The Handle System is a scheme for the persistent naming of objects on the Internet. I will not talk about the handle architecture at all in this presentation, except to say that it is a distributed client server architecture. You can find out about it from the handle website, www.handle.net, beginning with the page, introduction.html. Instead, I will describe our experience implementing and using system.

We in the systems division of the University of Chicago library first became interested in the Handle System in may of 1998. We had been researching technologies for persistent naming as part of our retrospective digitization activities. After our initial experiments with the handle system, we decided to use it in production mode for one such project, funded by the NEH.

In May of 1998, when we first became interested in handle technology, the handle server was not available for downloading. this was because the legal department of CNRI (the Corporation for National Research Initiatives), which designed this implementation of the handle system, had not yet finished what it needed to do to allow CNRI to make it available. In other words, the text for the page at the bottom of which one clicks "I agree," had not yet been cleared by legal. By September, as CNRI had promised, the situation had changed, and i downloaded and installed the handle server. Actually, I installed it twice, the first time just to get a sense of what the install script really did, the second time for real.

I will say a little about that system itself. The core server is written in C. Many of the utilities and the administrative API is implemented in Python. Binaries for a local handle server exist for the following platforms: Solaris 2.6 for Intel, Solaris 2.x for SPARC, and AIX 4.x. other, related software and some nice explanatory diagrams have appeared on the Handle website since we first looked at it; look at download.html at the same url if you're interested.

I need to say that CNRI is rewriting its Handle System implementation in Java, for platform-independence, and incorporates user feedback from sites such as ours. This version should be ready in September. Version 2 will be even more distributed than the current version, and one thing I'll say about this version right now is that the security has been totally rewritten, so that there is now secure authentication using public/private keys. This is a big improvement over version 1.

To use the Handle System, one needs to be registered as a naming

authority. By the time we became interested, CNRI had moved to a numeric only policy for naming authorities. So, for example, the National Library of Medicine is 1000. In the words of Larry Lannom, whom I initially contacted at CNRI about the possibility of obtaining the handle server for local use, the reason for using opaque strings in this manner is in part because "We don't want to get into the DNS-like problems of who is really who and what does that word mean in Spanish anyway. We started off with 7 digit random numbers, but have retreated to four digit sequential."As a result, we got the next available numeric naming authority: 1001.

The version 1 install script is interactive, and prompts you for a number of things I won't mention here. By and large, the script is easy to use, and I have only some minor quibbles with it. For example, it does not give one an opportunity to change the destination of the logfile directory. At our site, we like all our servers to log into one filespace hierarchy, to facilitate log management. Also, the script _is_ interactive. I cannot fault CNRI for this, because the Handle System I downloaded is about as close to being shrinkwrapped, with the exception of a few details I'll get into presently, as such a system can be. So, for some sites, this will be "a good thing." Nevertheless, at our site there is a preference for installations which can be fully automated. Interactive scripts do not facilitate this. However, the install script is written in Python, so if you can hack Python, you can make it do whatever you want it to do, including making it stop being interactive.

I've mentioned logs: let me say a bit more about them. The current system has two logs. One is a kind of state of the server log, useful only to systems administrators. The other records transactions against the administrative database, in other words, what handles were created, and which were edited or deleted. This is useful to a handle administrator. The new version will include as an option a log of actual queries against the server. Version 1 did not have this option, because there was concern that having it would degrade performance. In version 2 it will be an option, in response to user feedback. For example, I might use this feature to track usage of remote resources, which our web server cannot track, for obvious reasons--the client is making the connection directly. Such resources include our own online catalog, Encyclopedia Britannica, etc.

The install script installs a version of the Apache web server and the Python programming language together with the handle server. At our site, we do not like this behaviour, because we already have Apache and Python installed elsewhere. I can't really fault CNRI for

packaging the system this way, but I did tell them that the installation of these supplementary distributions should be optional.

Running the install script, which configures the system according to the values you've been prompted for, does not require one to be root. Installing the system does, but these are two separate steps unless you run the install script as root initially.

Running the install script for the first time creates a server configuration file. You can edit this file manually, then re-run the script once again, before installation. The script will read the configuration if it exists, and the interactive prompts will default to those values. I edited the config file to put the logs in a different place. Though the install script still did not prompt me about logs, it honored this value in the config file when it did the install.

