# Universal citekey

In Papers 2.0, we introduced 'Magic Citations', an incredibly easy user interface to insert citations and format a manuscript. We also introduced a new way to generate a citekey for each of your papers, which we call a 'universal citekey' generator. By using a hash on the title or on the doi, Papers2 is able to generate a citekey that will be different for two different papers, and that will still be the same for two different users, even if they do not share their Papers library (and as long as they both have the correct title or doi). This document is intended to describe how universal citekeys are generated by Papers. This is a very technical description, that will only be of interest to the geekiest of you. It will also be of interest for other app developers that would like to use the same algorithm and generate the same universal citekey (that would be great!).

## Citekey Definition

For the purpose of this document, a citekey is an identifier that can be used to reference a publication in a manuscript. For instance, a citekey could be 'Smith:1997'. The citekey is typically embedded in some kind of code-like wrapper, for instance '{Smith:1997}', which can then be inserted in the text of a manuscript, for instance:

```
"... as shown previously {Smith:1997}, pigs can fly ..."
```

The citekey then acts as a placeholder for the citation of a publication, or of several publications. Thanks to the distinctive syntax in which they are wrapped (e.g. the curly braces), the citekeys can later be easily detected and resolved back to the full paper reference. This in turn allows easy replacement with a properly formatted citation, dependent on the desired style, e.g. to get:

```
"... as shown previously [12], pigs can fly ..."
```

All reference managers use the citekey placeholder trick one way or another, though it's not always called a citekey.

Though they all **use** citekeys, different software packages have introduced various strategies to **generate** citekeys:

- EndNote generates a "temporary citation" (another name for a citekey) that includes the first author's last name, year of publication, and a unique record number specific for that entry in the EndNote library, to produce for instance `'{Smith, 1997 #42}'`
- BibTeX lets the user set any citekey they want for a paper. To make it easier, it can autogenerate a citekey based on various fields, like the first author name, year, title,etc… The most popular scheme typically uses the first author's last name and year of publication separated by a colon, and followed by a suffix of 1 or 2 letters to disambiguate publications with same first author and year; a BibTeX citekey is then included as part of a cite command, and would for instance look like `'\cite{Smith:1997a}'`

In all cases, a citekey is thus designed to be short, human-readable (so the user is hopefully able to know which paper this is at first glance) and unique enough that each citekey can be unequivocally resolved back to a specific publication in your library.

## Shortcomings of Citekeys

A good citekey scheme should have these basic properties:

- human-readable
- easily parsable by our program
- easily disambiguated by the user

But here are some more properties that are useful when going beyond one manuscript or beyond one user of the manuscript:

- compatibility across multiple manuscripts for an individual author
- compatibility across multiple manuscripts and multiple libraries for co-authors collaboration

It is relatively easy to create a scheme that produces unique citekeys for one library for one user. Both BibTeX and EndNote schemes fit the first set of requirements, with maybe the BibTeX citekey generally shorter, thus a little better. But Both EndNote and BibTeX citekey schemes have shortcomings when it comes to sharing citekeys across multiple libraries or manuscripts.

The EndNote record number is specific for a library, which makes it more difficult to copy-paste a citekey from a manuscript to another, unless one keeps a unique large library with all the papers ever cited (probably in fact the solution adopted by a number of users). But then, the number of reference in that library can be quite large, and one ends up with 4 or 5 digit record number in the citekey, which make those citekeys bulkier and more confusing to parse for the eyes. That big library also has to be shared between the co-authors and kept in sync.

A BibTeX citekey can stay the same between manuscripts and between libraries. However, the probability of citekey collision is still high. If 2 authors work in the same field, they are likely to already have the same publications in their libraries when they start collaborating on a manuscript, but maybe with different citekeys. At some point, they still have to agree on a common citekey (which in turn may result in inconsistent citekeys with previous manuscripts for at least one of the authors).

