

# Is a GUID unique 100% of the time?

Asked 16 years, 3 months ago   Modified 1 year, 2 months ago

Viewed 284k times



## Is a GUID unique 100% of the time?

686

## Will it stay unique over multiple threads?



## language-agnostic

guid



Share



Improve this question

Follow

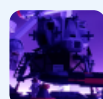
edited Sep 2, 2008 at 15:22



ijs

37.8k ● 36 ● 109 ● 124

asked Sep 2, 2008 at 15:17



David Basarab

73.2k ● 43 ● 130 ● 157

[illegible]

67 First of all, a GUID is not infinite, which means that for the literal meaning of "100% of the time", would mean that no matter how long you keep generating GUID's, they would always be unique. This is not the case. Also, since the original implementation, where the network card unique serial/id/MAC was used to produce a portion of the key is no longer used, for various reasons, a GUID is not really

*globally* unique any more. It is, however, *locally* unique. In other words, if you keep generating GUIDs on a single machine, you will not get duplicates. – [Lasse V. Karlsen](#) Jan 29, 2010 at 23:55

---

37 @ojrac I just choose to round down... :P – [JohannesH](#) Feb 4, 2010 at 9:07

---

549 Every time I generate GUID I feel like I'm stealing one from the Universe. Sometimes I think about evil people who generate much more GUIDs than they need and those wasted GUIDs are so lonely not being used or generated again... – [asavartsov](#) Feb 9, 2013 at 21:40

---

66 @asavartsov I think you'll like [wasteaguid.info](#) ^\_^ – [Navin](#) May 13, 2014 at 3:04

---

25 Answers

Sorted by:

Highest score (default)



544



While each generated GUID is not guaranteed to be unique, the total number of unique keys ( $2^{128}$  or  $3.4 \times 10^{38}$ ) is so large that the probability of the same number being generated twice is very small. For example, consider the observable universe, which contains about  $5 \times 10^{22}$  stars; every star could then have  $6.8 \times 10^{15}$  universally unique GUIDs.

From [Wikipedia](#).

---

These are some good articles on how a GUID is made (for .NET) and how you could get the same guid in the

right situation.

<https://ericlippert.com/2012/04/24/guid-guide-part-one/>

<https://ericlippert.com/2012/04/30/guid-guide-part-two/>

<https://ericlippert.com/2012/05/07/guid-guide-part-three/>

Share Improve this answer

edited Jun 20, 2020 at 9:12

Follow



Community Bot

1 • 1

answered Sep 2, 2008 at 15:19



Adam Davis

93.5k • 60 • 271 • 333

---

141 Wouldn't they be called a UUID, then? ;) – Arafangion May 8, 2009 at 6:23

---

38 A GUID is microsoft's specifica implementation of the UUID standard. So, it's both. Globally unique ID vs Universally unique ID. – Adam Davis May 8, 2009 at 11:49

---

49 Technically, it is not  $2^{128}$ , because in a v4 GUID, you have one hex digit that will always be a 4 (effectively removing 4 bits), and two bits further on are also reserved. However,  $2^{122}$  valid V4 GUIDs still leaves about  $5 \times 10^{36}$ , which will do for me. and for you too. Each star will have to accept just about  $1.1 \times 10^{14}$  GUIDs apiece. – Andrew Shelansky Oct 8, 2010 at 18:53 ✎

---

90 If you're like me, then you'll want to know that  $2^{128}$  written out is approximately:  
34, 028, 236, 692, 093, 846, 346, 337, 460, 743, 177, 000, 000 . Statistically, if you calculated 1000 GUIDs every

second, it would still take trillions of years to get a duplicate.

– [Entity](#) Jul 24, 2012 at 4:01

43 I just thought its funny to read it out so here have fun guys :) Thirty four undecillion twenty eight decillion two hundred thirty six nonillion six hundred ninety two octillion ninety three septillion eight hundred forty six sextillion three hundred forty six quintillion three hundred thirty seven quadrillion four hundred sixty trillion seven hundred forty three billion one hundred seventy seven million – [hjavaaher](#) May 11, 2013 at 4:45



If you are scared of the same GUID values then put two of them next to each other.

