HANDLE.NET (version 7.0)

Technical Manual
Version 1.1

Corporation for National Research Initiatives
December 2010
hdl:4263537/5043

Credits

CNRI wishes to thank the prototyping team of the Los Alamos National Laboratory Research Library for their collaboration in the deployment and testing of a very large Handle System implementation, leading to new designs for high performance handle administration, and handle users at the Max Planck Institute for Psycholinguistics and Lund University Libraries NetLab for their instructions for using PostgreSQL as custom handle storage.

Version Note

Handle System Version 7.0 constitutes a major upgrade to the Handle System. The major improvements include template handles, offline signatures, multiple location type, a DNS interface, Berkeley DB as the default storage system, and a Jython interpreter that can run script files or show an interactive shell, enabling any Python programmer to create scripts to interact with the handle server. (See the Release Notes and Change Log for details.)

Not all of these upgrades are fully documented in this version of the Handle Technical Manual. The manual is being updated and a more completed version is expected in the near future.

Please send questions or comments to the Handle System Administrator at hdladmin@cnri.reston.va.us.

Table of Contents

1 Introduction

The Handle System is a comprehensive system for assigning, managing, and resolving persistent identifiers for digital objects and other resources on the Internet. The Handle System includes an open set of protocols, a namespace, and an implementation of the protocols. The protocols enable a distributed computer system to store identifiers of digital resources and resolve those identifiers into the information necessary to locate and access the resources. This associated information can be changed as needed to reflect the current state of the identified resource without changing the identifier, thus allowing the name of the item to persist over changes of location and other state information.

1.1 Handle Syntax

Within the handle namespace, every identifier consists of two parts: its prefix, and a unique local name under the prefix known as its suffix. The prefix and suffix are separated by the ASCII character "/". A handle may thus be defined as

<Handle> ::= <Prefix> "/" <Handle Local Name>

For example, handle "12345/hdl1" is defined under the Handle Prefix "12345", and its unique local name is "hdl1".

Handles may consist of any printable characters from the Universal Character Set, two-octet form (UCS-2) of ISO/IEC 10646, which is the exact character set defined by Unicode v2.0. The UCS-2 character set encompasses most characters used in every major language written today. To allow compatibility with most of the existing systems and prevent ambiguity among different encoding, handle protocol mandates UTF-8 to be the only encoding used for handles. The UTF-8 encoding preserves any ASCII encoded names, which allows maximum compatibility to existing systems without causing naming conflict.

By default, handles are case sensitive. However, any handle service, including the Global Handle Registry® (GHR), may define its namespace such that all ASCII characters within any identifier are case insensitive.

The handle namespace can be considered as a superset of many local namespaces, with each local namespace having its own unique handle prefix. The prefix identifies the administrative unit of creation, although not necessarily continuing administration, of the associated handles. Each prefix is guaranteed to be globally unique within the Handle System. Any existing local namespace can join the global handle namespace by obtaining a unique prefix, with the resulting identifiers being a combination of prefix and local name as shown above.

Each prefix may have many derived prefixes registered underneath. Any derived prefix can only be registered by its parent after its parent prefix is registered. Every handle is then defined under a prefix. The collection of local names under a prefix is the local namespace for that prefix. Any local name must be unique under its local namespace. The uniqueness of a prefix and a local name under that prefix ensures that any identifier is globally unique within the context of the Handle System.

1.2 Handle System Scalability

Scalability was a critical design criterion for the Handle System. The problem can be divided into storage and performance. That is, is there some limit to the number of identifiers that can be added? And, does performance go down, or do some functions simply break with increased numbers of identifiers, such that at some point the system becomes unusable? Specific details on this are given below, but it is important to keep two higher level issues in mind. First, it is important to distinguish between Handle System design and any given implementation. Scalability in design may or may not work out as expected in any given implementation, but if the design is fundamentally scalable, specific implementation problems can be corrected as they are encountered. Secondly, use of the Handle System through some other service, e.g., an http proxy, may introduce other scalability issues which the basic Handle System design does not and cannot address.

1.2.1 Storage

The Handle System has been designed at a very basic level as a distributed system; that is, it will run across as many computers as are required to provide the desired functionality.

Handles are held in and resolved by handle servers and the handle servers are grouped into one or more handle sites within each handle service. There are no design limits on the total number of handle services which constitute the Handle System, there are no design limits on the number of sites which make up each service, and there are no limits on the number of servers which make up each site. Replication by site, within a handle service, does not require that each site contain the same number of servers; that is, while each site will have the same replicated set of handles, each site may allocate that set of handles across a different number of servers. Thus, increased numbers of handles within a site can be accommodated by adding additional servers, either on the same or additional computers, additional sites can be added to a handle service at any time, and additional handle services can be created. Every service must be registered with the Global Handle Registry, but that handle service can also have as many sites with as many servers as needed. The result is that the number of identifiers that can be accommodated in the current Handle System is limited only by the number of computers available.

1.2.2 Performance

Constant performance across increasing numbers of identifiers is addressed by hashing, replication, and caching.

Hashing, a technique well known to database designers, is used in the Handle System to evenly allocate any number of identifiers across any number of servers within a site, and allows a single computation to determine on which server within a set of servers a given handle is located, regardless of the number of handles or the number of servers. Each server within a site is responsible for a subset of handles managed by that site. Given a specific identifier and knowledge of the handle service responsible for that identifier, a handle client selects a site within that handle service and performs a single computation on the handle to determine which server within the site contains the handle. The result of the computation becomes a pointer into a hash table, which is unique to each handle site and can be thought of as a map of the given site, mapping which handles belong to which servers. The computation is independent of the number of servers and handles, and it will not take a client any longer to locate and query the correct server for a handle within a handle service that contains billions of handles and hundreds of servers, than for a handle service that contains only millions of handles and only a few servers.

The connection between a given handle and the responsible handle service is determined by prefix. Prefix records are maintained by the GHR as handles, and these handles are hashed across the GHR sites in the same way that all other handles are hashed across their respective handle services. The only hierarchy in Handle System services is the two level distinction between the single GHR and all locals, which means that the worst case resolution would be that a client with no built in or cached knowledge would have to consult the GHR and one local.

Another aspect of Handle System scalability is replication. The individual handle services within the Handle System each consist of one or more handle service sites, where each site replicates the complete individual handle service, at least for the purposes of handle resolution. Thus, increased demand on a given handle service can be met with additional sites, and increased demand on a given site can be met with additional servers. This also opens up the option, so far not implemented by any existing clients, of optimizing resolution performance by selecting the "best" server from a group of replicated servers.

Caching may also be used to improve performance and reduce the possibility of bottleneck situations in the Handle System, as is the case in many distributed systems. The Handle System data model and protocol design includes a space for cache time-outs and handle caching servers have been developed and are in use. (See RFC 3651, Handle System Namespace and Service Definition, Section 4.2, Handle System Middleware Components, hdl:4263537/4068.)

1.3 Authentication

The current distribution of the HANDLE.NET software uses the Sun Java™ Cryptography Extension and Provider for low level cryptography routines. This library comes standard with Java™ version 5.

1.3.1 Types of Authentication

As further described in *System Fundamentals:* Authentication on the Handle System web site, the Handle System provides two forms of authentication: public key and secret key.

In the current implementation, public key authentication is performed using the DSA algorithm. The key length is variable from 512 to 1024 bits, and can be chosen by the user when generating keys. The HANDLE.NET software distribution defaults to a 1024 bit key.

Public key authentication requires two keys: a public key and a private key. The public key is stored in a handle to make it available to the public. The private key should be securely stored on the computer with the handle client that will be authenticated. To prevent unauthorized use of a private key, it can be encrypted using a symmetric algorithm. The current implementation of the Handle System uses 56 bit DES for this purpose.

Secret key authentication relies on a secure hashing algorithm, chosen by the client being authenticated. Currently, this algorithm can be either MD5 or SHA1. A secret key consists of a single byte string of size ranging from 0 to 2147483648. This byte string is stored as plain text in a handle. It is highly advisable to restrict read permissions on the handle to ensure the secrecy of the secret key.

1.3.2 Certification

Clients can request that a handle server cryptographically certify its messages with its public key. This certification can be used to verify the authenticity of handle server transmissions. The current implementation of the Handle System uses DSA for this purpose. The DSA public key for a handle server is stored in its site information record.

1.3.3 Sessions

Establishing sessions with a handle server offers additional security functionality. For background on handle server sessions, see *System Fundamentals:* Sessions on the Handle System web site.

The Handle System allows for encryption of communication after establishing a session with a handle server. This is equivalent to SSL or TLS as used in protocols such as HTTPS, as it affords protection from eavesdropping and man-in-the-middle attacks. The current implementation of the Handle System encrypts session communications using 56bit DES.

For instructions on enabling session encryption see the Handle Tool User Manual and Chapter 3, *Batch Operation*.

1.3.4 Algorithms

More information on the algorithms mentioned above can be found at the locations below.

DSA: http://www.itl.nist.gov/fipspubs/fip186.htm

DES: http://www.itl.nist.gov/fipspubs/fip462.htm

2 Installing and Running a Handle Server

This section outlines the steps required to get the HANDLE.NET software, install it, set it up, and acquire a prefix.

The distribution includes a GUI tool developed by CNRI for performing handle operations such as creating and deleting handles, authenticating, and setting up a handle server. The Handle Tool has been found to be useful for quickly creating and updating handle URL values that are used with the World Wide Web.See also the Handle Tool User Manual.

2.1 Installing Java™

The HANDLE.NET software requires Java™ to run. If you already have Java™ installed, confirm that you have version 5. You can check this by running the command 'java –version'.

If you need to install Java™, you can obtain a version for several popular platforms at http://java.sun.com/javase/. For other platforms, contact your vendor.

2.2 Unpacking the Distribution

Download the HANDLE.NET software distribution from http://www.handle.net. To uncompress the distribution package on Unix-like platforms use the gunzip and tar programs. The distribution package can be uncompressed on Win32 based systems using the program WinZip©.

The distribution will uncompress to have subdirectories bin, doc, lib, as well as the zipped source in src.zip. The examples in this section will assume that the distribution was unpacked into /hs/hsj-7.0.0. All of the executable/batch files necessary to setup the server are in the directory /hs/hsj-7.0.0/bin.

2.3 Choosing an Installation Directory

Create a folder for the server configuration. We will assume this folder is called "/hs/srv_1". Be sure to create a new directory for each server on the same machine. To create this directory on Unix, run this command:

```
mkdir -p /hs/svr_1
```

For more information on the files contained in the installation directory, see Section 2.7, *Installation Directory*.

