>money_punct_byname&lt;char, false&gt;::money_punct_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>money_punct_byname&lt;char, false&gt;::money_punct_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

**7.1.126 Class `money_punct_byname<char, true>`****7.1.126.1 Class data for `money_punct_byname<char, true>`**

The virtual table for the `std::money_punct_byname<char, true>` class is described by Table 7-258

**Table 7-258 Primary vtable for `money_punct_byname<char, true>`**

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeid</code> for <code>money_punct_byname&lt;char, true&gt;</code>
<code>vfunc[0]:</code>	<code>money_punct_byname&lt;char, true&gt;::~~money_punct_byname()</code>
<code>vfunc[1]:</code>	<code>money_punct_byname&lt;char, true&gt;::~~money_punct_byname()</code>
<code>vfunc[2]:</code>	<code>money_punct&lt;char, true&gt;::do_decimal_point() const</code>
<code>vfunc[3]:</code>	<code>money_punct&lt;char, true&gt;::do_thousands_sep() const</code>
<code>vfunc[4]:</code>	<code>money_punct&lt;char, true&gt;::do_grouping() const</code>
<code>vfunc[5]:</code>	<code>money_punct&lt;char, true&gt;::do_curr_symbol() const</code>
<code>vfunc[6]:</code>	<code>money_punct&lt;char, true&gt;::do_positive_sign() const</code>
<code>vfunc[7]:</code>	<code>money_punct&lt;char, true&gt;::do_negative_sign() const</code>
<code>vfunc[8]:</code>	<code>money_punct&lt;char, true&gt;::do_frac_digits() const</code>
<code>vfunc[9]:</code>	<code>money_punct&lt;char, true&gt;::do_pos_format() const</code>
<code>vfunc[10]:</code>	<code>money_punct&lt;char, true&gt;::do_neg_format() const</code>

The Run Time Type Information for the `std::money_punct_byname<char, true>` class is described by Table 7-259



Table 7-259 typeid for money\_punct\_byname&lt;char, true&gt;

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for money_punct_byname<char, true>

### 7.1.126.2 Interfaces for Class money\_punct\_byname<char, true>

An LSB conforming implementation shall provide the architecture specific methods for Class std::money\_punct\_byname<char, true> specified in Table 7-260, with the full mandatory functionality as described in the referenced underlying specification.

Table 7-260 libstdc++ - Class money\_punct\_byname&lt;char, true&gt; Function Interfaces

money_punct_byname<char, true>::money_punct_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_punct_byname<char, true>::money_punct_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.127 Class money\_punct\_byname<wchar\_t, false>

#### 7.1.127.1 Class data for money\_punct\_byname<wchar\_t, false>

The virtual table for the std::money\_punct\_byname<wchar\_t, false> class is described by Table 7-261

Table 7-261 Primary vtable for money\_punct\_byname&lt;wchar\_t, false&gt;

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for money_punct_byname<wchar_t, false>
vfunc[0]:	money_punct_byname<wchar_t, false>::~~money_punct_byname()
vfunc[1]:	money_punct_byname<wchar_t, false>::~~money_punct_byname()
vfunc[2]:	money_punct<wchar_t, false>::do_decimal_point() const
vfunc[3]:	money_punct<wchar_t, false>::do_thousands_sep() const
vfunc[4]:	money_punct<wchar_t, false>::do_grouping() const
vfunc[5]:	money_punct<wchar_t, false>::do_curr_symbol() const
vfunc[6]:	money_punct<wchar_t, false>::do_positive_sign() const

vfunc[7]:	money_punct<wchar_t, false>::do_negative_sign() const
vfunc[8]:	money_punct<wchar_t, false>::do_frac_digits() const
vfunc[9]:	money_punct<wchar_t, false>::do_pos_format() const
vfunc[10]:	money_punct<wchar_t, false>::do_neg_format() const

The Run Time Type Information for the `std::money_punct_byname<wchar_t, false>` class is described by Table 7-262

**Table 7-262 typeid for `money_punct_byname<wchar_t, false>`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>money_punct_byname&lt;wchar_t, false&gt;</code>