179



```
Guid.NewGuid().ToString() +  
Guid.NewGuid().ToString();
```



If you are too paranoid then put three.



Share Improve this answer

answered May 9, 2014 at 18:04

Follow



[Bura Chuhadar](#)

3,751 ● 1 ● 15 ● 17

144 You have to be very, very, very, very paranoid to append 3 GUIDs. – [harsimranb](#) Dec 21, 2016 at 19:22

66 @harsimranb No... very, very, very, very paranoid is 6 GUIDs. Paranoid is one appended, very paranoid is two appended, etc. – [Suamere](#) Nov 27, 2017 at 16:27

143 @Suamere I have created a website for calculating your paranoid level [jogge.github.io/HowParanoidAmI](https://jogge.github.io/HowParanoidAmI) – [Jogge](#) Aug 13, 2018 at 9:26

- 15 @Jogge xD That is amazing, lol. After 9 9's 999999999 in your form, I think Paranoia will a-splode my Browser.  
– Suamere Aug 13, 2018 at 20:16
- 
- 8 @Jogge your website crashed after I put that I am level 10,000 paranoid. Now I am even more paranoid  
– Munene Ndereba Jul 18, 2021 at 13:50
- 



78



The simple answer is yes.

Raymond Chen wrote a [great article](#) on GUIDs and why substrings of GUIDs are *not* guaranteed unique. The article goes in to some depth as to the way GUIDs are generated and the data they use to ensure uniqueness, which should go to some length in explaining *why* they are :-)

Share Improve this answer

Follow

edited Feb 12, 2020 at 19:16



Tomalak

338k ● 68 ● 545 ● 635

answered Sep 2, 2008 at 15:20



ljs

37.8k ● 36 ● 109 ● 124

- 
- 28 I think Chen's article is referring to V1 of the GUID generation algorithm, which uses a MAC address & timestamp -- the current V4 uses a pseudo-random number instead:  
[en.wikipedia.org/wiki/Globally\\_Unique\\_Identifier#Algorithm](https://en.wikipedia.org/wiki/Globally_Unique_Identifier#Algorithm)  
– Barrett Jul 7, 2009 at 14:37
-



49

As a side note, I was playing around with Volume GUIDs in Windows XP. This is a very obscure partition layout with three disks and fourteen volumes.



```
\\?\Volume{23005604-eb1b-11de-85ba-806d6172696f}\
(F:)
\\?\Volume{23005605-eb1b-11de-85ba-806d6172696f}\
(G:)
\\?\Volume{23005606-eb1b-11de-85ba-806d6172696f}\
(H:)
\\?\Volume{23005607-eb1b-11de-85ba-806d6172696f}\
(J:)
\\?\Volume{23005608-eb1b-11de-85ba-806d6172696f}\
(D:)
\\?\Volume{23005609-eb1b-11de-85ba-806d6172696f}\
(P:)
\\?\Volume{2300560b-eb1b-11de-85ba-806d6172696f}\
(K:)
\\?\Volume{2300560c-eb1b-11de-85ba-806d6172696f}\
(L:)
\\?\Volume{2300560d-eb1b-11de-85ba-806d6172696f}\
(M:)
\\?\Volume{2300560e-eb1b-11de-85ba-806d6172696f}\
(N:)
\\?\Volume{2300560f-eb1b-11de-85ba-806d6172696f}\
(O:)
\\?\Volume{23005610-eb1b-11de-85ba-806d6172696f}\
(E:)
\\?\Volume{23005611-eb1b-11de-85ba-806d6172696f}\
(R:)
```

6f = o

69 = i

72 = r

61 = a

6d = m

```
| | | | |
| | | | +--
| | | +----
| | +-----
| +-----
+-----
```

It's not that the GUIDs are very similar but the fact that all GUIDs have the string "mario" in them. Is that a coincidence or is there an explanation behind this?

