Document number: *N3620*
Date: *2013-03-18*
Project: Programming Language C++
Reply-to: *Kyle Kloepper* *Kyle.Kloepper@riverbed.com*

# Network byte order conversion

## I. Table of Contents



## II. Introduction

This proposal adds support to C++ for converting between host and network byte order.

## III. Motivation and Scope

Converting between host and network byte ordering is an essential component of many network programs. This proposal adds support for the four existing byte order converting function specified by IEEE Std 1003.1-2008 that have been in use since the 1970s. Additionally two generic functions, and their specializations for unsigned integer types, are specified for converting between network and host byte order.

## IV. Impact On the Standard

This is a pure library extension that adds four functions, two template functions, and a number of specializations for the two generic functions.

# V. Design Decisions

The functions `htonl()`, `htons()`, `ntohl()`, and `ntohs()` are intentionally included even though they are not necessary given the `hton<T>()` and `ntoh<T>()` template functions. This is to maintain compatibility with the POSIX standard (IEEE Std 1003.1-2008) and to allow backwards compatibility with existing networking programs.

`hton<T>()` and `ntoh<T>()` were specified as generic templates rather than listing overloads for `hton()` and `ntoh()`. This allows user extension of this function.

# VI. Technical Specifications

## 1. Header <net> synopsis

```
#include <cstdint>

namespace std {
namespace net {

uint32_t htonl(uint32_t host32);
uint16_t htons(uint16_t host16);
uint32_t ntohl(uint32_t net32);
uint32_t ntohs(uint16_t net16);

template <class T>
constexpr T hton(T host);
template <>
constexpr unsigned-integral hton(unsigned-integral host);

template <class T>
constexpr T ntoh(T net);
template <>
constexpr unsigned-integral ntoh(unsigned-integral host);

} // namespace net
} // namespace std
```

- There shall be full specializations of the `hton()` and `ntoh()` templates for the unsigned integer types listed in 3.9.1 and `<cstdint>`.

```
template <class T>
constexpr T hton(T value);
```

- *Returns:* The argument value converted from host to network byte order.

```
template <class T>
constexpr T ntoh(T net);
```

- *Returns:* The argument value converted from network to host byte order.

```
uint32_t htonl(uint32_t host32);
uint16_t htons(uint16_t host16);
```

- *Remarks:* `htonl()` behaves the same as `hton<uint32_t>(). htons()` behaves the same as `hton<uint16_t>()`.
- *Returns:* The argument value converted from host to network byte order.

```
uint32_t ntohl(uint32_t net32);
uint32_t ntohs(uint16_t net16);
```

- *Remarks:* `ntohl()` behaves the same as `ntoh<uint32_t>(). ntohs()` behaves the same as `ntoh<uint16_t>()`.
- *Returns:* The argument value converted from network to host byte order.

## VII. References


- Open Group Base Specifications Issue 7, IEEE Std 1003.1-2008