BibTeX citekeys typically use the first author's last name and year of publication. This is a good way to define a publication with a good level of specificity, and collisions are relatively rare. But they are frequent enough that it remains an issue for most libraries of more than a few hundred entries (whatever the software used). For some common author names, the collision issue is actually very bad, and that can be a source of a lot of headaches. For example, here is the results from doing Pubmed searches on some extreme cases (and it is getting worse in recent years):

- first author == 'Smith' and year == '1999' —-> 1,167 results
- first author == 'Smith' and year == '2009' —-> 1,535 results
- first author == 'Wang' and year == '1999' —-> 1,992 results
- first author == 'Wang' and year == '2009' —-> 7,574 results
- first author == 'Lee' and year == '1999' —-> 1,574 results
- first author == 'Lee' and year == '2009' —-> 5,030 results
- first author == 'Zhang' and year == '1999' —-> 1,278 results
- first author == 'Zhang' and year == '2009' —-> 6,124 results

More specifically, citekey ambiguities will result in the following situations:

- Unknown citekey: even if 2 authors have the same publications in their libraries, they may have used a different citekey generation scheme in BibTeX, or have a different suffix (e.g. 'Smith:1997a' and 'Smith:1997b' may in fact be the same paper), or in the case of EndNote have a different record number.
- False matches: 2 papers could be different, but have the same citekey for the 2 different users (e.g. 'Smith:1997a' may represent 2 different papers for 2 collaborators)
- Both situations are also applicable to a single person working on several manuscripts over the year, trying to copy-paste pieces from one manuscript to the next, and maybe using different libraries.

The different software packages have different ways of dealing with these problems, but the only thing one can really do is warn the user when an unknown or ambiguous citekey is found. In some cases, the software may try to solve the ambiguity silently, which can in fact be considered worse, as this could lead to a wrong reference going unnoticed. The worst situation is a 'false match' (the second bullet point above). This can easily go undetected as no warning will be given, but both users will end up with a different result, and a different list of references in the bibliography. When collaborating on a manuscript, the only safe thing to do is keep the libraries of the multiple authors in sync, in addition to the different versions of the manuscript.

For Papers, we thought we might be able to devise a smarter scheme for the citekey generation, and minimize or fully tackle some of those issues.

## Universal Citekey: General Strategy

For our universal citekey scheme, we decided to use the Bibtex citekey scheme as a starting point. The first reason was to avoid reinventing the wheel, and at least starting with something that would be fully compatible with one of the existing scheme already existing (BibTeX citekeys). Relying on the first author last name and the year of publication is well understood and well accepted by authors and publishers, is widely used for all existing citekey scheme (including EndNote) and is also quite good at narrowing things down.

We estimated the problem of the BibTeX citekeys was the added suffix, which is typically just an 'a', then 'b' when another similar paper is used, then 'c', etc… The problem, as pointed out in the previous section, is that the suffix can easily become inconsistent between different users, or even for the same user, between different libraries. We wanted to be able to generate a citekey that should be the same if the publication is the same, independently of the user and regardless of how the publication was obtained. One common workaround is to use the first page number, but for some journals that can be a 5-digit number and/or contain also letters, which can result in long unsightly citekeys. Instead, we decided to generate a 2-letter suffix, applying some kind of hash on some unambiguous metadata of the publication, like the title or the doi. The resulting citekey would be something like `'Smith:1997tu'` and would have a lot of properties that would warrant the name of **"universal"** citekey.

Here are a few characteristics that we figured this hash should have:

- 2 **similar** but distinct papers should **not** have have the same citekey (for instance, 2 papers from the same author, same year, with similar title; or 2 papers back to back on the same subject in the same issue of a journal, with similar titles; etc…). Fortunately, that's a fundamental property of even mediocre hashes.
- The algorithm should be such that the citekey could be generated by anyone, not restricted to Papers. In fact, we'd love other software package to use it as well, so the name *universal* citekey is even more apt (apter?). This means as much as possible we should use well-known algorithms, and keep it simple.
- We do not need irreversible cryptography-safe hashes. In fact, in theory, a reversible computation would be considered a feature: getting back the metadata from the 2-letter hash would be nice (unrealistic for a title or a doi, though!)

As you will see in the implementation details below, the hash makes use of CRC32.

One final consideration. A useful scheme should allow a practical implementation of both these tasks:

- generating the citekey: going from a publication to a citekey
- resolving the citekey: going from the citekey to the publication

We thus provide here not just a specification, but some practical implementation details that ensure an efficient behavior for both tasks.

## Universal Citekey Generation

As should now be clear, Papers universal citekeys have three components in that order:

- the citekey base, typically the first author name
- the publication year
- the citekey suffix, a 2-letter hash generated from the doi or the title

It looks like this, with just an additional colon to separate the citekey base and year:

```
 _________________     ________   ________
|                 |   |        | |        |
|  citekey base   | : |  year  | | suffix |
|_________________|   |________| |________|
```

If there is no publication year, it is simply omitted (the colon remains). If the year is negative, then the citekey will include the minus sign, or the era abbreviation if applicable.