Now, when [googling for part 4](#) in the GUID I found approx 125.000 hits with volume GUIDs.

**Conclusion:** When it comes to Volume GUIDs they aren't as unique as other GUIDs.

Share Improve this answer

Follow

edited Jan 8, 2015 at 16:28



Alex

7,919 ● 3 ● 47 ● 60

answered Jan 14, 2010 at 7:33



Jonas Engström

5,065 ● 3 ● 40 ● 36

---

42 Remember that Super Mario Bros 3 ad from the 80's? All those people yelling "Mario! Mario! Mario!" around the world upset the randomness of the universe a bit. – [MGOwen](#) Apr 29, 2010 at 5:05

---

31 If you manually un-install Office 2010 with `msiexec`, it lists all the MSI GUID's of the office program. They all spell `0FF1CE`. Seems like Microsoft have a fairly... loose... interpretation of how to generate a GUID ;) – [Mark Henderson](#) Apr 30, 2011 at 3:21

---

3 These partition GUIDs were all created together at 2009-12-17 @ 2:47:45 PM UTC. They are unique to your machine, but putting "mario" as the node identifier is incorrect - it means they're not RFC-4122-compliant. Likewise, the `0FF1CE` GUIDs fall under the "NCS backwards compatibility" section of RFC-4122, but it's unlikely that

Microsoft is following the NCS rules for those values.

– [Stephen Cleary](#) Sep 3, 2011 at 2:44

---

25 I knew it, the Nintendo Security Administration has compromised the random number generators. – [MetaGuru](#) Jul 14, 2014 at 15:37

---

2 maybe it's this same ball park as the name of the company making a mineral water (heard they lead the market) Evian. Spelled backwards gives Naive :-)) – [Mariusz](#) Jun 1, 2015 at 9:23

---



39



It should not happen. However, when .NET is under a heavy load, it is possible to get duplicate guids. I have two different web servers using two different sql servers. I went to merge the data and found I had 15 million guids and 7 duplicates.



Share Improve this answer

answered Jan 29, 2010 at 23:43



Follow



[Tim](#)

423 ● 4 ● 2

12 This would only be true for v1 guids which uses MAC addresses (not machine name) as part of the GUID generation. The v4, which is the de facto STD no longer uses Mac addresses but a pseudo random number. – [Xander](#) Feb 10, 2011 at 13:52

---

20 `Guid.NewGuid` always generates v4 GUIDs (and always has). Tim must have had extremely poor entropy sources. – [Stephen Cleary](#) Sep 3, 2011 at 2:45

---

2 Is that have ever been replicated? that's a huge problem if it's the case. – [Zyo](#) Nov 17, 2015 at 22:16

---



2 Same here while Importing very large Datasets. From about 10-100 Million you get duplicates from Guid.NewGuid  
– [Stephan Baltzer](#) Jun 8, 2019 at 8:57

---

3 @StephanBaltzer No, [that's simply impossible](#). If this actually happened to you there was either a bug in your code which e.g. truncated GUIDs or which confused rows of data. In fact, it would be more likely that there's a bug in the `NewGuid` implementation than that you'd really observe this collision without a bug. But so far no such bug has been reported so I'd bet a nontrivial amount of money that issue was in your code. – [Konrad Rudolph](#) Apr 23, 2021 at 8:19

---



31



Guids are statistically unique. The odds of two different clients generating the same Guid are infinitesimally small (assuming no bugs in the Guid generating code). You may as well worry about your processor glitching due to a cosmic ray and deciding that  $2+2=5$  today.

Multiple threads allocating new guids will get unique values, but you should get that the function you are calling is thread safe. Which environment is this in?