There is one thing that I suggest you be dead careful about if you decide you want to download this implementation of the Handle System and use it. Again, I first downloaded the system last September, and do not know what the version two install will do. So you need to make sure about the following aspect of installation, because it may still work the same way it did last year. That is, if you're not careful, it is possible to totally trash the handles database. This is because, unless you edit the server configuration file to tell the install script not to re-initialize the db, it will. With re-initialization, there will disappear any handles that you might have created. Obviously, with the first install, initialization makes sense, and also perhaps with a second install, if what you've done is fooled around with the system, creating a few handles, editing them, deleting them, but now you want to start afresh, and do it for real. But if, for example, you're re-running the install script only to change port numbers on one of the various components of the system that talk to each other via TCP or UDP, this is _not_ what you want to do. Of course, a good tape backup can bail you out of this situation if you do happen to screw up, so you might want to be sure you've got those going for this system, too.

If you install the handle system on a Solaris box, it is very easy to fire up. /etc/init.d/handle_server start will start it up, and /etc/init.d/handle_server stop will stop it. Handle_Server is a script which in turn fires up three servers. Without going into very much detail, what you have is a server that resolves handles using either UDP or TCP, and an administration server, with which you create handles interactively using a web browser. You can also use the

administration server to load a file of handles in batch mode. You can also install a secondary or mirror handle server. I did not do this.

I found the installation documentation very clear, and the installation process relatively painless.

Neither Netscape nor Internet Explorer come with Handle Support built in. I asked CNRI whether they'd spoken to Netscape or Microsoft about this. They said they had, but it was not high on either company's priority list. This means that to resolve handles, you must use either a browser plug-in or some kind of proxy mechanism.

I have never looked at the browser plug-in option. If we're going to create Handles and expect our users to use them on a production basis, the process has to be transparent to them, otherwise I think it is unacceptable. so I investigated the proxy option.

When we first downloaded the system, we also downloaded a stand-alone web proxy server that could proxy a request for a handle via a url. So, for example, http://www.yoursite.edu/something-or-other/yourNamingAuthority/yourHandle would resolve itself into the actual, non-persistent, URL. From my perspective, though this proxy mechanism was better than nothing, it was sub-optimal. For one thing, I had to dedicate a machine to this, because the server was the old CERN server, which does not support virtual hosting. For another, it was a modified version of the old CERN server, and web servers are, like lots of other servers one is obliged to run, potential security risks. Finally, i did not find this server to be stable. It would run for a few days then die. So what we did was download the handle client library to try to write a cgi-bin script to attach to our own production web server which could proxy handle requests. To do this, we needed a Python binding for the client code, and contacted CNRI. What resulted was that I downloaded the grail web browser from the CNRI site, which contained a Python implementation of the client library. One of my staff, Sarah Burke, who's here today, worked that up into a cgi-bin script, which we put into production. Now when we create handles, and want to embed them in the 856 field of a MARC record, or a web page, we envelope them with the proxy url string: http://www.lib.uchicago.edu/cgi-bin/hdl/. This is rather long, I admit, and one might wonder how can something be persistent if it's wrapped in a url. The best I can say is that it is unlikely that these components, the http scheme, the domain name system, and the cgi-bin convention, will go away so soon as to make these proxied creations a problem. They're certainly more robust than

their alternatives, the actual urls.

We made one other custom modification to the system as distributed. The Handle Administration System allows one to list handles created by one's naming authority, but the information provided this way is really pretty weak. Seeing the handle by itself is useless, I think. What one wants is the handle/url correspondence, if what one is doing is associating with urls--this is probably the most common use of handles, but the system supports other correspondences, though only the url correspondences can be proxied. One also wants one's handle/url correspondences to be live links, for two reasons: one wants to click on them, maybe, and one wants to run a url-checker on them, certainly. We contacted CNRI and they had some code that they used for doing just what we wanted, and gave it to us. Sarah adapted it according to specs I gave her. She cron'd it to run once a day, and now we have a very nice web page with our handle/url correspondences for both our naming and sub-naming authority--I'll get to that in a moment. Of course, this approach will not scale for pages and pages of handles, but it's good enough for the moment, and better than what we had at the start.