2.4 Running the Setup Program

The HANDLE.NET software includes an installation program. The program requires Java™ , so make sure you have the java binary directory in your system path. Navigate to the "hsj-7.0.0" directory and execute the following commands:

On Unix-like systems it can be invoked by running the executable: bin/hdl-setup-server /hs/srv_1
On Windows, DOS, or OS/2 like systems: bin\hdl-setup-server.bat \hs\svr_1

The installation program guides you through a series of configuration options. Once complete, there will be a file called 'sitebndl.zip' in your handle server directory which you will email to the Handle System Administrator at hdladmin@cnri.reston.va.us. The Administrator will use the 'sitebndl.zip' file to create a prefix in the root service (GHR), and will notify you when this has been completed. You will not be able to continue the install until you receive information from the Handle System Administrator concerning your prefix.

Please note: A Handle System Service Agreement is required in order to provide identifier/ resolution services. (Please see* http://hdl.handle.net/4263537/5029 *for more information.) You will not receive a prefix from the Handle System Administrator until you have agreed to the terms of the Service Agreement. Once you have agreed to the Service Agreement, paid the appropriate fees, and received prefix information from the Handle System Administrator, you may proceed with the following steps to 'Home' your prefix to your new handle service.

2.4.1 Setup Tool

Below is an example of the output generated during a Setup session, showing the configuration options. The file sitebndl.zip, generated by running the tool, is used to create a prefix in the GHR.

```
To configure your new handle server, please answer the questions which follow; default
answers, shown in [square brackets] when available, can be chosen by pressing Enter.

Will this be a regular handle server or caching server?
 1 - Regular Handle Server (recommended)
 2 - Caching Handle Server

Please choose 1 or 2 and press Enter [1]: 1

Will this be a "primary" server (i.e., not a mirror of another server)?(y/n) [y]
Through what network-accessible IP address should clients connect to this server?
[<your_IP>]:

If different, enter the IP address to which the server should bind.  [<your_IP>]:
Enter the (TCP/UDP) port number this server will listen to [2641]:

What port number will the HTTP interface be listening to? [8000]:

Would you like to log all accesses to this server?(y/n) [n]:

Please indicate whether log files should be automatically rotated, and if so, how
often.

("N" (Never), "M" (Monthly), "W" (Weekly), or "D" (Daily))? [Never] : M

NOTE: Auto-saves and restarts will be done on the first of each month.

Each handle site has a version/serial number assigned to it. This is so that a client
can tell if a particular site's configuration has changed since the last time it
```

```
accessed a server in the site. Every time you modify a site (by changing an IP
address, port, or adding a server, etc), you should increment the version/serial
number for that site.

Enter the version/serial number of this site [1]:

Please enter a short description of this server/site: <Description HERE.>

Please enter the name of your organization: <ORG NAME>

Please enter the name of a contact person for ORG NAME (optional) [(none)]: <CONTACT
NAME>

Please enter the telephone number of CONTACT NAME or of  ORG NAME (optional) [(none)]

Please enter the email address of CONTACT NAME or of ORG NAME: <contact_name@org_name>
The Handle System can communicate via UDP and/or TCP sockets. Since UDP messages are
blocked by many network firewalls, you may want to disable UDP services if you are
behind such a firewall.

Would you like to disable UDP services?(y/n) [n]:

Generating keys for: Server Certification

The private key that is about to be generated should be stored in an encrypted form on
your computer.  Encryption of the private key requires that you choose a secret
passphrase that will need to be entered whenever the server is started. Your private
key may be stored unencrypted. Please take all precautions to make sure that only
authorized users can read your private key.

Would you like to encrypt your private key?(y/n) [y]: y

Please enter the private key passphrase for Server Certification:

Note: Your passphrase will be displayed as it is entered

oneexample

Please re-enter the private key passphrase:

Note: Your passphrase will be displayed as it is entered

oneexample

Generating keys for: Administration

The private key that is about to be generated should be stored in an encrypted form on
your computer.  Encryption of the private key requires that you choose a secret
passphrase that will need to be entered whenever the server is started. Your private
```

```
key may be stored unencrypted. Please take all precautions to make sure that only
authorized users can read your private key.

Would you like to encrypt your private key?(y/n) [y]:
Please enter the private key passphrase for Administration:
Note: Your passphrase will be displayed as it is entered

anotherexample

Please re-enter the private key passphrase:

Note: Your passphrase will be displayed as it is entered

anotherexample

Generating site info record...

-------------------------------------------------------

You have finished configuring your regular (primary) handle service. This service now
needs to be registered in the Global Handle Registry(GHS).

Please go to http//hdl.handle.net/4236567/5014 to upload your newly created
sitebndl.zip file. Please carefully read the directions on this page.  When the handle
administrator receives your file, a prefix will be created and you will  receive
notification via email.

Please send all questions to hdladmin@cnri.reston.va.us.  Thank you for your interest
in CNRI's Handle System.
```

2.5 Running the Handle Server for the First Time

Go to your 'svr_1' directory (where you installed your HANDLE.NET software) and edit the 'config.dct' file. Replace the YOUR_NAMING_AUTHORITY under server_admins and replication_admins with your prefix (as indicated in your email message from hdladmin). This allows anyone with the private key that matches your public key to be an administrator for your server.

Start the server using the configuration just generated.

On Unix-like systems: bin/hdl-server /hs/srv_1
On Windows-like systems: bin\hdl-server.bat /hs/srv_1

Note: If you chose to encrypt your private key(s), you will be prompted for your passphrase here. Also note that Java™ does not have the ability to disconnect from a terminal so you will have to put the process in the background. On Unix systems type Ctrl Z, then bg, then press ENTER.

Start the Handle Tool using the following command:

On Unix-like systems: bin/hdl-admintool
On Windows: bin\hdl-admintool.bat

Click on the 'Authenticate' button. You will be prompted for your authentication information.

The 'Your ID' will be 0.NA/YourPrefix.
The blank field to the right is the Index and should be 300.
The 'Key Type' should be Public/Private Key.

Browse to find your private key file. It will be in the "svr_1" directory (where you installed your server) and is named "admpriv.bin". Click "OK". You will be prompted for your secret passphrase. This is the password you entered for Administration when you ran the setup program.

Next, "Home" your prefix. (See the Handle Tool User Manual, Chapter 11, "Homing/Unhoming a Prefix".) Select "Tools>Home/Unhome Prefix". This example assumes you were given handle "0.NA/YourPrefix".

"Prefix" field should contain: YourPrefix
Select the "Home Prefix" radio button.
Select "By Site Info File" and locate your "siteinfo.bin" file in "svr_1"
OR
Select "By Address" and enter the "Server Address" and "Server Port"
Click "Do It"

Please note that the Handle System does not use DNS.

2.6 How Your Prefix Was Set Up

This section describes how your prefix was set up and how authentication and permissions can be configured.

The prefix 12345 would have the following properties:

Prefix Handle: 0.NA/12345
Prefix Admin Group: 0.NA/12345 index 200
Prefix Public Key: 0.NA/12345 index 300

When authenticating, identify yourself using the Prefix Admin Public Key and the associated private key which is in your 'admpriv.bin' file on your computer. The instructions are in the 'INSTALL.txt' file (see Section 2.5, *Running the HANDLE.NET Server For the First Time*. This is what you used to authenticate yourself to "home" your prefix.

When creating new identifiers, specify an administrator who will have permission to modify or delete each new identifier (handle) by adding an Admin value that references a public key, secret key or admin group to each new handle. We recommend that you specify your Prefix Admin Group (see above) value as the administrator for each new handle.

Every value in a handle has a different index. The following pattern works well. Start with 100 for all admin values, start admin group values at 200 and make the public key index 300. So the values of a handle (12345/hdl1), might look like this:

```
100 HS_ADMIN 0.NA/12345 index 200
3 URL http://www.someorg.com/info
4 email someone@someorg.com
```

2.7 Installation Directory

The installation directory contains a number of files. This section explains the function of each.

2.7.1 logs/access.log

If you enabled 'log accesses' on any of your handle server interfaces, all requests sent to those interfaces are logged here. Below is a sample line from this file.

```
10.27.4.211 TCP:HDL "2001-12-18 18:48:33.589-0500" 101 1 111ms 200/8
```

The first column is the IP address of the host that made the request. The second column shows the interface the request was made on. Next is the date and time the request was made. The time is followed by the Handle Operation Requested Code (OP) of the request. In this case the OP is 101, or delete handle. The OP is followed by the Handle Server Response Code (RC). In this case, the RC is 1, or success. Here is a list of possible OPs and RCs:

Handle Operation Requested Code (OP)		**Handle Server Response Code (RC)**	
1	Query Handle	1	Success
100	Create Handle	2	Error
101	Delete Handle	3	Server Too Busy
102	Add Value	4	Protocol Error
103	Remove Value	5	Operation Not Supported
104	Modify Value	6	Recursion Count Too High
105	List Handles	7	Authentication Error
200	Challenge Response	100	Handle Not Found
201	Verify Response	101	Handle Already Exists
400	Session Setup	102	Invalid Handle
401	Session Terminate	200	Values Not Found
402	Session Exchange Key	201	Value Already Exists
1003	Backup Server	300	Out of Date Site Info
		301	Server Not Responsible
		302	Service Referral
		303	Server Backup

Handle Operation Requested Code (OP)	**Handle Server Response Code (RC)**	
	400	Invalid Admin
	401	Insufficient Permissions
	402	Authentication Needed
	403	Authentication Failed
	404	Invalid Credential
	405	Authentication Timed Out
	406	Authentication Error
	500	Session Timeout
	501	Session Failed
	502	Invalid Session Key
	504	Invalid Session Setup Request

After the rc code, the log entry shows the number of milliseconds the server took to respond to the request. This is useful for gauging the performance of a handle server. The final column of the log entry indicates the handle that was requested (if applicable).

2.7.2 logs/error.log

As suggested by its name, this file contains a log of server errors.

2.7.3 config.dct

This is the server configuration file. See Chapter 4 *Advanced Server Configuration*, for more information.

2.7.4 $HOME/.handle/root info

The 'root_info' file contains the 'HS SITE' records for the global handle servers, necessary for handle resolution. This file isn't actually stored in the handle server directory, but in the subdirectory '.handle' under the home directory of whatever user runs the handle server.