### 7.1.127.2 Interfaces for Class `money_punct_byname<wchar_t, false>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::money_punct_byname<wchar_t, false>` specified in Table 7-263, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-263 `libstdc++` - Class `money_punct_byname<wchar_t, false>` Function Interfaces**

<code>money_punct_byname&lt;wchar_t, false&gt;::money_punct_byname(char const*, unsigned int)</code> [GLIBCXX_3.4] [ISOCXX]
<code>money_punct_byname&lt;wchar_t, false&gt;::money_punct_byname(char const*, unsigned int)</code> [GLIBCXX_3.4] [ISOCXX]

### 7.1.128 Class `money_punct_byname<wchar_t, true>`

#### 7.1.128.1 Class data for `money_punct_byname<wchar_t, true>`

The virtual table for the `std::money_punct_byname<wchar_t, true>` class is described by Table 7-264

**Table 7-264 Primary vtable for `money_punct_byname<wchar_t, true>`**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>money_punct_byname&lt;wchar_t, true&gt;</code>
vfunc[0]:	<code>money_punct_byname&lt;wchar_t, true&gt;::~~money_punct_byname()</code>

vfunc[1]:	money_punct_byname<wchar_t, true>::~~money_punct_byname()
vfunc[2]:	money_punct<wchar_t, true>::do_decimal_point() const
vfunc[3]:	money_punct<wchar_t, true>::do_thousands_sep() const
vfunc[4]:	money_punct<wchar_t, true>::do_grouping() const
vfunc[5]:	money_punct<wchar_t, true>::do_curr_symbol() const
vfunc[6]:	money_punct<wchar_t, true>::do_positive_sign() const
vfunc[7]:	money_punct<wchar_t, true>::do_negative_sign() const
vfunc[8]:	money_punct<wchar_t, true>::do_frac_digits() const
vfunc[9]:	money_punct<wchar_t, true>::do_pos_format() const
vfunc[10]:	money_punct<wchar_t, true>::do_neg_format() const

The Run Time Type Information for the `std::money_punct_byname<wchar_t, true>` class is described by Table 7-265

**Table 7-265 typeid for `money_punct_byname<wchar_t, true>`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>money_punct_byname&lt;wchar_t, true&gt;</code>

### 7.1.128.2 Interfaces for Class `money_punct_byname<wchar_t, true>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::money_punct_byname<wchar_t, true>` specified in Table 7-266, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-266 `libstdc++` - Class `money_punct_byname<wchar_t, true>` Function Interfaces**

<code>money_punct_byname&lt;wchar_t, true&gt;::money_punct_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>money_punct_byname&lt;wchar_t, true&gt;::money_punct_byname(char const*, unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

**7.1.129 Class money\_base****7.1.129.1 Class data for money\_base**

The Run Time Type Information for the `std::money_base` class is described by Table 7-267

**Table 7-267 typeinfo for money\_base**

Base Vtable	vtable for <code>__cxxabiv1::__class_type_info</code>
Name	typeinfo name for <code>money_base</code>

**7.1.129.2 Interfaces for Class money\_base**

No external methods are defined for `libstdcxx` - Class `std::money_base` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.130 Class money\_get<char, istreambuf\_iterator<char, char\_traits<char>>>****7.1.130.1 Class data for money\_get<char, istreambuf\_iterator<char, char\_traits<char>>>**

The virtual table for the `std::money_get<char, std::istreambuf_iterator<char, std::char_traits<char>>>` class is described by Table 7-268

**Table 7-268 Primary vtable for money\_get<char, istreambuf\_iterator<char, char\_traits<char>>>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code>
vfunc[0]:	<code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;::~~money_get()</code>
vfunc[1]:	<code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;::~~money_get()</code>
vfunc[2]:	<code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;::do_get(istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;, bool, ios_base&amp;, _Ios_Iostate&amp;, long double&amp;) const</code>
vfunc[3]:	<code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code>

