Clause 20 (Utilities Library) Issues (Revision 3)

** Revision History:

Revision 0 - 22 May 1995 [was Version 1]
Revision 1 - 09 Jul 1995 [was Version 2] (edits before Monterey)
Revision 2 - 26 Sep 1995 (pre-Tokyo)
Revision 3 - 30 Jan 1996 (pre-Santa Cruz)

** Introduction

This document is a summary of issues identified for the Clause 20, identifying resolutions as they are voted on, and offering recommendations for unsolved problems in the Draft where possible.

-------------

** Work Group: Library: Utilities Clause 20
** Issue Number: 20-014
** Title: allocator could be a template again
** Sections: [lib.allocator.requirements], [lib.default.allocator]
** Status: active

** Description:

In many containers, what one allocates is not objects of type T, but objects of type (e.g.) Node<T>. Therefore, in most cases the container would be passed an allocator<T> when what it needs is an allocator<Node<T>>, and possibly other instantiations as well.

** Discussion:

A separate proposal spells out the details.

1. A template constructor:

   template <class U>
   allocator(const allocator<U>&) throw();

2. and a member template containing a typedef:

   template <class U> struct rehost { typedef allocator<U> other; };

These two changes permit a container to construct an allocator of the required type, given one for any other type.

** Proposed Resolution:
As in N0790 = 95-0190, Allocator Cleanup.

** Requester: Myers

-------------

** Work Group: Library: Utilities Clause 20
** Issue Number: 20-010
** Title: auto_ptr specification wrong.
** Sections: 20 [lib.auto.ptr]
** Status: active
** Description:
The specification for auto_ptr in the July Draft did not match the defining proposal, in many details. I don’t know if Greg is satisfied yet.

** Proposed Resolution
Change the specification to match the resolution accepted by the committee.

** Requestor: Greg Colvin
** Owner:

--------------

** Work Group: Library: Utilities Clause 20
** Issue Number: 20-020
** Title: Template constructor for pair<>
** Sections: [lib.pairs]
** Status: active

** Description:
make_pair() doesn’t do what is needed for its most common use: constructing pairs for maps. A small change in pair<> would solve the problem.

** Discussion:

** Proposed Resolution:

Add to pair a template constructor:

    template <class U, class V> pair<const pair<U,V>& p);

Effects: initializes members from the corresponding members of the argument, performing implicit conversions as needed.

** Requestor: Nathan Myers <ncm@cantrip.org>
** Owner:

--------------

** Work Group: Library: Utilities Clause 20
** Issue Number: 20-023
** Title: pair<> should have typedefs
** Sections: [lib.utilities]
** Status: active

** Description:
Given a pair, one cannot get the types of the elements T1 and T2.

** Proposed Resolution:

In [lib.pairs]:

    Add to struct pair:

    typedef T1 first_type;
    typedef T2 second_type;

[Note: this is now part of omnibus proposal N0845 = 96-0027.]

** Requestor: Myers
** Owner:
Closed issues:

** Issue Number: 20-001  
** Title: Allocator needs operator ==  
** Resolution: passed

** Issue Number: 20-002  
** Title: allocator::types<> has no public members  
** Resolution: passed

** Issue Number: 20-003  
** Title: Allocator requirements incomplete  
** Resolution: passed

** Issue Number: 20-004  
** Title: allocator parameter "hint" needs hints on usage  
** Resolution: passed

** Issue Number: 20-005  
** Title: Default allocator member allocate<T>() doesn’t "new T".  
** Resolution: passed

** Issue Number: 20-006  
** Title: allocator::max_size() not documented  
** Resolution: passed

** Issue Number: 20-007  
** Title: C functions asctime() and strftime() use global locale  
** Status: closed by default (Tokyo)

** Issue Number: 20-008  
** Title: construct() and destroy() functions should be members  
** Resolution: passed

** Issue Number: 20-009  
** Title: Allocator member init_page_size() no longer appropriate.  
** Resolution: closed

** Issue Number: 20-011  
** Title: specialization of allocator::types< void > incomplete  
** Resolution: passed

** Issue Number: 20-012  
** Title: get_temporary_buffer has extra argument declared  
** Resolution: passed

** Issue Number: 20-013  
** Title: get_temporary_buffer semantics incomplete  
** Resolution: passed

** Issue Number: 20-015  
** Title: class unary_negate ill-specified.  
** Resolution: passed

** Issue Number: 20-016  
** Title: binder<1st|2nd>::value types wrong.  
** Resolution: passed

** Issue Number: 20-017  
** Title: implicit_cast template wanted  
** Status: closed, no action (Tokyo)

** Issue Number: 20-018  
** Title: auto_ptr::reset to self
** Status: closed, implemented choice 2 (Tokyo)

** Issue Number: 20-019
** Title: no default ctors on many lib classes
** Status: closed, no action (Tokyo)

** Issue Number: 20-021
** Title: should pair<> have a default constructor?
** Status: closed, implemented (Tokyo)

** Issue Number: 20-022
** Title: unary_compose and binary_compose missing.
** Status: closed, no action (Tokyo)