2.7.5 cache.jdb

This is a database used by the server as a temporary handle cache.

2.7.6 bdbje/, handles.jdb, nas.jdb

By default the handle server uses a Berkeley DB Java Edition database to store handles and handle values as well as prefixes that are homed to the server. This database is located in the bdbje subdirectory. The previous storage default in earlier versions was to store this information in two files handles.jdb and nas.jdb.

2.7.7 dbtxns.log

This is a log of database transactions, and it is used for disaster recovery. It may grow large in size. It may be deleted at any point to free up disk space, but once it's gone, there will be no way to recover if the handle database becomes corrupted.

2.7.8 siteinfo.bin

This contains the 'HS_SITE' record for this server. It is stored in the handle prefixes that this server is responsible for.

2.7.9 admpub.bin, admpriv.bin

These are the public and private keys that were created for the administrator during the installation process.

2.7.10 pubkey.bin, privkey.bin

These are the public and private keys for the server. The public key is stored in the servers 'HS_SITE' entry. The private key is used to sign responses to requests.

2.7.11 txns/

This directory stores the server transaction queue. This keeps track of handle administration operations in order to replicate to secondary servers. The transaction queue is separated into separate files so that it will be easy to wipe out old transactions. The old files may be deleted in order to free up disk space.

Individual records within each transaction log look something like this:

```
<CRLF><txnID>|<action>|<date>|<hashOnAll>|<hashOnNA>| <hashOnId>|<handlehexencoded>|
```

2.8 Splitting a Handle Server

This section describes how to split a single handle server into multiple handle servers. This does not describe how to split sites consisting of multiple servers, which is a much more complicated process. Contact hdladmin@cnri.reston.va.us for assistance.

The goal of this procedure is to minimize the down time for the primary site. The database splitting process can be performed on a checkpoint/backup of the source database, and then completed using the transaction log of the source database.

To split a single server site:

1. Create the directories and configuration files for the new servers. Do this by running the hdl-setup-server program to create a new directory, config file, and 'siteinfo.bin' for each new server.
2. Combine the 'siteinfo.bin' files for each of the new servers into one HS_SITE record using the Handle Admin tool:

i. Start the Handle Admin tool.

ii. Click the 'Create Handle' button to open a create-handle window.

iii. Click the 'Add Custom Data' button.

iv. Select the 'HS_SITE' value type, and click the 'Value Data' button.

v. Set the protocol for the new site info to 2 and 1.

vi. Set the serial # for the new site to something greater than the serial number of the existing server.

vii. Enter the information for each new server (IP address, the new server's public key and ports) into the site-info window (this information is in the server directory) by clicking the 'Add' button in the 'Servers' section of the site-info window.

viii. Set the ServerID for each server to a unique number. The number will be specified in the 'config.dct' file for each server, so that the server knows which server it is, relative to the other servers in the 'siteinfo.bin' file.

Save the newly created site-info value to a new 'siteinfo.bin' file. Copy this new file into the installation directory for each of the new servers. Add this_server_id setting to the server_config section of each new server's config.dct file.

3. Run a checkpoint/backup on the server that is to be split. Then wait until the checkpoint/backup operation is complete. The old primary server should still be running at this point.

4. Decrease the timeout/TTL value for the 'HS_SITE' value that is being modified in the prefix or service handle. A good setting is probably 1200 (20 minutes). Do this so that when you change the 'HS_SITE' value, clients don't still try to access the old server for an entire day after the 'HS_SITE' value is changed.

5. Run the hdl-splitserver program to split the 'handles.bak' file into the new server directories. Make sure to specify the new server directories in the correct order on the command line. For example:

```
bin/hdl-splitserver
/usr/local/current_hs/handles.bak /usr/local/new_hs1
 /usr/local/new_hs2 /usr/local/new_hs3
```

On Windows:

```
bin\hdl-splitserver.bat
 \usr\local\current_hs\handles.bak \usr\local\new_hs1
 \usr\local\new_hs2 \usr\local\new_hs3
```

This phase can take a long time when splitting large databases. If the database is large enough the process may even take several days. The old server should still be running at this point.

6. Run the hdl-splitrecoverylog program to process the 'dbtxns.log' file to scan all of the handle modifications, creations, and deletions that have taken place since the backup/checkpoint operation. When this is finished, the date of the last transaction processed will be printed. Record this for future use.

7. *Only perform this step if you are splitting a primary server.* Shut down the old handle server. Copy the 'txn_id' file from the old server to each of the new server directories. This file will only be used by one of the new servers, but it is difficult to tell which one so we copy it to each server directory just to be safe.

8. Run the hdl-splitrecoverylog program again, this time providing the date of the last transaction processed in Step 6 on the command line. (Run the hdl-splitrecoverylog program with no arguments to see the syntax of the command.) This step will quickly bring the new databases into sync with the single primary by processing the transactions that have occurred on the primary since Step 6.

9. Start the new servers and make sure they come up without any problems.

10. Update the 'HS_SITE' value of the prefix or service handle with the new 'siteinfo.bin' file created in Step 2.

11. Wait at least 20 minutes so that the old 'HS_SITE' value is timed out in the caches of all possible handle clients/administrators.

12. *Only perform this step if you are splitting a primary server.* Copy the new 'siteinfo.bin' file into the directory of the old handle server. Start the old handle server back up. No administration requests should arrive. Performing this step should cause any secondary servers to notice the siteinfo version number change and retrieve the new siteinfo record from the old primary.

After some time has passed and everything is working, change the timeout/TTL of the modified HS_SITE value back to 86400 (one day).

2.9 Shutting Down a Handle Server

To stop a handle server, use the kill command for Unix systems. Find the process id and then 'kill process_id'. Then start the server using the command:

```
hdl-server your_svr_dir
```

Please notify CNRI via email to hdladmin@cnri.reston.va.us if you plan to shut down your server permanently.

2.10 Inactive Prefixes

The Handle System Service Agreement requires Resolution Service Providers to ensure that the identifiers they create will resolve. If a handle service is shut down, the Handle System Administrator must be notified in advance, and arrangements must be made to enable clients to correctly inform users of the status of those handles.

In future, to help ensure proper resolution of such identifiers, the handle type HS_NAMESPACE is being proposed for use in alerting handle clients that a prefix is no longer supported or being used and handles created under that prefix have been removed. The Handle System Administrator would add an HS_NAMESPACE value type (a simple XML structure) to an inactive prefix that includes status, contact information, and a customized message :

```
<namespace>
<contact>yourName@yourOrg.org</contact>
<status>inactive</status>
<statusmsg>This prefix has been deactivated by the administrator as of September 2006.</statusmsg>
</namespace>
```

Namespace-aware native handle clients (including the proxy server), that attempt to resolve handles under a prefix with an HS_NAMESPACE value indicating the prefix is inactive, will return a "Handle Not Found" error message including the contact and statusmsg data (if provided) stating that the prefix is no longer being maintained and all handles beginning with the prefix have been removed.

3 Batch Operation – Command Line

It may be desirable to perform more handle operations than it is possible to perform using either of the administration tools. In those cases it is best to use the batch facilities included with the HANDLE.NET software distribution.

Submit batches using the 'GenericBatch' command line utility, which can be invoked using the following command:

```
bin/hdl-genericbatch <batchfile> [<LogFile>]
```

On Windows: bin\hdl-genericbatch.bat <batchfile> [<LogFile>]

All batch files are plain text format. One batch file can have more than one handle operation. The handle operations are: Create Handle, Delete Handle, Home/Unhome Prefix, Add Handle Value, Remove Handle Value, Modify Handle Value, Authenticate User, Setup Session.

If you need to change authentication for subsequent batch operations, the new authentication information should be put before the batch block. If you authenticate during the batch submission, then you need not include the authentication information in the batch file.

3.1 Create Handle Batch Format

Operation name is 'CREATE'. The first line is composed of the following:

```
CREATE + space + handle_name
```

The next lines are handle value lines. (See Section 3.9, *Handle Value Line Format*.) There must be a handle value line to define the administrator of the handle. End the 'CREATE' handle operation with a blank line.

The list of predefined handle value types is as follows: HS_ADMIN, HS_VLIST, HS_SECKEY, HS_PUBKEY, HS_SITE, HS_SERV, HS_ALIAS, EMAIL, URL, URN, INET_HOST, 10320/LOC. Each handle value line must start with a unique index number, followed by the handle value type from the list above, ttl (the time to live in seconds), the permission set (admin read, admin write, public read, public write), and the value data. See Section 4.9, *Handle Value Line Format* for more detail.

Example:

```
CREATE 12345/hdl1
100 HS_ADMIN 86400 1110 ADMIN
300:111111111111:12345/hdl1
300 HS_SECKEY 86400 1100 UTF8 my_password
3 URL 86400 1110 UTF8 http://www.handle.net

 CREATE 12345/hdl2
100 HS_ADMIN 86400 1110 ADMIN 200:111111111111:0.NA/12345
```

```
3 URL 86400 1110 UTF8 http://www.yourorg.org
```

3.2 Delete Handle Batch Format

Operation name is 'DELETE'. This operation deletes an existing handle completely. Every record is a line with:

DELETE + space + handle_name

Example:

```
DELETE 12345/hdl1
DELETE 12345/hdl2
```

3.3 (Un)Home Prefix Batch Format

Operation name is 'HOME' or 'UNHOME'. This operation associates a prefix with a handle server. It only works on existing prefixes and active handle servers. It tells the server which prefixes will be homed or unhomed to it. The first line provides the service information:

HOME/UNHOME + space + server_ip:server_port:protocol(tcp,udp,http)

The next lines give the prefix names which will be homed/unhomed at this server.