	<pre>&gt;::do_get(istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt;, bool, ios_base&amp;, _ios_istate&amp;, basic_string&lt;char, char_traits&lt;char&gt;, allocator&lt;char&gt; &gt;&amp;) const</pre>
--	--

The Run Time Type Information for the `std::money_get<char, std::istreambuf_iterator<char, std::char_traits<char> > >` class is described by Table 7-269

**Table 7-269** typeid for `money_get<char, istreambuf_iterator<char, char_traits<char> > >`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;</code>

### 7.1.130.2 Interfaces for Class `money_get<char, istreambuf_iterator<char, char_traits<char> > >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::money_get<char, std::istreambuf_iterator<char, std::char_traits<char> > >` specified in Table 7-270, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-270** `libstdcxx` - Class `money_get<char, istreambuf_iterator<char, char_traits<char> > >` Function Interfaces

<code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;::money_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>money_get&lt;char, istreambuf_iterator&lt;char, char_traits&lt;char&gt; &gt; &gt;::money_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.131 Class `money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >`

#### 7.1.131.1 Class data for `money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >`

The virtual table for the `std::money_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > >` class is described by Table 7-271

**Table 7-271** Primary vtable for `money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >`

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for <code>money_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t,</code>

	<code>char_traits&lt;wchar_t&gt; &gt; &gt;</code>
<code>vfunc[0]:</code>	<code>money_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::~money_get()</code>
<code>vfunc[1]:</code>	<code>money_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::~money_get()</code>
<code>vfunc[2]:</code>	<code>money_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::do_get(istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;, bool, ios_base&amp;, _Ios_Iostate&amp;, long double&amp;) const</code>
<code>vfunc[3]:</code>	<code>money_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::do_get(istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt;, bool, ios_base&amp;, _Ios_Iostate&amp;, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt;&amp;) const</code>

The Run Time Type Information for the `std::money_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > >` class is described by Table 7-272

**Table 7-272 typeinfo for `money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>money_get&lt;wchar_t, istreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;</code>

### 7.1.131.2 Interfaces for Class `money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::money_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > >` specified in Table 7-273, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-273 libstdcxx - Class money\_get<wchar\_t, istreambuf\_iterator<wchar\_t, char\_traits<wchar\_t>>> Function Interfaces**

money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>::money_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>::money_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.132 Class money\_put<char, ostreambuf\_iterator<char, char\_traits<char>>>

#### 7.1.132.1 Class data for money\_put<char, ostreambuf\_iterator<char, char\_traits<char>>>

The virtual table for the std::money\_put<char, std::ostreambuf\_iterator<char, std::char\_traits<char>>> class is described by Table 7-274

**Table 7-274 Primary vtable for money\_put<char, ostreambuf\_iterator<char, char\_traits<char>>>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for money_put<char, ostreambuf_iterator<char, char_traits<char>>>
vfunc[0]:	money_put<char, ostreambuf_iterator<char, char_traits<char>>>::~~money_put()
vfunc[1]:	money_put<char, ostreambuf_iterator<char, char_traits<char>>>::~~money_put()
vfunc[2]:	money_put<char, ostreambuf_iterator<char, char_traits<char>>>::do_put(ostreambuf_iterator<char, char_traits<char>>, bool, ios_base&, char, long double) const
vfunc[3]:	money_put<char, ostreambuf_iterator<char, char_traits<char>>>::do_put(ostreambuf_iterator<char, char_traits<char>>, bool, ios_base&, char, basic_string<char, char_traits<char>, allocator<char>> const&) const

The Run Time Type Information for the std::money\_put<char, std::ostreambuf\_iterator<char, std::char\_traits<char>>> class is described by Table 7-275

**Table 7-275 typeid for money\_put<char, ostreambuf\_iterator<char, char\_traits<char>>>**

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for money_put<char, ostreambuf_iterator<char, char_traits<char>>>