Share Improve this answer

answered Sep 2, 2008 at 15:21

Follow




[Rob Walker](#)

47.4k ● 15 ● 100 ● 137

---

Depending on the guid version you're using based on the specs. Some guids are time and mac addressed based. Meaning for V2 the guid would have to be generated on the same machine at the same picosecond. This is like throwing a bag of 1000 pennies into the air and they all land heads up in a stack on their sides. It is possible but unlikely to the point

that it doesn't bear mentioning as a risk unless lives are at stake. – [Urasquirrel](#) Apr 22, 2021 at 23:30 

---

---



31



Yes, a GUID should always be unique. It is based on both hardware and time, plus a few extra bits to make sure it's unique. I'm sure it's theoretically possible to end up with two identical ones, but extremely unlikely in a real-world scenario.

Here's a great article by Raymond Chen on GUIDs:

<https://blogs.msdn.com/oldnewthing/archive/2008/06/27/8659071.aspx>

Share Improve this answer

Follow

edited Nov 13, 2018 at 9:28



Jogge

1,842 ● 17 ● 40

answered Sep 2, 2008 at 15:19



Eric Z Beard

38.4k ● 27 ● 101 ● 147

---

6 This article is rather old and referring to v1 of GUIDs. v4 does not use hardware/time but a random number algorithm instead.

[en.wikipedia.org/wiki/Globally\\_unique\\_identifier#Algorithm](https://en.wikipedia.org/wiki/Globally_unique_identifier#Algorithm)

– Mani Gandham Dec 13, 2015 at 5:22

---

This link is broken – Marcel Dec 10, 2019 at 7:08

---

Here is the link:

[devblogs.microsoft.com/oldnewthing/20080627-00/?p=21823](https://devblogs.microsoft.com/oldnewthing/20080627-00/?p=21823)

– Olanrewaju O. Joseph Dec 27, 2019 at 1:04

---



Eric Lippert has written a very interesting series of articles about GUIDs.

27



There are on the order  $2^{30}$  personal computers in the world (and of course lots of hand-held devices or non-PC computing devices that have more or less the same levels of computing power, but lets ignore those). Let's assume that we put all those PCs in the world to the task of generating GUIDs; if each one can generate, say,  $2^{20}$  GUIDs per second then after only about  $2^{72}$  seconds -- **one hundred and fifty trillion years** -- you'll have a *very high* chance of generating a collision with your specific GUID. And the odds of collision get pretty good after only thirty trillion years.

- [GUID Guide, part one](#)
- [GUID Guide, part two](#)
- [GUID Guide, part three](#)

Share Improve this answer

edited Jan 23, 2020 at 14:07

Follow


answered Jun 8, 2012 at 14:59



Paolo Moretti

55.8k ● 23 ● 102 ● 93

---

39 ...and he continues in the next paragraph: *"But that's looking for a collision with a specific GUID. [...] So if we put those billion PCs to work generating 122-bits-of-randomness GUIDs, the probability that two of them somewhere in there would collide gets really high after about  $2^{61}$  GUIDs are generated. Since we're assuming that about  $2^{30}$  machines are doing  $2^{20}$  GUIDs per second, **we'd expect a collision after about  $2^{11}$  seconds, which is about an hour.**"* (And finally he explains that, of course, not that many GUIDs are generated.) – [Arjan](#) Oct 26, 2013 at 10:08 

---



25

Theoretically, no, they are not unique. It's possible to generate an identical guid over and over. However, the chances of it happening are so low that you can assume they are unique.



I've read before that the chances are so low that you really should stress about something else--like your server spontaneously combusting or other bugs in your code. That is, assume it's unique and don't build in any code to "catch" duplicates--spend your time on something more likely to happen (i.e. *anything* else).

I [made an attempt](#) to describe the usefulness of GUIDs to my blog audience (non-technical family memebbers). From there (via Wikipedia), the odds of generating a duplicate GUID:

- 1 in  $2^{128}$
- 1 in 340 undecillion (don't worry, undecillion is not on the quiz)



Because  $2^{128}$  is much much larger than  $n$ , we can approximate this to:

$$(1-1/2^{128})^{n(n-1)/2}$$

And because we can assume  $n$  is much much larger than 0, we can approximate that to:

$$(1-1/2^{128})^{n^2/2}$$

Now we can equate this to the "acceptable" probability, let's say 1%:

$$(1-1/2^{128})^{n^2/2} = 0.01$$

Which we solve for  $n$  and get:

$$n = \sqrt{2 \cdot \log 0.01 / \log (1-1/2^{128})}$$

Which Wolfram Alpha gets to be  **$5.598318 \times 10^{19}$**

To put that number into perspective, let's take 10000 machines, each having a 4 core CPU, doing 4Ghz and spending 10000 cycles to generate a Guid and doing nothing else. It would then take ~111 years before they generate a duplicate.