Examples:

```
HOME 10.27.10.28:2641:TCP
0.NA/12345

UNHOME 10.27.10.28:2641:TCP
0.NA/12345
0.NA/TEST1.t1
```

3.4 Add Handle Value Batch Format

Operation name is 'ADD'. This operation adds new handle values to an existing handle. The first line is composed of the following:

ADD + space + handle_name

The next lines are handle value lines. (See Section 3.9, *Handle Value Line Format*.) There must be a handle value line to define the administrator of the handle. End the CREATE handle operation with a blank line. The list of predefined handle value , is as follows: HS_ADMIN, HS_VLIST, HS SECKEY, HS PUBKEY, HS SITE, HS SERV, HS ALIAS, EMAIL, URL, URN, INET HOST, 10320/LOC. Each handle value line must start with a unique index number, followed by the handle value type from the list above, ttl (the time to live in seconds), the permission set (admin read, admin write, public read, public write), and the value data.

```
ADD 12345/hdl1
5 URL 86400 1110 UTF8 http://www.handle.net/admin.html
6 EMAIL 86400 1110 UTF8 hdladmin@cnri.reston.va.us

ADD 12345/hdl2
5 URL 86400 1110 UTF8 http://www.cnri.reston.va.us
6 EMAIL 8600 1110 UTF8 hdladmin@cnri.reston.va.us
```

3.5 Remove Handle Value Batch Format

Operation name is 'REMOVE'. This operation removes one or more handle values from an existing handle. Every record is a line with:

```
REMOVE + space + indexes:handle_name
```

Each index is separated by ','

Example:

```
REMOVE 5:12345/hdl1
REMOVE 5,6:12345/hdl2
```

3.6 Modify Handle Batch Format

Operation name is 'MODIFY'. This operation changes one or more handle values for an existing handle. The first line is composed of the following:

```
MODIFY + space + handle_name
```

The next lines are handle value lines. (See Section 3.9, *Handle Value Line Format*.) There must be a handle value line to define the administrator of the handle. End the CREATE handle operation with a predefined. The list of predefined handle value types is as follows: HS_ADMIN, HS_VLIST, HS_SECKEY, HS_PUBKEY, HS_SITE, HS_SERV, HS_ALIAS, EMAIL, URL, URN, INET_HOST, 10320/LOC. Each handle value line must start with a unique index number, followed by the handle value type from the list above, ttl (the time to live in seconds), the permission set (admin read, admin write, public read, public write), and the value data.

Example:

```
MODIFY 12345/hdl1
2 URL 86400 1110 UTF8 http://www.handle.net/newadmin.html
```

```
3 EMAIL 86400 1110 UTF8 hdladmin@cnri.reston.va.us

MODIFY 12345/hdl2
2 URL 86400 1110 UTF8 http://www.cnn.com/newentainment.html
3 URL 86400 1100 UTF8 http://www.cnn.com/newshow.html
```

3.7 Authentication Information Format

Operation name is 'AUTHENTICATE'. For secret key authentication:

First line: AUTHENTICATE+space+SECKEY:admin_index:admin_handle
Second line: Password

Example:

```
AUTHENTICATE SECKEY:301:0.NA/12345
my_password
```

For private key authentication:

First line: AUTHENTICATE PUBKEY:admin_index:admin_handle

Second line: If your private key was created and encrypted by passphrase, then:
    private_key_file_path + '|' + passphrase
Otherwise:
    private_key_file_path

Example:

```
AUTHENTICATE PUBKEY:300:0.NA/12345
c:\home\keyfile|my_pass_phrase
AUTHENTICATE PUBKEY:300:0.NA/12345
c:\home\keyfile
```

3.8 Session Setup Information Format

Operation name is 'SESSIONSETUP'. The 'USESESSION' flag is mandatory. Remaining fields are used to specify optional public key pair information, session attributes (e.g., "Encrypted", "Authenticated"), "If session fails, use challenge response" flag and "Timeout".

The first line is composed of the following:

```
SESSIONSETUP
```

Use the following lines to specify mandatory and optional session setup data:

```
USESESSION:<session_on_or_off_flag>
PUBEXNGKEYFILE:public_exchange_key_file
PUBEXNGKEYREF:pub_exchange_key_ref_index:pub_exchange_key_ ref_handle
PRIVEXNGKEYFILE:private_exchange_key_file PASSPHRASE:pass_phrase_for_private_exchange_key
OPTIONS:<encrypt><authenticate><fallback on challenge response>
TIMEOUT:time_out_in_hours
```

End the 'SESSIONSETUP' operation with a blank line.

In the above lines, the 'USESESSION' flag is mandatory. Either 'PUBEXNGKEYFILE:' or 'PUBEXNGKEYREF:', and 'PRIVEXNGKEYFILE:', 'OPTIONS:', 'TIMEOUT:' are optional. 'PASSPHRASE:' is conditional.

If 'OPTIONS:' is omitted, session messages will neither be encrypted nor authenticated; however, the "If session fails, use challenge response" flag will be set to make sure requests are carried through without session. The 'SESSIONSETUP' line must come first. The remaining lines can be in any order. Do not include a blank line until it ends.

Example 1: Use public exchange key from server.

```
SESSIONSETUP
USESESSION:1
```

Example 2: Use public exchange key from a file (client provides exchange keys).

```
SESSIONSETUP
USESESSION:1
PUBEXNGKEYFILE:c:\hs\bin\PubKey.bin
PRIVEXNGKEYFILE:c:\hs\bin\PrivKey.bin
PASSPHRASE:secret
OPTIONS:111
TIMEOUT:24
```

Example 3: Use public exchange key from a handle value reference (client provides exchange keys).

```
SESSIONSETUP
USESESSION:1
PUBEXNGKEYREF:300:0.NA/12345
PRIVEXNGKEYFILE:c:\hs\bin\PrivKey.bin
```

3.9 Handle Value Line Format

Each handle value line is composed of:

value_index + space + value_type + space + ttl + space + permission_ set + space + value_data

The value index is a unique integer within the specific handle. The value types are: HS_ADMIN, HS_SECKEY, EMAIL, URL, HS_PUBKEY, URN, HS_SERV, HS_VLIST, HS_ALIAS.

ttl: handle's time to live in cache counted by seconds. Default is 86400(24 hours).
Permission set: permission values indicated by 4 characters, '1' is true, '0' is false, order is: admin read, admin write, public read, public write.
Value data: If the handle value data defines an Administrator, its data format is:
ADMIN + space + admin index:admin permission set + admin handle

The admin permission set is twelve characters with the following order: add handle, delete handle, add naming authority, delete naming authority, modify values, remove values, add values, read values, modify administrator, remove administrator, add administrator and list handles.

If the handle value type is one of HS_SECKEY, HS_SERV, HS_ALIAS, EMAIL, URL, URN, its data will be a string. The value data format is:

UTF8 + space + string_content

If the handle value data is a local file, its data format is:

FILE + space + file_path

If the handle value data is a value reference list, its data format is:

LIST + space + index1:handle1;index2:handle2;

Examples:

(1) Where handle value data is an administrative record:

```
100 HS_ADMIN 86400 1110 ADMIN
300:110011111111:0.NA/12345
```

Explanation:
100 is index;
HS_ADMIN is type;
86400 is the time to live in cache in seconds;
1110 is the value permissions which allow admin write, admin read, public read;
ADMIN indicates that this value data is an administrator record;

300 is the administrator handle index;

110011111111 defines the administration permissions (add handle, delete handle, no add naming authority, no delete naming authority, modify values, remove values, add values, read values, modify administrator, remove administrator, add administrator, list handles);
0.NA/12345 is the administrator handle name;

(2) Where handle value data is a string:

```
2 URL 86400 1110 UTF8 http://www.handle.net/
```

(3) Where handle value data comes from a local file:

```
300 HS_PUBKEY 86400 1110 FILE
c:\somewhere\pubkey.bin 2 HS_SITE 86400 1110 FILE
c:\somewhere\siteinfo.bin
```

(4) Where handle value data is a handle value reference list:

```
1 HS_VLIST 86400 1110 LIST 300:10.50/USR1; 300:10.50/USR2;
```

(5) Example using some of the registered handle value types:

```
100 HS_ADMIN 86400 1110 ADMIN 300:111111111111:0.NA/12345
1 HS_SITE 86400 1110 FILE c:\somewhere\siteinfo.bin
2 HS_SERV 86400 1110 UTF8 0.NA/12345
300 HS_PUBKEY 86400 1110 FILE c:\somewhere\publickey.bin
301 HS_SECKEY 86400 1100 UTF8 my password
400 HS_VLIST 86400 1110 LIST 300:12346/USR1; 300:12347/USR2;
7 EMAIL 86400 1110 UTF8 hdladmin@cnri.reston.va.us
8 URL 86400 1110 UTF8 http://www.handle.net
9 DESC 86400 1110 UTF8 Info about this handle
```

4. Advanced Server Configuration

A handle server can be further configured through the 'config.dct' file located in its installation directory. This file is divided into various sections, each of which consists of a set of configuration values related to a specific part of the server. These sections are detailed below.

4.1 hdl_http_config

This section contains the configuration variables for the HTTP interface to the HANDLE.NET server. You can access this interface by pointing a web browser at your server's IP and HTTP port.

- bind_address: The IP address this interface should use.
- num_threads: The number of threads that should be reserved for answering requests.
- bind_port: The port which the HTTP interface should use.
- backlog: The number of incoming connections that can be queued while all threads are in use.
- log_accesses: "yes" or "no". If set to "yes", each access on the HTTP interface will be logged.
- track_threads: "yes" or "no". If set to "yes", debugging messages concerning thread usage on the HTTP interface will be printed to stdout periodically.
- query_page: Full path and name of the HTML file that your HTTP proxy should use as the query page. If no page is specified, the default Handle System query page will be used.
- response_page: Full path and name of the HTML file that your HTTP proxy should use as the data response page. If no page is specified, the default Handle System response page will be used.
- error_page: Full path and name of the HTML file that your HTTP proxy should use as error page. If no page is specified, the default Handle System error page will be used.
- virtual_hosts: One or more virtual host names and corresponding pages can be specified in this entry. Details of virtual hosts are specified by subentries inside.

An example of virtual host setting would be:

```
"virtual_hosts"=
{         "hostname"="significant.myvirtualhost.com"
          "query_page" ="/home/www/query_page.html"
          "response_page"="/home/www/response_page.html"   "error_page"="/home/www/error_page.html"
}
```

4.2 hdl_tcp_config

This section contains the configuration variables for the TCP interface to the handle server. The TCP interface is required for handle administration.

- bind_address: The IP address the TCP interface should use.
- num_threads: The number of threads that should be reserved for answering requests.
- bind_port: The port the TCP interface should use.
- backlog: The number of incoming connections that can be queued while all threads are in use.
- log_accesses: "yes" or "no". If set to "yes", each access on the TCP interface will be logged.
- track_threads: "yes" or "no". If set to "yes", debugging messages concerning thread usage on the TCP interface will be printed to stdout periodically.

4.3 hdl_udp_config

This section contains the configuration variables for the UDP interface to the handle server.