### 7.1.132.2 Interfaces for Class money\_put<char, ostreambuf\_iterator<char, char\_traits<char>>>

An LSB conforming implementation shall provide the architecture specific methods for Class std::money\_put<char, std::ostreambuf\_iterator<char, std::char\_traits<char>>> specified in Table 7-276, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-276 libstdcxx - Class money\_put<char, ostreambuf\_iterator<char, char\_traits<char>>> Function Interfaces**

money_put<char, ostreambuf_iterator<char, char_traits<char>>> >::money_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]
money_put<char, ostreambuf_iterator<char, char_traits<char>>> >::money_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.133 Class money\_put<wchar\_t, ostreambuf\_iterator<wchar\_t, char\_traits<wchar\_t>>>

#### 7.1.133.1 Class data for money\_put<wchar\_t, ostreambuf\_iterator<wchar\_t, char\_traits<wchar\_t>>>

The virtual table for the std::money\_put<wchar\_t, std::ostreambuf\_iterator<wchar\_t, std::char\_traits<wchar\_t>>> class is described by Table 7-277

**Table 7-277 Primary vtable for money\_put<wchar\_t, ostreambuf\_iterator<wchar\_t, char\_traits<wchar\_t>>>**

Base Offset	0
Virtual Base Offset	0
RTTI	typeid for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>
vfunc[0]:	money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>> >::~money_put()
vfunc[1]:	money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>> >::~money_put()
vfunc[2]:	money_put<wchar_t,



	<pre>ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::do_put(ostreambuf_iterator&lt;wcha r_t, char_traits&lt;wchar_t&gt; &gt;, bool, ios_base&amp;, wchar_t, long double) const</pre>
vfunc[3]:	<pre>money_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::do_put(ostreambuf_iterator&lt;wcha r_t, char_traits&lt;wchar_t&gt; &gt;, bool, ios_base&amp;, wchar_t, basic_string&lt;wchar_t, char_traits&lt;wchar_t&gt;, allocator&lt;wchar_t&gt; &gt; const&amp;) const</pre>

The Run Time Type Information for the `std::money_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > >` class is described by Table 7-278

**Table 7-278 typeid for `money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >`**

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>money_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;</code>

### 7.1.133.2 Interfaces for Class `money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::money_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > >` specified in Table 7-279, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-279 libstdcxx - Class `money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >` Function Interfaces**

<code>money_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::money_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>money_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt; &gt; &gt;::money_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

## 7.1.134 Class locale

### 7.1.134.1 Interfaces for Class locale

An LSB conforming implementation shall provide the architecture specific methods for Class `std::locale` specified in Table 7-280, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-280 libstdcxx - Class locale Function Interfaces**

locale::_Impl::_Impl(char const*, unsigned int)(GLIBCXX_3.4) [LSB]
locale::_Impl::_Impl(locale::_Impl const&, unsigned int)(GLIBCXX_3.4) [LSB]
locale::_Impl::_Impl(unsigned int)(GLIBCXX_3.4) [LSB]
locale::_Impl::_Impl(char const*, unsigned int)(GLIBCXX_3.4) [LSB]
locale::_Impl::_Impl(locale::_Impl const&, unsigned int)(GLIBCXX_3.4) [LSB]
locale::_Impl::_Impl(unsigned int)(GLIBCXX_3.4) [LSB]

**7.1.135 Class locale::facet****7.1.135.1 Class data for locale::facet**

The virtual table for the std::locale::facet class is described by Table 7-281

**Table 7-281 Primary vtable for locale::facet**

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for locale::facet
vfunc[0]:	locale::facet::~~facet()
vfunc[1]:	locale::facet::~~facet()

The Run Time Type Information for the std::locale::facet class is described by Table 7-282

**Table 7-282 typeinfo for locale::facet**

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for locale::facet

**7.1.135.2 Interfaces for Class locale::facet**

No external methods are defined for libstdcxx - Class std::locale::facet in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.136 facet functions****7.1.136.1 Interfaces for facet functions**

No external methods are defined for libstdcxx - facet functions in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

**7.1.137 Class \_\_num\_base****7.1.137.1 Class data for \_\_num\_base**

The Run Time Type Information for the std::\_\_num\_base class is described by Table 7-283

