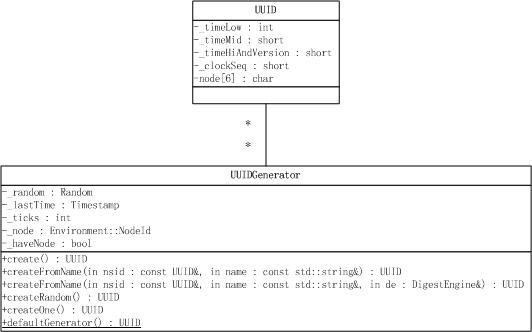
**POCO C++库学习和分析 -- UUID**

**1. 概述**

             通用唯一识别码（英语：Universally Unique Identifier，简称UUID）是一种软件建构的标准。UUID的目的，是让分散式系统中的所有元素，都能有唯一的辨识信息，而不需要通过中央控制端来做辨识信息的指定。一组UUID，是由一串16位组（亦称128位）的16进位数字所构成，是故UUID理论上的总数为216 x 8=2128，约等于3.4 x 1038。UUID的标准型式包含32个16进位数字，以连字号分为五段，形式为8-4-4-4-12的36个字符。  
  
             Poco中提供了Poco::UUID类来存储UUID信息，提供了Poco::UUIDGenerator类生成UUID信息。其类图如下：



**2. Poco::UUID与Poco::UUIDGenerator类**

             Poco::UUID类很简单，用于存储UUID信息，支持所有的值语义，包括所有的关系运算符，也支持字符串之间的相互准换。  
             Poco::UUIDGenerator类用于生成UUID信息。UUIDGenerator类生成UUID信息的方法有3中，分别是基于时间生成、基于名字(数字摘要)生成、基于随机数生成。  
  
             基于时间的函数如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698473)

1. UUID UUIDGenerator::create()
2. {
3. FastMutex::ScopedLock lock(\_mutex);
5. **if** (!\_haveNode)
6. {
7. Environment::nodeId(\_node);
8. \_haveNode = **true**;
9. }
10. Timestamp::UtcTimeVal tv = timeStamp();
11. UInt32 timeLow = UInt32(tv & 0xFFFFFFFF);
12. UInt16 timeMid = UInt16((tv >> 32) & 0xFFFF);
13. UInt16 timeHiAndVersion = UInt16((tv >> 48) & 0x0FFF) + (UUID::UUID\_TIME\_BASED << 12);
14. UInt16 clockSeq = (UInt16(\_random.next() >> 4) & 0x3FFF) | 0x8000;
15. **return** UUID(timeLow, timeMid, timeHiAndVersion, clockSeq, \_node);
16. }

             从上面我们看到，其获取了Mac地址信息和当前时间戳，最终生成uuid信息。Poco中获取Mac地址，可见文章[<平台与环境>](http://blog.csdn.net/arau_sh/article/details/8698406)。  
  
  
             基于名称的生成函数如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698473)

1. UUID UUIDGenerator::createFromName(**const** UUID& nsid, **const** std::string& name, DigestEngine& de)
2. {
3. poco\_assert\_dbg (de.digestLength() >= 16);
5. UUID netNsid = nsid;
6. netNsid.toNetwork();
7. de.reset();
8. de.update(&netNsid.\_timeLow, **sizeof**(netNsid.\_timeLow));
9. de.update(&netNsid.\_timeMid, **sizeof**(netNsid.\_timeMid));
10. de.update(&netNsid.\_timeHiAndVersion, **sizeof**(netNsid.\_timeHiAndVersion));
11. de.update(&netNsid.\_clockSeq, **sizeof**(netNsid.\_clockSeq));
12. de.update(&netNsid.\_node[0], **sizeof**(netNsid.\_node));
13. de.update(name);
14. **char** buffer[16];
15. **const** DigestEngine::Digest& d = de.digest();
16. **for** (**int** i = 0; i < 16; ++i)
17. {
18. buffer[i] = d[i];
19. }
20. **return** UUID(buffer, UUID::UUID\_NAME\_BASED);
21. }

             从上面我们看到，其根据一个已有的uuid和一个字符串生成了16位的数字摘要。获取数字摘要，可见文章[<随机数和数字摘要>](http://blog.csdn.net/arau_sh/article/details/8698418)。  
  
  
             基于随机数的生成函数如下：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698473)

1. UUID UUIDGenerator::createRandom()
2. {
3. **char** buffer[16];
4. RandomInputStream ris;
5. ris.read(buffer, **sizeof**(buffer));
6. **return** UUID(buffer, UUID::UUID\_RANDOM);
7. }

             从上面我们看到，其生成了一串16个字符的数组。获取随机数，可见文章[<随机数和数字摘要>](http://blog.csdn.net/arau_sh/article/details/8698418)。  
  
  
             下面是其使用的一个例子：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698473)

1. #include "Poco/UUID.h"
2. #include "Poco/UUIDGenerator.h"
3. #include <iostream>
4. **using** Poco::UUID;
5. **using** Poco::UUIDGenerator;

8. **int** main(**int** argc, **char**\*\* argv)
9. {
10. UUIDGenerator& generator = UUIDGenerator::defaultGenerator();
11. UUID uuid1(generator.create()); // time based
12. UUID uuid2(generator.createRandom());
13. UUID uuid3(generator.createFromName(UUID::uri(), "http://appinf.com"));
14. std::cout << uuid1.toString() << std::endl;
15. std::cout << uuid2.toString() << std::endl;
16. std::cout << uuid3.toString() << std::endl;
17. **return** 0;
18. }

（版权所有，转载时请注明作者和出处  <http://blog.csdn.net/arau_sh/article/details/8698473>）