- bind_address: The IP address the UDP interface should use.
- num_threads: The number of threads that should be reserved for answering requests.
- bind_port: The port the UDP interface should use.
- backlog: The number of incoming connections that can be queued while all threads are in use.
- log_accesses: "yes" or "no". If set to "yes", all accesses on the UDP interface will be logged.
- track_threads: "yes" or "no". If set to "yes", debugging messages concerning thread usage on the UDP interface will be printed to stdout periodically.

4.4 server_config

This section contains the configuration variables for the server.

- server_admins: A list of administrators with permission to un/home prefixes, perform replication, and perform database backups.
- backup_admins: A list of administrators with permission to checkpoint the database.
- replication_admins: A list of administrators with permission to replicate handles on the server.
- replication_interval: Time interval in milliseconds between mirror server updates.
- replication_authentication: On mirror handle servers, this setting indicates the authentication handle that the primary server will use when performing handle replication. Should be of the form handle:index.

- this_server_id: The identification number of this particular server. Used to distinguish from other servers within the same site.
- max_auth_time: The number of seconds to wait for a client to respond to an authentication challenge.
- case_sensitive: "yes" or "no". Whether or not handles are case sensitive. It is highly recommended to always leave this set to "no".
- allow_recursion: "yes" or "no". If set to "yes", the server will act as a proxy for resolving external handles. When a client requests a handle that the server is not responsible for, the server will resolve the handle and return the results, just as if it were stored locally. If allow_recursion is set to "no", the proxy will only allow resolution of handles stored on the local handle server. It is usually safe to set this to "yes", which is the default value.
- preferred_global: A handle server will often need to resolve prefixes from one of the Global Handle Registry servers. Setting this configuration variable to the IP address of a particular global handle server will force the handle server to always use that global handle server or another server within its site (See Section 1.2.1, *Storage*). This may be desirable to reduce latencies if there is a global handle server that is significantly closer geographically than the others.
- max_session_time: Time in milliseconds that an authenticated client session can persist. Minimum is one minute. See the description of sessions in Section 3.8, *Session Setup Information Format* for more information. Defaults to 24 hours.
- replication_timeout: Time in milliseconds before an unresponsive replication operation will timeout. Defaults to 5 minutes.
- storage_type: "bdbje", "jdb", "sql", or "custom"; defaults to "bdbje". This allows manual selection of the storage mechanism used by the server. The "bdbje" storage option uses the Berkeley DB JE hash-style database. The "jdb" storage option is a custom hash style database that comes included with the Handle System Server distribution. The "sql" option allows use of a SQL database for handle storage. See Section 7.1 *Using a SQL Database for Storage*, for more information. Finally, the "custom" setting directs the handle server to use an external Java™ class specified using the storage_class setting.
- storage_class: The name of the Java™ class that should be used when storage_type is set to custom. This class must implement the net.handle.hdllib.HandleStorage interface included with the distribution. It must also be in the Java™ classpath when the handle server is started.
- server_admin_full_access: "yes" or "no". If set to "no" the "server_admins" will have default permissions at the prefix level. These include the ability to home and unhome prefixes, perform replication, and perform database backups. If set to "yes" the "server_admins" will have the default permissions to do any prefix level operations as well as handle level operations, such as creating, deleting, and modifying handles. When server_admin_full_access is enabled, server_admins will be able to modify and delete existing handles, even if they are not explicitly given permission in the handle.
- allow_list_hdls: "yes" or "no". If set to "no" the 'list handles' operation will be disabled on the server.

If you wish to configure additional settings specific to the BDBJE handle storage you can add any of the following lines. The values below are the default values for each setting.

- "db_directory" = "bdbje": This tells the BDBJE which folder should contain the database files.
- "enable_recovery_log" = "false": This tells the storage module if it should or should not keep a log of all changes to the database in a dbtxns.log file in the database directory. These changes are recorded at the handle level. The default is not to record these transactions since BDBJE has its own checkpoint/recovery system.
- "bdbje_dump_on_checkpoint" = "true": This tells the storage module if it should or should not dump a copy of the entire database when a checkpoint is performed. When a checkpoint is performed, the BDBJE checkpoint operation is invoked, but a backup of the database is only made if this option is set to "true".
- "bdbje_no_sync_on_write" = "false": This tells BDBJE to write changes to the database, but do not synchronize afterwards. Setting this to true improves performance, with a slight cost in reliability (which may be negated when using a journaling file system).
- "bdbje_enable_status_handle" = "true": This tells the storage module to send database status information in response to a query for the 0.SITE/DB_STATUS handle, as long as that handle doesn't already exist in the database.

4.5 log_save_config

The code for rotating logs is in the net.handle.server.Main class and the ServerLog class.

The ServerLog looks for an entry named log_save_config to determine the rotation method. The value of the log_save_config is a hashtable similar to the following:

```
log_save_config =
{       log_save_interval = "Weekly"
        log_save_weekday = "Wednesday"
        log_save_directory = "/var/log/hdl/"
}
```

The log_save_interval can be Monthly, Weekly, Daily, or Never (the default value is "Never", so that upgrading users won't see an unexpected change). If it is "Weekly" then there is a log_save_weekday entry that should contain one of Sunday | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday, with the default set to Sunday. (English language weekday names are required.)

There is also a log_save_directory with the value a directory/path where the log files are to be saved. The default is to store the log files in the server directory. If this is a relative path, it will be interpreted as relative to the server directory. Note that the Setup tool asks for the log rotation interval and sets up the config.dct file properly, but omits the log_save_directory and the Weekly rotation option, leaving those as manual settings that can be made by those who wish to edit the config.dct file by hand.

4.6 Additional Settings

- **server_type**: This is a single setting. This can be either "server" or "cache". A cache server does not store any handles or answer for a particular prefix, it just acts as a caching gateway. The benefit of clients using a caching server is that they can all make use of the single cache that the caching server provides. A regular HANDLE.NET server will behave just like a caching server if it does not have any prefixes homed to it.
- **Interfaces**: This is a list of interfaces that the server should answer on. It should contain one or more of "hdl_tcp", "hdl_udp", and "hdl_http". If you wish to disable access via a protocol, remove that protocol from this list.
- **trace_resolution**: Setting this to "yes" will cause the handle server to print out debugging messages concerning handle resolution.
- **tcp_timeout**: This is the number of milliseconds that will pass before an outgoing TCP connection fails. This number can be set lower to avoid wasting threads due to broken connections. However, setting too low will cause slow connections to fail unnecessarily.
- **no_udp_resolution**: Setting this to "yes" will prevent the server from resolving external handles using the UDP based handle protocol. This may be necessary to run a handle server from behind a firewall.

4.7 Example config.dct File

```
{
"hdl_http_config" = {
"bind_address" = "132.151.1.132"
"num_threads" = "15"
"bind_port" = "8000"
"backlog" = "5"
"log_accesses" = "yes"
}
"hdl_tcp_config" = {
"bind_address" = "132.151.1.132"
"num_threads" = "15"
"bind_port" = "2641"
"backlog" = "5"
"log_accesses" = "yes"
}
"hdl_udp_config" = {
"bind_address" = "132.151.1.132"
"num_threads" = "15"
"bind_port" = "2641"
"log_accesses" = "yes"
}
"server_config" = {
"server_admins" = (
"300:0.NA/1234"
)
"replication_admins" = (
"300:0.NA/1234"
)
"max_session_time" = "86400000"
"this_server_id" = "1"
"max_auth_time" = "60000"
"backup_admins" = (
"300:0.NA/1234"
)
"case_sensitive" = "no"
}
"log_save_config" = {
"log_save_weekday" = "Sunday"
"log_save_time" = "00:00:00"
"log_save_directory" = "/dspace/handle-server"
"log_save_interval" = "Weekly"
}
"no_udp_resolution" = "y"
"interfaces" = (
"hdl_udp"
"hdl_tcp"
"hdl_http"
)
"server_type" = "server"
}
```

5. Other Tools and Features

The Handle System Server distribution also includes other small utilities for maintaining a handle server. A selection of them are described below.

5.1 DBTool

DBTool is a graphical utility for operating on a handle server's built-in database. DBTool can be invoked using the following command from the directory containing the "bin" and server configuration directories:

```
bin/hdl-dbtool <serverdir>
```
On Windows: `bin\hdl-dbtool.bat <serverdir>`

where <serverdir> is the directory containing the server configuration and database files.

WARNING: *Do not run this command on a database that is currently in use by a handle server as it could lead to database corruption.*

5.2 DBList/DBRemove

If you would like to operate directly on a handle server's database, but without a GUI, there are two other utilities: DBList and DBRemove.

WARNING: *Do not run these commands on a database that is currently in use by a handle server as it could lead to database corruption.*

DBList will list all the handles in a handle server database. It can be invoked using the following command from the directory containing the "bin" and server configuration directories:

```
bin/hdl-dblist <serverdir>
```
On Windows: `bin\hdl-dblist.bat <serverdir>`

DBRemove will remove a specified handle from the database. It can be run using the command:

```
bin/hdl-dbremove <serverdir>
```
On Windows: `bin\hdl-dbremove.bat <serverdir>`

5.3 Query Resolver

The handle server distribution includes a simple resolver GUI that may be preferable over the resolution facilities in the Admin Tool. It can be run using the following command from the directory containing the "bin" and server configuration directories:

```
bin/hdl-qresolverGUI
```
On Windows: `bin\hdl-qresolverGUI.bat`

There is also a command line resolver that can be run using the command:

```
bin/hdl-qresolverGUI <handle>
```
On Windows: `bin\hdl-qresolverGUI.bat <handle>`

5.4 Test Tool

This tool performs client and local/global handle server diagnostic tests. It is run from the command line and requires certain arguments. The following commands should be run in the same directory containing the "bin" directory.