We explain below in more details how the citekey base and suffix are generated.

### Citekey Base

In most cases, the citekey base is built from the last name of the first author. For instance, a paper with first author John Smith will have the citekey base `'Smith'` (without the quotes).

There are a few more details:

- particles and suffixes are included in the last name
- if the author is unknown, the name is replaced with 'Anonymous'
- for periodicals (journals, magazines,…), use the periodical abbreviation instead of the author name, or 'Unknown' if the name of the journal is unknown
- for websites, conferences or other media files, use the title instead of the author name, or 'Untitled' if it has no title
- the last name (or abbreviation, or title) is transformed to get the canonical string, as detailed on **http://unicode.org/reports/tr15/**. This normalizes accentuated characters
- whitespace characters are replaced with dashes, and capitalization is preserved

Note that to help with citekey resolution, the citekey base is precomputed and cached in the Papers database for each publication. This helps later narrow down the number of candidate publications for a given citekey, with just one database query (see section 'Universal Citekey Resolution').

### Citekey Hash

The 2-letter hash for the universal citekey can be generated from either the doi or the title. For citekey generation in Papers, the doi hash is prefered, with the title hash as the alternative fallback. The reason Papers favors the doi is that it seems like a more reproducible and consistent metadata information.

In both cases, the implementation uses the CRC32 hash. This hash returns a 32-bit number. To get this number down to a 2-letter suffix, we apply various modulos to it, as explained below.

**Universal Citekey from DOI.**

The 2-letter hash is in the range ba, bb,…, kz, determined based on the integer `n_2`, which has a value between 0 and 259, and as follows:

- `n_1` = CRC32 applied to the doi string
- `n_2 = n_1 mod 260`
- `n_3 = n_2 div 26` —> first letter of the suffix (b=0, c=1,…, k=9)
- `n_4 = n_2 mod 26` —> second letter of the suffix (a=0, b=1,…, z=25)

**Universal Citekey from Title**

The 2-letter hash is in the range ta, tb,…, wz, determined based on the integer `n_2`, which has a value between 0 and 102, and as follows:

- the title string is transformed into its equivalent canonical string, all lowercase
- `n_1` = CRC32 applied to the title string
- `n_2 = n_1 mod 102`
- `n_3 = n_2 div 26` —> first letter of the suffix (t=0, u=1, v=2, w=3)
- `n_4 = n_2 mod 26` —> second letter of the suffix (a=0, b=1,…, z=25)

### Namespaces

The citekey hash is generated in such a way that the values from the doi or the title cannot overlap. In general, we defined the following 'namespaces' for the 2-letter suffix:

- range aa-az (26 values): unused to avoid collision with most suffixes already in use in existing BibTeX libraries
- range ba-kz (260 values): citekeys from DOI (see above)
- range la-oz (104 values): unused to reduce confusion between letter l and digit 1, letters m and n, and letter o and digit 0
- range pa-sz (104 values): reserved for future use by Mekentosj
- range ta-wz (104 values): citekeys from title (see above)
- range ya-zz (52 values): unused

This separation reduces the number of possible hashes, but makes it possible to resolve citekeys with either DOI or title, simply based on the first letter of the suffix.

### Final Ouput

The final universal citekey used by Papers is the one generated from the DOI if the DOI is available. Otherwise, Papers will use the one generated from the title. To clarify, both hashes are valid, and could be used by the implementation. The fact that Papers favors the doi hash is just an implementation choice, but other implementations could decide to output the hash based on the title, or even to use either one randomly if the implementer thinks it's funny. As will be explained in the next section, for citekey resolution, both hashes will be considered anyway.

Not Detailed here, Papers will also fall back on some other hashes when neither the doi or title are present, or when the metadata is of low quality and incomplete. In other words, even if the publication has a title, Papers may decide to not use the universal citekey if the metadata appears crappy in general (that's to avoid generating a universal citekey that's anyway wrong and can only cause more confusion). In such cases, the hope for a universal citekey become unrealistic anyway, so these cases won't be covered here (we might document these cases at a later date).

