

# IEEE EUI-64 Unique Node Identifier

**TEP:** 122  
**Group:** Core Working Group  
**Type:** Documentary  
**Status:** Draft  
**TinyOS-Version:** 2.x  
**Author:** Gilman Tolle, Jonathan Hui  
**Draft-Created:** 26-Apr-2006  
**Draft-Version:**  
**Draft-Modified:**  
**Draft-Discuss:** TinyOS Developer List <tinyos-devel at mail.millennium.berkeley.edu>

## Note

This memo documents a part of TinyOS for the TinyOS Community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited. This memo is in full compliance with TEP 1.

## Abstract

A TinyOS application developer may desire a globally-unique node identifier within the IEEE EUI-64 namespace. This document describes the TinyOS components used to access such an identifier.

## 1. Interfaces

A platform that can provide a valid IEEE EUI-64 globally-unique node identifier SHOULD provide it through a component with the signature defined here, enabling platform-independent access to the identifier:

```
configuration LocalIeeeEui64C {  
    provides interface LocalIeeeEui64;  
}
```

The identifier is accessed through the following interface:

```
interface LocalIeeeEui64 {  
    command ieee_eui64_t getId();  
}
```

The `ieee_eui64_t` type is defined in `tos/types/IeeeEui64.h` as:

```
enum { IEEE_EUI64_LENGTH = 8; }

typedef struct ieee_eui64 {
    uint8_t data[IEEE_EUI64_LENGTH];
} ieee_eui64_t;
```

If the platform can provide a valid IEEE EUI-64, the value returned from this call MUST follow the IEEE EUI-64 standard.

If a platform can provide a unique identifier that is not a valid IEEE EUI-64 identifier, it SHOULD provide its unique identifier through a component with a different name and a different interface. The definition of such an interface is outside the scope of this TEP.

## 2. IEEE EUI-64

The IEEE EUI-64 structure is copied here:

company_id				extension identifier			
addr+0	addr+1	addr+2	addr+3	addr+4	addr+5	addr+6	addr+7
AC	DE	48	23	45	67	AB	CD
10101100	11011110	01001000	00100011	01000101	01100111	10101011	11001101
most significant byte				least significant byte			
most-significant bit				least-significant bit			

If provided in byte-addressable media, the original byte-address order of the manufacturer is specified: the most through least significant bytes of the EUI-64 value are contained within the lowest through highest byte addresses, as illustrated above.

See: <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>

The author of the LocalIeeeEui64C component MUST ensure that the getId() call returns a valid EUI-64 identifier that follows the standard, with the bytes in the order described above.

## 3. Implementation Notes

Some TinyOS node platforms contain a unique hardware identifier that can be used to build the EUI-64 node identifier. That hardware identifier may be obtained from several places, e.g. a dedicated serial ID chip or a flash storage device. Users of the interface described in this document MUST NOT require knowledge of how the unique identifier is generated.

The EUI-64 node identifier MUST be available before the Boot.booted() event is signalled. If the EUI-64 is derived from a hardware device, the hardware device should be accessed during the Init portion of the boot sequence.

## 4. Author's Address

Gilman Tolle

Arch Rock Corporation  
657 Mission St. Suite 600  
San Francisco, CA 94105

phone - +1 415 692 0828  
email - [gtolle@archrock.com](mailto:gtolle@archrock.com)

Jonathan Hui  
Arch Rock Corporation  
657 Mission St. Suite 600  
San Francisco, CA 94105

phone - +1 415 692 0828  
email - [jhui@archrock.com](mailto:jhui@archrock.com)