Share Improve this answer

edited Jun 2, 2017 at 3:14

Follow

answered Jan 3, 2017 at 14:56



Cine

4,391 ● 29 ● 48

---

I've edited your post following to [this post](#) - please edit if I did a mistake ;). – [shA.t](#) Jan 4, 2017 at 11:32

---

Hi @Cine, I have the power to edit your response but have opted not to because I want to get a chance for you to rebut it first, I'll probably come by in a month-ish to formally change it if I don't hear from you. I'm fairly certain your math is wrong though. the real equation for determining a 1% chance is this:  $((2^{128} - 1) / 2^{128})^{(n(n-1)/2)} = .01$ . Your exponent is wrong. it isn't just n. You need C(n,2) (aka  $(n*(n-1))/2$ ) to calculate all the combinations when you generate "n" guides. See here for more information – [viggity](#) May 31, 2017 at 19:02 ✎

---

Thanks Cine, I too ended up approximating  $n^2/2$  since its so huge :) – [viggity](#) Jun 2, 2017 at 14:29

---

It would take 10000 machines 111 years to generate every single possible GUID, and then generate a duplicate. A duplicate would however occur long before all possible GUIDs have been generated. I think the approximate time-frame would depends on how 'random' the GUID generation process is. – [George K](#) Jul 4, 2018 at 15:29 ✎

---

- 3 @GeorgeK I think you misunderstood... It would take 10000 machines 111 years to have a 1% chance of encountering a duplicate. But yes, this math ofcourse assumes that the random generator is totally random. – [Cine](#) Jul 5, 2018 at 3:30
-





10



From <http://www.guidgenerator.com/online-guid-generator.aspx>

## What is a GUID?

GUID (or UUID) is an acronym for 'Globally Unique Identifier' (or 'Universally Unique Identifier'). It is a 128-bit integer number used to identify resources. The term GUID is generally used by developers working with Microsoft technologies, while UUID is used everywhere else.

## How unique is a GUID?

128-bits is big enough and the generation algorithm is unique enough that if 1,000,000,000 GUIDs per second were generated for 1 year the probability of a duplicate would be only 50%. Or if every human on Earth generated 600,000,000 GUIDs there would only be a 50% probability of a duplicate.

Share Improve this answer

Follow

edited Jun 20, 2020 at 9:12



Community Bot

1 • 1

answered May 9, 2014 at 17:45



Tono Nam

36k • 81 • 324 • 489

- 
- 13 isn't a 50% chance of a duplicate high enough to cause fear?  
– [disklosr](#) Jan 26, 2015 at 12:59
- 
- 5 @disklosr yeah it's enough to cause fear if your systems are generating 1 billion GUIDs per second. In the extremely unlikely event you are generating that amount then just chain two GUIDs together... – [maxshuty](#) Mar 5, 2019 at 18:35
- 



9



Is a GUID unique 100% of the time?

Not guaranteed, since there are several ways of generating one. However, you can try to calculate the chance of creating two GUIDs that are identical and you get the idea: a GUID has 128 bits, hence, there are  $2^{128}$  distinct GUIDs – **much** more than there are stars in the known universe. Read the [wikipedia article](#) for more details.

Share Improve this answer

Follow

answered Sep 2, 2008 at 15:20



[Konrad Rudolph](#)

545k ● 139 ● 956 ● 1.2k



8



I experienced a duplicate GUID.