I just mentioned sub-naming authority. After I had experimented with handles for a while, I decided I had enough experience with the system to instruct a cataloger in its use for the NEH-funded project I mentioned earlier. A distinguishing feature of this project is the urls of the digital objects created for it: they're long, hard to type, hard to transcribe, and hard to communicate. They did not survive intact between the University of Chicago library and OCLC. This is because though marc is a very robust communication standard, no two implementations of the MARC standard are exactly alike, and particularly not in the more obscure portions of character sets. Since the digital library objects in question were created by a programmer, and since he, like many, though by no means all, programmers, regards an underscore as a perfectly normal character, he used them in lieu of spaces; the naming convention for these objects is their LC class number, which, of course, can include spaces.

I contacted CNRI to register a sub-naming authority. Unless one wants techical support, this is the only reason to contact CNRI once one's original naming authority is granted. we decided to call our sub-naming authority 1001.cat, for the cataloging department. Creating a sub-naming authority allows one to delegate the creation of handles for production projects in such a way that the sub-naming authority administrator cannot inadvertently trash any handles one has created as part of the top-level authority, because you're working in separate

name spaces with separate permissions. However, the namespaces are hierarchically arranged from the permissions point of view: one gets to add and remove administrators for the sub-naming authority, and set their permissions. In general, there is a good mix of both flexibility and control in the way this is set up.

In general, I found the handle system both generally smooth to set up and administer, and generally well thought out. This is consistent with CNRI's website and the way their code is organized. The experience was largely a good and pleasant one. The problems I encountered were generally either the result of my not spending enough time getting to understand the system, because I was too busy, or else consisted of very picayune details. For example, the administrative web server, when one first fires it up and tries to connect to it with a browser, does not take one to the top-level page. I had to create a symlink by hand to facilitate this. I can think of only one maybe good reason CNRI did it this way, and a very good reason why they shouldn't have, but all in all it's a relatively minor point. It's not the sort of thing to mar a generally good impression. Because the system is stateless, with version 1 you have to type your password each time you change administrative functions. Version 2, as I said, has rewritten security. Typing your password twice is fine if you're going to create a bunch of handles interactively as part of a regular workflow--in this case you're prompted to create the next handle, and you can keep on going like that for as long as you want, but not so fine if you need to get back to the main menu to select another administrative task to do. Then this double-typing is a hassle. CNRI itself was not happy with this interface, and really wanted feedback on it. I don't know what it will look like in version 2, but these infelicities aside, it's certainly usable, even in its current form.

As an early adopter, I did find one bug in the system. At the time, I didn't know it was a bug, just a problem I was having, but CNRI identified my problem as the result of a major bug in the handle administration daemon code, not the handle resolution server itself, which it promptly fixed. That was the only bug of which I'm aware.

One other feature I'll mention is that in version 1, passwords between the browser and the handle administration system are transmitted in plaintext. They're also stored in plaintext on disk, in two places: the configuration file, and the transaction log. It is possible to protect the directories in which these files exist so that the files in question are not generally viewable, and at our site the problem is largely moot, because the handle server is running on a machine to which only digital library systems administrators and programmers have

login access, and on a relatively secure subnet. But it is clearly a security problem, and one which I brought to CNRI's attention.

There is online documentation for the administrative web server. Most important of all, I found CNRI very, very helpful in answering questions. The technical support, although it is informal and ad hoc, is nevertheless excellent. Jane Euler, with whom I had all but the initial e-mail exchange, is knowledgeable, prompt, thorough, and patient. In fact, I've kept her e-mail as ancillary documentation for the current system.

When CNRI asked me whether I wanted to beta-test version 2, I did not hesitate to jump at the opportunity. This isn't because I think version 1 was bad. Quite the contrary. In fact, I'm somewhat anxious to see whether version 2 is at least as easy to install and use, and whether it works as well as version 1. Major upgrades like these can go either way, of course. If version 2 is at least as good as version 1, then I'll be able to recommend that implementation of the handle system without reservation.

Example handle implementations:

hdl.handle.net [input handle into web form]
dx.doi.org [input handle into web form]
http://www.lib.uchicago.edu/cgi-bin/hdl/[add handle here]

handle examples (try them with any of the above implementations):

1001/0
4263537/4013