In practice, here is how Papers dynamically outputs the final universal citekey when necessary:

- if the DOI is available, generate the universal citekey from DOI
- if the title is available, and the metadata is reliable (based on some heuristics), generate the universal citekey from title
- otherwise, generate a citekey specific to the library (in short: a base-64 version of the first 6 bytes of the UUID in the library); the rationale is that Papers should not generate a universal citekey unless it can be later reliably resolved on other machines, based on somebody else's library; if the metadata is not good, all hope is lost

## Universal Citekey Resolution

When Papers parses the contents of a manuscript, the citekeys are extracted from the text. The citekeys are then resolved to determine the publication to which they correspond, as follows:

- bail out if the citekey does not follow the expected pattern of a universal citekey

```
 _________________     ________   ________
|                 |   |        | |        |
|  citekey base   | : |  year  | | suffix |
|_________________|   |________| |________|
```

- check the 2-letter suffix, and determine wether it is a doi or title-based hash

- query the database to get all the publications that fit the citekey base and year of publication
- for each of the candidate publications, generate the universal citekey based on the doi or title
- the publications that have the same universal citekeys are considered primary hits

If there is exactly 1 hit, this is the publication that correspond to the citekey. If there is no hit, or more than 1 hit, Papers will display warnings to the user, and request assistance to solve the conflict. Since it already know the first author and year, Papers can suggest some potential candidates.

## Implementation

If you are interested in the gory details, there are 2 ways you can play more with universal citekeys:

1. **Universal Citekey CSL Style.** In Papers 2.1.9, we added support for Papers2-specific CSL variables, that can be used to create a CSL style that will output those values using Papers2 itself (but of course, the same CSL used with other software will not output those values, hence the use of the 'papers2' prefix). Here are the variable names:

   ```
   papers2_universal_citekey
   papers2_universal_citekey_from_doi
   papers2_universal_citekey_from_title
   ```

   Here is an example style that will generate a bibliography with the universal citekeys:

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <style xmlns="http://purl.org/net/xbiblio/csl" class="in-text" version="1.0" demote-non-dropping-particle="sort-only">
       <info>
           <title>Papers2 Universal Citekey Output</title>
           <id>http://mekentosj.com/csl_styles/papers2-universal-citekey-output</id>
           <link href="http://mekentosj.com/csl_styles/papers2-universal-citekey-output" rel="self"/>
           <author>
               <name>Charles Parnot</name>
               <email>charles@mekentosj.com</email>
           </author>
           <category field="generic-base"/>
           <updated/>
           <summary>This style outputs the universal citekey, based on either the title or doi. It is only meant to be used for develop
           <rights>This work is licensed under a Creative Commons Attribution-Share Alike 3.0 License: http://creativecommons.org/licen
       </info>
       <citation collapse="citation-number">
           <sort>
               <key variable="citation-number"/>
           </sort>
           <layout prefix="[" suffix ="]" delimiter=",">
               <text variable="citation-number"/>
           </layout>
       </citation>
       <bibliography>
           <layout>
               <!-- &#10; is a newline character -->
               <group prefix='{&#10;    ' suffix='&#10;}' delimiter='&#10;    '>
                   <text variable="citation-number" prefix="citation-number: "/>
                   <text variable="DOI" prefix="DOI: " />
                   <text variable="title" prefix="title: "  />
                   <text variable="papers2_universal_citekey" prefix="universal_citekey: " />
                   <text variable="papers2_universal_citekey_from_doi" prefix="universal_citekey_from_doi: " />
                   <text variable="papers2_universal_citekey_from_title" prefix="universal_citekey_from_title: " />
               </group>
           </layout>
       </bibliography>
   </style>
   ```

To use this style, create a new file called `papers2-universal-citekey-output.csl` with the above XML content. Then save it into the 'Styles' folder in your Papers2 library. It can then be used with the Magic Manuscripts feature. The universal citekeys will be output when choosing the 'Format Manuscirpt' option, as part of the bibliography section. Or if you set this style as your favorite style in the Manuscripts preferences, you can get this same universal citekey output when selecting some papers, and then using the menu item 'Edit > Copy As… > Reference'. The clipboard will be populated with the list of papers and universal citekeys, and it can then be pasted in the document of your choice.

2. **Javascript Implementation.** You can check the **javascript implementation** from github. The repository includes 3 files:

```
universal-citekey.html
universal-citekey.js
universal-citekey.css
```

You can open the `universal-citekey.html` file in Safari (after downloading the repository) to test various combinations of DOI and title. The file `universal-citekey.js` contains the code that generates the universal citekey base and suffix.