Client Test with no arguments sends a request to each Global server and tests each interface. Also pings each server within the site and reports average round trip time and percent packet loss.

```
bin/hdl-testtool client
```
On Windows: `bin\hdl-testtool.bat client`

Client Test with prefix argument sends a request to server based on a specified prefix and tests each interface. Also pings each server and reports average round trip time and percent packet loss.

```
bin/hdl-testtool client <Prefix>
```
On Windows: `bin\hdl-testtool.bat client <Prefix>`

Example:

```
bin/hdl-testtool client 0.NA/12345
```

Server Test tests a local server by sending a request to the server and testing each interface.

```
bin/hdl-testtool server <config.dct>
```
On Windows: `bin\hdl-testtool.bat server <config.dct>`

Example:

```
bin/hdl-testtool server /hsj-7.0/svr_1/config.dct
```

Write Test tests handle operations: add, add value, modify value, delete value, delete. If 'admpriv.bin' is not located in the same directory as 'config.dct', user will be prompted for location.

```
bin/hdl-testtool write <config.dct>
```
On Windows: `bin\hdl-testtool.bat write <config.dct>`

Example:

```
bin/hdl-testtool write /hsj-7.0/svr_1
```

Test All performs server test, write test, and Global sites client test.

```
bin/hdl-testtool all <config.dct>
On Windows:  bin\hdl-testtool.bat all <config.dct>
```

Example:

```
bin/hdl-testtool all /hsj-7.0/svr_1
```

5.5 KeyUtil

The KeyUtil.java program allows decrypting and encrypting of private key files. It can be invoked using the command:

```
bin/hdl-keyutil <privkey.bin>
 On Windows:  bin\hdl-keyutil.bat <privkey.bin>
```

If you chose not to encrypt your key at installation, and later change your mind, use this program to encrypt your existing key, and then send the encrypted file to the Handle System Administrator.

5.6 GetRootInfo

GetRootInfo retrieves a current copy of the Global service information for the Handle System. This is the information found in the 'root_info' file.

Usage:

```
bin/hdl-getrootinfo <rootserver> <port> <outfile>
On Windows:  bin\hdl-getrootinfo.bat <root-server> <port> <out-file>
```

5.7 GetSiteInfo

GetSiteInfo retrieves the service information for a handle server. This is the information found in the 'siteinfo.bin' file.

Usage:

```
bin/hdl-getrootinfo <server> <port> <outfile>
On Windows:  bin\hdl-getrootinfo.bat <server> <port> <outfile>
```

5.8 DoCheckpoint

DoCheckpoint is a command line tool to send a checkpoint/backup command to all of the handle servers in a specified site. Execution is done with the following command:

```
bin/hdl-docheckpoint [siteinfo] [adminhdl] [adminindex] [keytype] [key]
```

On Windows:
```
bin\hdl-docheckpoint.bat [siteinfo] [admin-hdl] [admin-index] [keytype] [key]
```

Keytype can be either PRIVATE or SECRET. If PRIVATE, then key is the name of a file where the private key is stored. If SECRET, then key is the secret key itself.

6. Replication

The handle server allows for automatic replication of handles to one or more mirror servers. These mirror servers can be used to provide redundancy for resolution or simply as backup for disaster recovery. In sites with multiple servers, handles are distributed evenly across the site. It should be noted that mirror servers cannot be used for handle administration.

When servers are in the same site, the handles for that service are distributed evenly across the site. Mirrors should be able to replicate from either the primary or existing mirrors.

6.1 Setting up a Single Mirror Handle Server

See Chapter 4, *Advanced Server Configuration*, for details on the configuration settings outlined below.

- Run the Setup as explained in the 'INSTALL.txt' file. Be sure to choose "Mirror Server".
- Send the resulting sitebndl.zip to the Handle System Administrator. The HSA will add the replication public key value (replpub.bin) to your prefix, typically at index 301 (or a different index if 301 is already in use). You will be notified when the change has been made.
- Modify the mirror's config.dct:
  - server_admins, as described in the 'INSTALL.txt' file
  - this_server_id (if more than 1 server in HS SITE)
  - replication_authentication ("privatekey:index:handle")
- Add the replication authentication handle to the primary server’s replication_admins group.
- When the Handle System Administrator notifies you that your prefix has been updated, start the mirror server, and restart the primary server so that it will recognize the replication handle. If you do not see handles being replicated immediately, or see errors in the log, contact the HSA for assistance, because once the mirror’s site info has been added to your prefix, clients will attempt to use it to resolve handles.

The 'txnsrcsv.bin' file is the 'siteinfo.bin' file from the primary server. The 'txnstat.dct' file will be created once the server has replication information to store. When the mirror server is first started, the server has to "dump" all of the handles from the primary server. When that process is done, the mirror server creates and saves the 'txnstat.dct' file with the current transaction ID from each primary server.

6.2 Setting up a Second Mirror Handle Server

If the first mirror server's db file is relatively large, it may be necessary to copy the 'handles.jdb' directory, or the 'handles.jdb' and 'nas.jdb' files, as well as the 'txnstat.dct' file, to the second mirror's directory before starting the second mirror server. Be sure the first mirror server is shut down while copying the files.

7 Using Custom Handle Storage

This section explains how to configure your HANDLE.NET server to use a database for handle storage other than the built-in database. Instructions follow for using SQL, and in particular PostgreSQL.

7.1. Using a SQL Database for Storage

Using a SQL database as storage for a handle server allows greater control over data deposits as well as permitting complex data query.

7.1.1 Configuring the Handle Server

To configure a handle server with an SQL storage module, first run the Setup program for the HANDLE.NET software. Once the setup process is completed, a directory will exist that contains the files necessary to run the new handle server.

In the directory for the new handle server is a file named 'config.dct' that can be modified using a text editor. The 'config.dct' file contains all of the settings for the handle server. The 'config.dct' file has some server-wide settings as well as several subsections that affect different parts of the server. For example, the 'config.dct' file for most handle servers will have sections named hdl tcp config, hdl udp config and hdl http config. Each of these sections holds the settings for one type of "listener" for the handle server.

Normal handle servers (as opposed to simple handle caching servers or http gateways) will also have a section named server config that maintains the settings for the core part of the server. To tell the server to use an SQL backend for storing and retrieving the handles, add the following value to the server config section:

```
storage_type = "sql"
```

Since the specified storage type for this handle server is SQL, some extra settings need to be provided. The following subsection should also be added to the server config section:

```
sql_settings = {
sql_url = "jdbc:sybase:Tds:localhost:2638"
sql_driver = "com.sybase.jdbc.SybDriver"
sql_login = "sqluser"
sql_passwd = "sqlpassword"
sql_read_only = "no"}
```

You will need to change the values to suit your particular installation. Here is an informal description of what each item in this section is for:

- sql url: This setting should specify the JDBC URL that is used to connect to the SQL database. Consult the documentation for the database or JDBC driver for a description of what this setting should look like.

- sql driver: This is the name of a Java™ class that contains the driver for the JDBC connection. Consult the documentation for the database or JDBC driver for a description of what this setting should look like.
- sql login: The user name that should be used by the handle server to connect and perform operations on the database.
- sql passwd: The password that should be used by the handle server to connect and perform operations on the database.
- sql read only: a boolean setting (can be "yes" or "no") that indicates whether or not the server should ever need to modify the database in any way. This is a safeguard used for query-only handle servers.

7.1.2 Example SQL Tables

The default configuration assumes a specific database table setup. The following tables were used for RDBMS storage using MySQL.

```
create table nas (
        na varchar(255) not null,
        PRIMARY KEY(na)

) ;
create table handles (
        handle varchar(255) not null,
        idx int4 not null,
        type blob,
        data blob,
        ttl_type int2,
        ttl int4,
        timestamp int4,
        refs blob,
        admin_read bool,
        admin_write bool,
        pub_read bool,
        pub_write bool,
        PRIMARY KEY(handle, idx)

);
```

These tables were used for Oracle.

```
create table handles (
create table nas (
    na raw(512)
    ) ;
        create table handles (
            handle raw(512),
            idx number(10),
            type raw(128),
            data raw(600),
            ttl_type number(5),
            ttl number(10),
            timestamp number(10),
            refs varchar2(512),
            admin_read varchar2(5),
            admin_write varchar2(5),

        pub_read varchar2(5),
         pub_write varchar2(5)
        ) ;
```

7.1.3 Depositing Handles Outside the Handle Server

If you wish to create or modify handles in the SQL database using custom tools, rather than the handle server, you must use all capital letters for data in the "handle" field, since the Handle System is case insensitive.

7.1.4 Using Custom SQL Statements

It is also possible to specify the SQL that is used by the handle server to query the database. Changing these SQL statements is required if you do not use the same setup as above. The SQL handle storage object used by the handle server has default SQL statements that are used to query and update the database. To replace the default SQL statements with custom statements, simply add the corresponding configuration setting to the sql settings section described above. The following is a list of the SQL statements, their configuration setting, default values, and a short description of what the statement is used for.

7.1.4.1 get handle stmt

Default:

```
    select idx, type, data, ttl_type, ttl, timestamp, refs, admin_read, admin_write, pub_read, pub_write from
    handles where handle = ?
```

Description: This statement is used to retrieve the set of handle values associated with a handle from the database.

Input: The name of the handle to be queried.

Output:

> **idx positive integer value**; unique across all values for the handle
>
> **type alphanumeric value**; indicates the type of the data in each value
>
> data alphanumeric value; the data associated with the value ttl type byte/short; 0=relative, 1=absolute
>
> **ttl numeric**; cache timeout for this value in seconds (if ttl type is absolute, then this indicates the date/time of expiration in seconds since Jan 1 0:00:00 1970.
>
> **timestamp numeric**; the date that this value was last modified, in seconds since Jan 1 0:00:00 1970
>
> **refs alphanumeric**; list of tab delimited index:handle pairs. In each pair, tabs that occur in the handle part are escaped as \t.
>
> **admin read boolean**; indicates whether clients with administrative privileges have access to retrieve the handle value
>
> **admin write boolean**; indicates whether clients with administrative privileges have permission to modify the handle value
>
> **pub read boolean**; indicates whether all clients have permission to retrieve the handle value
>
> **pub write boolean**; indicates whether all clients have permission to modify the handle value

7.1.4.2 have na stmt

Default: select count(*) from nas where na = ?

Description: This statement is used to query whether or not the specified prefix is "homed" to this server.

> Input: The prefix (eg 0.NA/12345)
> Output: One row, with one field. The value of that field is >0 if this server is responsible for the given prefix, or <=0 if not.

7.1.4.3 del na stmt

Default: delete from nas where na = ?

Description: This statement is used to remove a prefix from the list of prefixes for which this server is responsible.

> Input: The prefix handle (e.g., 0.NA/12345)
> Output: None

7.1.4.4 add na stmt

Default: insert into nas (na) values (?)

Description: This statement is used to add a prefix to the list for which this server is responsible.

Input: The prefix to "home" (e.g., 0.NA/12345)
Output: None

7.1.4.5 scan handles stmt

Default: select distinct handle from handles

Description: This statement is used to get a list of all of the handles in the database.

Input: None
Output: a row for each distinct handle in the database.

7.1.4.6 scan by prefix stmt

Default: select distinct handle from handles where handle like ?