I use the Neat Receipts desktop scanner and it comes with proprietary database software. The software has a sync to cloud feature, and I kept getting an error upon syncing. A gander at the logs revealed the awesome line:



```
"errors":[{"code":1,"message":"creator_guid: is  
already taken","guid":"C83E5734-D77A-4B09-  
B8C1-9623CAC7B167"}]}
```

I was a bit in disbelief, but surely enough, when I found a way into my local neatworks database and deleted the record containing that GUID, the error stopped occurring.

So to answer your question with anecdotal evidence, no. A duplicate is possible. But it is likely that the reason it happened wasn't due to chance, but due to standard practice not being adhered to in some way. (I am just not that lucky) However, I cannot say for sure. It isn't my software.

Their customer support was EXTREMELY courteous and helpful, but they must have never encountered this issue before because after 3+ hours on the phone with them, they didn't find the solution. (FWIW, I am very impressed by Neat, and this glitch, however frustrating, didn't change my opinion of their product.)

Share Improve this answer

Follow

edited Jul 22, 2015 at 10:42



Mohit Jain

30.5k ● 8 ● 78 ● 103

answered Mar 8, 2013 at 19:01



exintrovert

117 ● 1 ● 3

---

27 Don't believe you got a duplicate. There was probably something else involved, like number was not truly random or problem in the sync process, or system tried to record twice, etc. A software issue is much more likely than you getting a duplicate GUID. – [orad](#) Sep 19, 2013 at 21:22

---

[MSDN](#):

There is a very low probability that the value of the new Guid is all zeroes or equal to any other Guid.

Share Improve this answer

answered Sep 2, 2008 at 15:20

Follow



[Jakub Štunc](#)

35.8k ● 25 ● 91 ● 115

If your system clock is set properly and hasn't wrapped around, and if your NIC has its own MAC (i.e. you haven't set a custom MAC) and your NIC vendor has not been recycling MACs (which they are not supposed to do but which has been known to occur), and if your system's GUID generation function is properly implemented, then your system will never generate duplicate GUIDs.

If everyone on earth who is generating GUIDs follows those rules then your GUIDs will be globally unique.

In practice, the number of people who break the rules is low, and their GUIDs are unlikely to "escape". Conflicts

are statistically improbable.

Share Improve this answer

answered Sep 2, 2008 at 16:16

Follow



DrPizza

18.3k ● 7 ● 42 ● 53

---

12 This would only be true for v1 guids. The v4, which is the de facto STD no longer uses Mac addresses but a pseudo random number. – [Pita.O](#) Jun 17, 2010 at 8:57

---

1 "then your system will never generate duplicate GUIDs" Even if all the rules were followed for a v1 guid as you say, your system could still generate duplicates. You are more correct at the bottom when you state "Conflicts are statistically improbable." – [Nick Meldrum](#) Feb 18, 2011 at 8:41

---



3



GUID algorithms are usually implemented according to the v4 GUID specification, which is essentially a pseudo-random string. Sadly, these fall into the category of **"likely non-unique"**, from Wikipedia (I don't know why so many people ignore this bit): "... other GUID versions have different uniqueness properties and probabilities, ranging from guaranteed uniqueness to likely non-uniqueness."

The pseudo-random properties of V8's JavaScript `Math.random()` are TERRIBLE at uniqueness, with collisions often coming after only a few thousand iterations, but V8 isn't the only culprit. I've seen real-world GUID collisions using both PHP and Ruby implementations of v4 GUIDs.

Because it's becoming more and more common to scale ID generation across multiple clients, and clusters of servers, entropy takes a big hit -- the chances of the same random seed being used to generate an ID escalate (time is often used as a random seed in pseudo-random generators), and GUID collisions escalate from "likely non-unique" to "very likely to cause lots of trouble".