Table 7-283

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for __num_base

### 7.1.137.2 Interfaces for Class \_\_num\_base

No external methods are defined for libstdcxx - Class std::\_\_num\_base in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

### 7.1.138 Class num\_get<char, istreambuf\_iterator<char, char\_traits<char>>>

#### 7.1.138.1 Interfaces for Class num\_get<char, istreambuf\_iterator<char, char\_traits<char>>>

An LSB conforming implementation shall provide the architecture specific methods for Class std::num\_get<char, std::istreambuf\_iterator<char, std::char\_traits<char>>> specified in Table 7-284, with the full mandatory functionality as described in the referenced underlying specification.

Table 7-284 libstdcxx - Class num\_get<char, istreambuf\_iterator<char, char\_traits<char>>> Function Interfaces

num_get<char, istreambuf_iterator<char, char_traits<char>>> >::num_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char>>> >::num_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.139 Class num\_get<wchar\_t, istreambuf\_iterator<wchar\_t, char\_traits<wchar\_t>>>

#### 7.1.139.1 Interfaces for Class num\_get<wchar\_t, istreambuf\_iterator<wchar\_t, char\_traits<wchar\_t>>>

An LSB conforming implementation shall provide the architecture specific methods for Class std::num\_get<wchar\_t, std::istreambuf\_iterator<wchar\_t, std::char\_traits<wchar\_t>>> specified in Table 7-285, with the full mandatory functionality as described in the referenced underlying specification.

Table 7-285 libstdcxx - Class num\_get<wchar\_t, istreambuf\_iterator<wchar\_t, char\_traits<wchar\_t>>> Function Interfaces

num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>> >::num_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>> >::num_get(unsigned int)(GLIBCXX_3.4) [ISOCXX]

### 7.1.140 Class `num_put<char, ostreambuf_iterator<char, char_traits<char>>>`

#### 7.1.140.1 Interfaces for Class `num_put<char, ostreambuf_iterator<char, char_traits<char>>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::num_put<char, std::ostreambuf_iterator<char, std::char_traits<char>>>` specified in Table 7-286, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-286 libstdcxx - Class `num_put<char, ostreambuf_iterator<char, char_traits<char>>>` Function Interfaces**

<code>num_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::_M_group_int(char const*, unsigned int, char, ios_base&amp;, char*, char*, int&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>num_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::_M_group_float(char const*, unsigned int, char, char const*, char*, char*, int&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>num_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::_M_pad(char, int, ios_base&amp;, char*, char const*, int&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>num_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::num_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>
<code>num_put&lt;char, ostreambuf_iterator&lt;char, char_traits&lt;char&gt;&gt;&gt;</code> <code>&gt;::num_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]</code>

### 7.1.141 Class `num_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

#### 7.1.141.1 Interfaces for Class `num_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::num_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t>>>` specified in Table 7-287, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-287 libstdcxx - Class `num_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>` Function Interfaces**

<code>num_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::_M_group_int(char const*, unsigned int, wchar_t, ios_base&amp;, wchar_t*, wchar_t*, int&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>num_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::_M_group_float(char const*, unsigned int, wchar_t, wchar_t const*, wchar_t*, wchar_t*, int&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>num_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code> <code>&gt;::_M_pad(wchar_t, int, ios_base&amp;, wchar_t*, wchar_t const*, int&amp;) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>num_put&lt;wchar_t, ostreambuf_iterator&lt;wchar_t, char_traits&lt;wchar_t&gt;&gt;&gt;</code>

>::num_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]
--

num_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::num_put(unsigned int)(GLIBCXX_3.4) [ISOCXX]
--

## 7.1.142 Class `gslice`

### 7.1.142.1 Class data for `gslice`

### 7.1.142.2 Interfaces for Class `gslice`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::gslice` specified in Table 7-288, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-288 libstdc++ - Class `gslice` Function Interfaces**

<code>gslice::_Indexer::_Indexer(unsigned int, valarray&lt;unsigned int&gt; const&amp;, valarray&lt;unsigned int&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
---

<code>gslice::_Indexer::_Indexer(unsigned int, valarray&lt;unsigned int&gt; const&amp;, valarray&lt;unsigned int&gt; const&amp;)(GLIBCXX_3.4) [ISOCXX]</code>
---

## 7.1.143 Class `__basic_file<char>`

### 7.1.143.1 Class data for `__basic_file<char>`

### 7.1.143.2 Interfaces for Class `__basic_file<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::__basic_file<char>` specified in Table 7-289, with the full mandatory functionality as described in the referenced underlying specification.