Description: This statement is used to get a list of all handles in the database that have a given prefix.

Input: The prefix, including the slash ('/') character
Output: A row for each distinct handle in the database that starts with the given prefix

7.1.4.7 scan nas stmt

Default: select distinct na from nas

Description: This statement is used to get a list of distinct prefixes that call this server home.

Input: None
Output: A row for each distinct prefix

7.1.4.8 delete all handles stmt

Default: delete from handles

Description: This statement is used to delete all of the handles in the database (!) This is only used when the handle server is acting as a secondary/mirror to a primary service and has gotten so far out of sync that it tries to delete and re-copy the entire database from the primary.

Input: None
Output: None

7.1.4.9 delete all nas stmt

Default: delete from nas

Description: This statement is used to delete all of the prefixes in the database. This is only invoked under the same circumstances as delete all handles stmt.

Input: None
Output: None

7.1.4.10 create handle stmt

Default: insert into handles (handle, idx, type, data, ttl_type, ttl, timestamp, refs, admin_read, admin_write, pub_read, pub_write) values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

Description: This statement is used to insert a single handle value into the database.

Input: The fields of the handle and handle value, with the same order and type specified in the default statement above. See get handle stmt for type information for each field.
Output: None

7.1.4.11 handle exists stmt

Default: select count(*) from handles where handle = ?

Description: This statement is used to query whether or not a given handle exists in the database.

Input: The handle being queried
Output: None

7.1.4.12 delete handle stmt

Default: delete from handles where handle = ?

Description: This statement is used to delete a given handle from the database.

Input: The handle being deleted
Output: None

7.1.4.13 modify value stmt

Default: update handles set type = ?, data = ?, ttl_type = ?, ttl = ?, timestamp = ?, refs = ?, admin_read = ?, admin_write = ?, pub_read = ?, pub_write = ? where handle = ? and idx = ?

Description: This statement is used to update a single handle value with new values. The value to update is identified by the handle and index.

Input:

type - alphanumeric; the new type for the handle value

data - alphanumeric; the new data value
ttl_type - byte/short int; indicates whether the cache timeout is specified in relative or absolute terms (1=absolute, 0=relative)
ttl - numeric; indicates the cache timeout in seconds (if ttl type is absolute then the ttl indicates the expiration date in seconds since Jan 1 0:00:00 1970
timestamp - numeric; date of the last modification to this handle value (should be the current date/time!)
refs - alphanumeric; tab delimited list of references for this handle value. See get handle stmt for encoding.
admin_read - boolean; indicates whether clients with administrative privileges have access to retrieve the handle value
admin_write - boolean; indicates whether clients with administrative privileges have permission to modify the handle value
pub_read - boolean; indicates whether all clients have permission to retrieve the handle value
pub_write - boolean; indicates whether all clients have permission to modify the handle value

7.2 Using PostgreSQL Database

The following instructions, provided courtesy of handle users at the Max Planck Institute for Psycholinguistics, are for setting up a PostgreSQL database for handle storage.

As postgres: createuser -PEDA handleserver

Make sure to define a password for that user. Add to

```
/var/lib/pgsql/data/pg_hba.conf
host  handlesystem  handleserver  192.168.0.0/16  md5
```

(This assumes that your intranet uses 192.168.x.x IP addresses.)

Activate the new account: pg_ctl restart -D /var/lib/pgsql/data/

(You may, depending on your configuration, have to replace /var/lib/pgsql/data here and above with something else.)

As an alternative to pg_ctl restart you may use: /etc/init.d/postgresql restart

Create the database and make sure that it uses Unicode: createdb -O handleserver -E unicode handlesystem

Now use the psql shell to create the tables, etc.:

```
psql -h yourservername -U handleserver -d handlesystem
create table nas (na bytea not null, primary key(na));
create table handles (handle bytea not null, idx int4 not
null, type bytea, data bytea, ttl_type int2, ttl int4, timestamp int4, refs text, admin_read bool, admin_write bool,
pub_read bool,
pub_write bool, primary key(handle, idx));
```

```
create index dataindex on handles ( data );
create index handleindex on handles ( handle );
grant all on nas,handles to handleserver;
grant select on nas,handles to public;
\q
```

The \q leaves psql. Note that many columns are bytes, not text.

To backup and restore your handle database, use:

```
to backup: pg_dump handlesystem -F t | gzip -c > handletable.tgz
to list: zcat handletable.tgz | pg_restore -F t -l
to restore: zcat handletable.tgz | pg_restore -F t
```

(With "ddlutils", you can also backup / restore between various databases and XML files, which might be useful for some people.)

To get a description of a database or table, in psql, use:

```
\d (describes the whole database)
\d tablename (describes one table)
```

(As usual, use \q to leave psql again. Note that psql also has nice features like history (cursor up/down) and tab completion.)

To "defragment" and auto-tune for the current contents, use in psql: vacuum analyze handles;

Do this from time to time, especially after larger writes, to gain speed.

The config.dct section for a PostgreSQL database:

```
"storage_type" = "sql"
"sql_settings" = {
"sql_url" = "jdbc:postgresql://YourServerIPAddress/handlesystem"
"sql_driver" = "org.postgresql.Driver"
"sql_login" = "handleserver"
"sql_passwd" = "yourpassword"
"sql_read_only" = "no"
}
```

When you start the handle server, you must have the JDBC for your database in your classpath. You can place the jar file (e.g. postgresql8jdbc3.jar) in the "lib" subdirectory of the unzipped HANDLE.NET distribution.

For the GUI, as usual:

```
bin/hdl-admintool
On Windows:  bin\hdl-admintool.bat
```

Note: These instructions are included courtesy of handle users at the Max Planck Institute for Psycholinguistics and Lund University Libraries NetLab. It is possible that your settings may differ slightly from those in the examples above.

8. The Handle HTTP Proxy

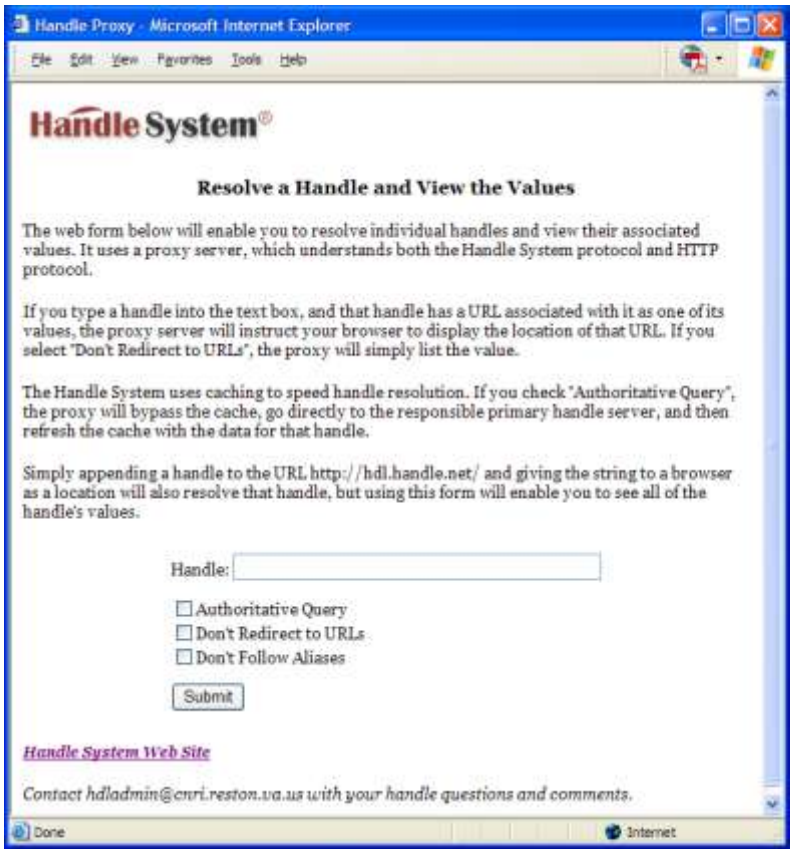
Proxy servers are not part of any Handle System administration or authentication hierarchy. The Handle System protocol does not authenticate any response from a proxy server. Use of a proxy server is a client option, and the client may have to rely on the proxy server to authenticate any service response from Handle System service components. Clients are responsible for setting up their own trust relationship with the proxy server they select.

8.1 Using the Proxy

Unless you disabled the http interface after setting up your handle server, handles can be directly resolved on your handle server using a web browser. A handle server used in this manner is often referred to as a proxy server.

Using HTTP URLs allows handles to be resolved from standard web browsers without additional client software, but requires that the handles be associated with a specific proxy server. If that proxy server changes its DNS name or otherwise becomes invalid, the reference (i.e., the HTTP URL) to the handle will break. Thus selection or use of proxy servers should be carefully evaluated.

You can connect to your server's HTTP interface by opening a URL like http://127.0.0.1:8000. Replace 127.0.0.1 with the IP address or hostname of your handle server. If you changed the HTTP port for the server, replace 8000 with the correct port number. You should see a page like the one below.


Handle Proxy - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Handle System®
Done
Internet

Resolve a Handle and View the Values

The web form below will enable you to resolve individual handles and view their associated values. It uses a proxy server, which understands both the Handle System protocol and HTTP protocol.

If you type a handle into the text box, and that handle has a URL associated with it as one of its values, the proxy server will instruct your browser to display the location of that URL. If you select "Don't Redirect to URLs", the proxy will simply list the value.

The Handle System uses caching to speed handle resolution. If you check "Authoritative Query", the proxy will bypass the cache, go directly to the responsible primary handle server, and then refresh the cache with the data for that handle.

Simply appending a handle to the URL http://hdl.handle.net/ and giving the string to a browser as a location will also resolve that handle, but using this form will enable you to see all of the handle's values.

Handle:

☐ Authoritative Query
☐ Don't Redirect to URLs
☐ Don't Follow Aliases

Submit

Handle System Web Site

Contact hdladmin@cnri.reston.va.us with your handle questions and comments.

Please note that this web form is not part of the standard HANDLE.NET protocol and may change or be completely removed from future versions of the software.

It is also possible to build URLs to the proxy which will automatically resolve or redirect to a specified handle. For a handle server with an IP address of 127.0.0.1 and HTTP interface port 8000 the handle 'my_handle00' can be resolved from a web browser through the URL http://127.0.0.1:8000/my_handle.