To solve this problem, I set out to create an ID algorithm that could scale safely, and make better guarantees against collision. It does so by using the timestamp, an in-memory client counter, client fingerprint, and random characters. The combination of factors creates an additive complexity that is particularly resistant to collision, even if you scale it across a number of hosts:

<http://usecuid.org/>

Share Improve this answer

answered Jun 27, 2013 at 6:39

Follow



**Eric Elliott**

4,751 ● 1 ● 28 ● 33



3



I have experienced the GUIDs not being unique during multi-threaded/multi-process unit-testing (too?). I guess that has to do with, all other things being equal, the identical seeding (or lack of seeding) of pseudo random generators. I was using it for generating unique file names. I found the OS is much better at doing that :)



# Trolling alert

You ask if GUIDs are 100% unique. That depends on the number of GUIDs it must be unique among. As the number of GUIDs approach infinity, the probability for duplicate GUIDs approach 100%.

Share Improve this answer

edited Aug 24, 2017 at 13:28

Follow

answered Aug 24, 2017 at 12:46



**Robert Jørgensgaard Engdahl**

3,354 ● 23 ● 34

---

I experienced that too, but if I talk about it I get ridiculed :D  
– [Spikolynn](#) May 20, 2023 at 10:16

---



3



I think that when people bury their thoughts and fears in statistics, they tend to forget the obvious. If a system is truly random, then the result you are least likely to expect (all ones, say) is equally as likely as any other unexpected value (all zeros, say). Neither fact prevents these occurring in succession, nor within the first pair of samples (even though that would be statistically "truly shocking"). And that's the problem with measuring chance: it ignores criticality (and rotten luck) entirely.

IF it ever happened, what's the outcome? Does your software stop working? Does someone get injured? Does

someone die? Does the world explode?

The more extreme the criticality, the worse the word "probability" sits in the mouth. In the end, chaining GUIDs (or XORing them, or whatever) is what you do when you regard (subjectively) your particular criticality (and your feeling of "luckiness") to be unacceptable. And if it could end the world, then please on behalf of all of us not involved in nuclear experiments in the Large Hadron Collider, don't use GUIDs or anything else indeterministic!

Share Improve this answer

Follow

edited Nov 27, 2021 at 0:38



Wai Ha Lee

8,795 ● 97 ● 59 ● 94

answered Nov 26, 2021 at 14:34



Alex T

415 ● 4 ● 8



2

In a more general sense, this is known as the "birthday problem" or "birthday paradox". Wikipedia has a pretty good overview at: [Wikipedia - Birthday Problem](#)



In very rough terms, the square root of the size of the pool is a rough approximation of when you can expect a 50% chance of a duplicate. The article includes a probability table of pool size and various probabilities, including a row for  $2^{128}$ . So for a 1% probability of collision you would expect to randomly pick  $2.6 \times 10^{18}$  128-bit numbers. A 50% chance requires  $2.2 \times 10^{19}$  picks, while  $\text{SQRT}(2^{128})$  is  $1.8 \times 10^{19}$ .





Of course, that is just the ideal case of a truly random process. As others mentioned, a lot is riding on the that *random* aspect - just how good is the generator and seed? It would be nice if there was some hardware support to assist with this process which would be more bullet-proof except that anything can be spoofed or virtualized. I suspect that might be the reason why MAC addresses/time-stamps are no longer incorporated.

Share Improve this answer

answered Sep 22, 2017 at 1:08

Follow



mszil

61 ● 2

---

I think the MAC problem was anonymity. I believe using an identifier such as a MAC address in a way that could be reversed was a privacy concern. I believe true random in hardware is very difficult? Cloudflare uses a camera and a row of lava lamps, however I think that with a precise understanding of physics, even that is not random?

Cloudflares lava lamp RNG:

[popularmechanics.com/technology/security/news/a28921/...](http://popularmechanics.com/technology/security/news/a28921/...)

– Jeff Block Aug 28, 2018 at 23:48

---



2



The Answer of "*Is a GUID is 100% unique?*" is simply "No" .

- If You want 100% uniqueness of GUID then do following.

1. generate GUID

2. check if that GUID is Exist in your table column where you are looking for uniqueness



3. if exist then goto step 1 else step 4

4. use this GUID as unique.