**Table 7-289 libstdc++ - Class `__basic_file<char>` Function Interfaces**

<code>__basic_file&lt;char&gt;::xsgetn(char*, int)(GLIBCXX_3.4) [ISOCXX]</code>
---

<code>__basic_file&lt;char&gt;::xspu(n(char const*, int)(GLIBCXX_3.4) [ISOCXX]</code>
---

<code>__basic_file&lt;char&gt;::seekoff(long long, _Ios_Seekdir)(GLIBCXX_3.4) [ISOCXX]</code>
---

<code>__basic_file&lt;char&gt;::xspu(n_2(char const*, int, char const*, int)(GLIBCXX_3.4) [ISOCXX]</code>
---

## 7.1.144 Class `_List_node_base`

### 7.1.144.1 Interfaces for Class `_List_node_base`

No external methods are defined for libstdc++ - Class `std::_List_node_base` in this part of the specification. See also the generic specification, ISO/IEC 23360 Part 1.

## 7.1.145 Class `valarray<unsigned int>`

### 7.1.145.1 Class data for `valarray<unsigned int>`

### 7.1.145.2 Interfaces for Class `valarray<unsigned int>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::valarray<unsigned int>` specified in Table 7-290, with the

full mandatory functionality as described in the referenced underlying specification.

**Table 7-290 libstdcxx - Class `valarray<unsigned int>` Function Interfaces**

<code>valarray&lt;unsigned int&gt;::size() const</code> (GLIBCXX_3.4) [ISOCXX]
<code>valarray&lt;unsigned int&gt;::valarray(valarray&lt;unsigned int&gt; const&amp;)</code> (GLIBCXX_3.4) [ISOCXX]
<code>valarray&lt;unsigned int&gt;::valarray(unsigned int)</code> (GLIBCXX_3.4) [ISOCXX]
<code>valarray&lt;unsigned int&gt;::valarray(valarray&lt;unsigned int&gt; const&amp;)</code> (GLIBCXX_3.4) [ISOCXX]
<code>valarray&lt;unsigned int&gt;::valarray(unsigned int)</code> (GLIBCXX_3.4) [ISOCXX]
<code>valarray&lt;unsigned int&gt;::~~valarray()</code> (GLIBCXX_3.4) [ISOCXX]
<code>valarray&lt;unsigned int&gt;::~~valarray()</code> (GLIBCXX_3.4) [ISOCXX]
<code>valarray&lt;unsigned int&gt;::operator[]</code> (unsigned int)(GLIBCXX_3.4) [ISOCXX]

## 7.2 Interface Definitions for libstdcxx

The interfaces defined on the following pages are included in libstdcxx and are defined by this specification. Unless otherwise noted, these interfaces shall be included in the source standard.

Other interfaces listed in Section 7.1 shall behave as described in the referenced base document.

## Annex A GNU Free Documentation License (Informative)

This specification is published under the terms of the GNU Free Documentation License, Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### A.1 PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### A.2 APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

### **A.3 VERBATIM COPYING**

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

### **A.4 COPYING IN QUANTITY**

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.



If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## A.5 MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## **A.6 COMBINING DOCUMENTS**

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the

name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

## **A.7 COLLECTIONS OF DOCUMENTS**

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## **A.8 AGGREGATION WITH INDEPENDENT WORKS**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

## **A.9 TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

## **A.10 TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or

rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## **A.11 FUTURE REVISIONS OF THIS LICENSE**

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## **A.12 How to use this License for your documents**

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.