Additional Criteria

Other desirable characteristics:

I don't actually know if these are important.

1) have some sort of error detection, like check digit, so user can tell if they have cut/pasted wrong. ISBN has check digit, as do the handles generated by the lsdi noid generation. (And OCR can tell if the identifier is wrong. why should we care about OCR? See cell phone scenario below)

2) have a *fixed length*, and *limited syntax*. Why? If we pick a type of identifier with a large identifier space, but an extremely rigid syntax, the identifier is then more easily found in html pages, and manipulated by such things as greasemonkey, or libX. In addition, if we are dealing with an identifier that has been printed, and then needs to be recognized by OCR (think about a cell phone camera grabbing the doi's from a printed pdf, and bringing them up on your laptop screen), the job is much easier (I mean, actually possible) if the identifiers implement a very rigid syntax. arxiv identifers, namely, arxiv:xxxx , are really quite good in this respect, I think. Currently, I think doi's do not get used by end-users as much as we would like because they are hardly ever genuinely used as bookmarkable objects.

Keeping these things in mind, something to think about is fixed size handle.

Another system along these lines is the UUID system.

http://en.wikipedia.org/wiki/UUID

> *A UUID is a 16-byte (128-bit) number. The number of theoretically possible UUIDs is therefore $2^{16*8} = 2^{128} = 256^{16}$ or about 3.4×10^{38}. This means that 1 trillion UUIDs would have to be created every nanosecond for 10 billion years to exhaust the number of UUIDs.*
>
> *In its canonical form, a UUID consists of 32 hexadecimal digits, displayed in 5 groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters.*
> *550e8400-e29b-41d4-a716-446655440000*
>
> *A UUID may also be used with a specific identifier intentionally used repeatedly to identify the same thing in different contexts. For example, in Microsoft's Component Object Model, every component must implement the IUnknown interface, which is done by creating a UUID representing IUnknown. In all cases wherever IUnknown is used, whether it is being used by a process trying to access the IUnknown interface in a component, or by a component implementing the IUnknown interface, it is always referenced by the same identifier: 00000000-0000-0000-C000-000000000046.*

Features of this model are:

Local generation, with no central authority, or coordination, with only a vanishingly small chance of collision.

Global uniqueness. Because they do not collide, they can be combined later into a single system without worries of overlap.

OCR scenarios -