Share Improve this answer

answered Aug 14, 2018 at 11:55

Follow



[Baba Khedkar](#)

141 ● 7

2 This does not make it unique. Your algorithm does not save the newly created GUID in the table. The next time you create a GUID it could collide with one before. If you were to insert the GUID to the table, the GUID could already had been inserted by another peer in between you checked for uniqueness and you inserted the GUID into the table. The GUID is only unique within YOUR system, so if you were to import or merge two databases they could still collide. Also GUID are often used when you do not have access to a centrilized database. If you had why not just pull an ID from the database? – [Jogge](#) May 25, 2020 at 11:41

I set a unique index on the column and handle the DB exception. And before anyone thinks that exceptions waste valuable computing time: Well, given the probability, I think it's a justifiable investment. – [Marcus](#) Feb 2 at 23:33



2

For more better result the best way is to append the GUID with the timestamp (Just to make sure that it stays unique)



```
Guid.NewGuid().ToString() +  
DateTime.Now.ToString();
```



Share Improve this answer

answered Aug 20, 2019 at 9:57

Follow



Adithya Sai

1,740 ● 3 ● 21 ● 35

---

What if you get two collisions in the same second?

– [Wai Ha Lee](#) Aug 20, 2019 at 11:36

---

That's the worst-case but still, we can't have the same two Guid's generated at the same time. – [Adithya Sai](#) Aug 20, 2019 at 12:41

---

Somewhere they argue, that one should copy from the answer on SO, not the question, but I am not so sure now....  
– [Marcel](#) Dec 10, 2019 at 7:07

- 
- 1 How about Guid.NewGuid().ToString().Replace("-", "") + DateTime.Now.Ticks .... Not questionable about uniqueness and can be used as a primary key – [d-coder](#) Jun 10, 2020 at 23:02
- 



The hardest part is not about generating a duplicated Guid.

0



The hardest part is designed a database to store all of the generated ones to check if it is actually duplicated.



From WIKI:



For example, the number of random version 4 UUIDs which need to be generated in order to have a 50% probability of at least one collision is 2.71 quintillion, computed as follows:

[enter image description here](#)

This number is equivalent to generating 1 billion UUIDs per second for about 85 years, and a file containing this many UUIDs, at 16 bytes per UUID, would be about 45 exabytes, many times larger than the largest databases currently in existence, which are on the order of hundreds of petabytes

Share Improve this answer

answered Jan 9, 2019 at 10:09

Follow



Trong Hiep Le

39 ● 1 ● 5



GUID stands for Global Unique Identifier

0

In Brief: (the clue is in the name)



In Detail: GUIDs are designed to be unique; they are calculated using a random method based on the computers clock and computer itself, if you are creating many GUIDs at the same millisecond on the same machine it is possible they may match but for almost all normal operations they should be considered unique.



Share Improve this answer

answered Apr 3, 2019 at 22:59

Follow



Benjamin Roberts

546 ● 5 ● 12



-1

So incredibly low as to be almost 0, but never 0. Ask me how I know... I actually came here to find out just how improbable it really was because I just discovered there



are two pieces of data who share the same GUID in one of my dbs...



Share Improve this answer

answered Oct 12, 2023 at 14:41

Follow



[Viktor Moberg](#)

29 ● 1

---

Please don't duplicate existing answers – [Nico Haase](#) Oct 12, 2023 at 14:44

---



-2

Enough GUIDs to assign one to each and every hypothetical grain of sand on every hypothetical planet around each and every star in the visible universe.



Enough so that if every computer in the world generates 1000 GUIDs a second for 200 years, there might (MIGHT) be a collision.



Given the number of current local uses for GUIDs (one sequence per table per database for instance) it is extraordinarily unlikely to ever be a problem for us limited creatures (and machines with lifetimes that are usually less than a decade if not a year or two for mobile phones).

... Can we close this thread now?

Share Improve this answer

answered Aug 9, 2021 at 17:39

Follow



[William M. Rawls](#)

135 ● 10



**Highly active question.** Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.