If the 'allow_recursion' option is set in the server's configuration (config.dct) file, the HTTP interface will allow resolution of handles which are not stored on the local handle server. When a client requests an external handle, the handle server will resolve the handle and return the results, just as if it were stored locally. This is how public proxy servers like hdl.handle.net and dx.doi.org are configured. If 'allow_recursion' is disabled, the proxy will only allow resolution of handles stored on the local handle server.

8.2 Using Custom HTML Pages

The HANDLE.NET proxy code supports customization of the query, response and error pages. Simple customization of the pages can be performed by modifying copies of the template files included in the 'handle.jar' file. These templates are located in the jar directory 'net/handle/server/html'.

If new templates are created, the server configuration must be modified to use them. See Section 4.1, *hdl_http_config* for instructions.

8.3 Web Interface for Handle Administration

The HANDLE.NET distribution comes with Java™ servlets for handle administration. These servlets are meant to run in a web server using a Java™ servlet engine like Apache Tomcat.

To compile servlets, get the servlet API classes by installing the JSDK (Java™ Servlet Development Kit) and include the JSDK classes in your CLASSPATH. (For information see http://java.sun.com/products/servlet/.

These servlet API classes may be included with servlet engines like Apache Tomcat as a jar file (usually "servlet-api.jar"). If you already have a servlet engine, simply add the jar file to your CLASSPATH to compile servlets.

Instructions for use:

1. Install Java™ version 5 or greater on your computer.

2. You will need to have a web server that supports Java™ servlets.

3. Download, gunzip, and untar the HANDLE.NET distribution.

4. Extract the 'src.jar' file from the handle server directory. The files for setting up the web interface are in 'net/handle/apps/admin_servlets'. The servlet code, for web administration and handle resolution, is four files,

and there is one corresponding 'htdocs' directory containing HTML files that will be returned from the servlet/proxy.

5. In 'Admin.java', change the value of the YOUR_NAMING_AUTHORITY variable to your prefix. Change the value of the 'ADMIN_NA' variable to your administrative handle. The default index values for SEC_KEY_IDX is 300 and ADMIN_GROUP_IDX is 200. Include the 'handle.jar' file and the servlet library from your servlet engine (usually 'jsdk.jar') with your CLASSPATH variable. Compile 'Admin.java'.

6. You will need to modify the HTML files under

   ```
   'net/handle/apps/admin_servlets/htdocs'
   ```

   to fit your specific purpose. Change the email address and prefix found in the following html files: admin_footer.html, admin_login.html, help.html, index.html, qform.html.

7. 'Admin.java' uses two parameters, TEMPLATE_DIR_KEY and BATCH_DIR_KEY, which are specified in the servlet properties file.

   ```
   TEMPLATE_DIR_KEY: path to html files
   BATCH_DIR_KEY: path to batch directory
   ```

8. In an Apache JServ setup, the zone.properties file should be modified as follows:

   ```
   servlet.net.handle.apps.admin_servlet.Admin.initArgs=
           admin_servlets.html_template_dir=<path to the html files>,
           admin_servlets.dir=<path to the batch directory>
   ```

9. In an Apache Jakarta Tomcat setup, the web.xml file should be modified as follows:

   ```
   <initparam>
           <paramname>webadmin_servlets.html_template_dir</paramname>
           <paramvalue><path to the html files></paramvalue>
            <paramname>webadmin_servlets.batch_dir</paramname>
           <paramvalue><path to the batch directory></paramvalue>
   </initparam>
   ```

10. Add the servlet files to your web server or servlet engine.

11. To access the Admin servlet, use the following syntax:

    http://host/servlets/net.handle.apps.admin_servlets.Admin
    You need to add the servlet files to your web server or servlet engine.

See configuration file for mount point for servlet zones.

After setting up and accessing the administration servlet you must login. The administration servlet uses secret key authentication. Create an administrative handle for use in the servlet. For example, if you plan to administer handles under the prefix 0.NA/345678, create the handle 345678/admin to use for web administration. Create the handle with the following values using one of the handle administration tools described in Chapter 3:

- An HS_ADMIN value at index 100, with Admin ID Handle=345678/admin, Admin ID Index=300 and any permissions you want.
- An HS_SECKEY value at index 300 with any value, e.g., "secretkey". This value will be your login password.

The handle (345678/admin) must be added to the admin group list in the prefix record on GHR.

When you access the administration servlet through your web browser, do the following to log in:

Handle Prefix: 345678
User ID: admin
Password: your_secretkey

9 Handle Clients & the Client Library (Java™ Version)

Communicating with the Handle System is accomplished by sending requests to servers which then return a response. To resolve a handle, a ResolutionRequest is sent to a server. To create a handle, a CreateHandleRequest is sent. To modify, remove, or add values to (or from) a handle, a ModifyValueRequest, RemoveValueRequest, or AddValueRequest is sent to a server.

There is an object for each of these requests in the net.handle.hdllib java package. One way to send these messages to a server is to use a HandleResolver object which is located in the net.handle.hdllib package. For most messages, the HandleResolver object will locate the server that your messages should go to, send them, and return the response that was sent by the server. The following is an example that shows one way of programmatically resolving a handle:

```
import net.handle.hdllib.*;
...
// Get the UTF8 encoding of the desired handle.
byte someHandle[] = Util.encodeString("45678/1");

// Create a resolution request.
// (without specifying any types, indexes, or authentication info)
ResolutionRequest request =
  new ResolutionRequest(someHandle, null, null, null);

HandleResolver resolver = new HandleResolver();

// Create a resolver that will send the request and return the
  response.
AbstractResponse response = resolver.processRequest(request);

// Check the response to see if the operation was successful. if(response.responseCode ==
AbstractMessage.RC_SUCCESS) {

// The resolution was successful, so we'll cast the response
// and get the handle values.
  HandleValue values[]
  =((ResolutionResponse)response).getHandleValues();
  for(int i=0; i < values.length; i++) {
  System.out.println(String.valueOf(values[i]));
          }
}
```

To simply resolve a handle, the much simpler resolveHandle method of the HandleResolver can be used, as shown below.

```
import net.handle.hdllib.*;
...
HandleValue values[] =
  new HandleResolver().resolveHandle("12345/1", null, null); for(int i=0; i < values.length; i++){
  System.out.println(String.valueOf(values[i]));
}
```

The HANDLE.NET software distribution include a "simple" package with command line tools to create, delete, and list handles. It also includes programs to home a prefix and trace handle resolution. These programs provide a good starting point and simple guide to developing Java™-based custom handle client software with the API. Each example program includes steps needed to form a handle request to send to a handle server. The programs are run from the command line and require certain arguments. The following commands should be run from the directory containing the "bin" directory.

(1) Create Handle:

Simple tool for handle creation. It uses public key authentication.

```
bin/hdl-create <auth handle> <auth index> <privkey> <handle>
```
On Windows: `bin\hdl-create.bat <auth handle> <auth index> <privkey> <handle>`

(2) Delete Handle:

Simple tool for handle deletion. It uses public key authentication.

```
bin/hdl-delete <auth handle> <privkey> <filename_of_file_with_handles_to_delete>
```
On Windows: `bin\hdl-delete.bat <auth handle> <auth index> <privkey> <handle>`

(3) List Handles:

Simple tool for listing handles. It uses public key authentication.

```
bin/hdl-list <auth handle> <auth index> <privkey> <prefix>
```
On Windows: `bin\hdl-list.bat <auth handle> <auth index> <privkey> <prefix>`

(4) Trace handle:

Simple tool for resolving a handle.

```
bin/hdl-trace <handle>
```

On Windows: bin\hdl-trace.bat <handle>

(5) Home Prefixes:

Simple tool for homing Prefixes. It uses public key authentication.

```
bin/hdl-home-na <auth hdl> <privkey> <server ip> <NA handle>
```
On Windows: bin\hdl-home-na.bat <auth hdl> <privkey> <server ip> <NA handle>

10. Configuring an Independent Handle Service

An independent or private handle service (such as a service maintained behind a firewall that is not publicly accessible) operates without contacting the Global Handle Registry. Configuring an independent service requires changes to the client for resolution to occur, and to the server for enabling authentication, homing and administrative tasks to be performed without the GHR.

Resolution Service Providers who wish to operate an independent handle service must notify the Handle System Administrator in advance.

10.1 Client Configuration Details

This section explains how to configure the java client software to resolve handles locally, either through a resolution/caching server, or by directing specific prefixes to a certain service/site.

To specify a local handle server that should be used to process all resolution requests, follow these instructions:

Copy the siteinfo.bin file that describes the site/server where all resolution should be performed into a file called "resolver_site" in the ".handle" sub-directory of the user's "home" directory. This will cause all non-administrative requests to be sent through the site described by siteinfo.bin. This should make resolution faster for organizations that can use the resolution server as a shared cache.

By default, no administrative messages are sent through this site (because administration must be done directly with the site that is responsible for each prefix and cannot be "tunneled".) To force all messages (including administration messages) to go to the local resolution server described above, the user must specify the prefixes that are "homed" on the resolution server. All other prefixes will bypass the local resolution server. To specify the prefixes, do the following:

Create a file called "local_nas" in the ".handle" sub-directory of the users home directory. This file should contain one prefix handle on each line (e.g., "0.NA/11234"), encoded in UTF8 (ASCII is OK as long as there are no special characters). Every request for a handle having a prefix contained in this file will be sent to the local resolution site. If no resolver_site file is provided, the local_nas file is ignored.

If there is more than one local handle gateway the methods described in the previous sections will not work. In this case a local_info file must be placed in the .handle directory on each local client machine. This file is prepared using a special utility that can be invoked using the command:

```
bin/hdl-local-info
```
On Windows: `bin\hdl-local-info.bat`

This tool allows creation of a local_info file by creating a list of local handle gateways. For each gateway, use the "Load From File" button in the tool to import the siteinfo.bin file from the gateway's handle server directory. Once the site information is loaded, use the "Add" button in the Naming Authorities section to create a list of prefixes this gateway should be responsible for.

Once the site information and naming authority list has been set for each local handle gateway, you can use the "Save to File" button on the main window to save your local_info file.

10.2 Server Side Configuration

By default, authentication, homing, and administering handles on a handle server require your handle server to communicate with the Global Handle Registry. This section describes how to configure your handle server so that administration of handles can be done without communicating with the Global Handle Registry.

(1) Modify the config.dct file:

"server_admin_full_access" = "y"

"allow_na_admins" = "no"

(2) To home a prefix on your handle server without contacting the Global Handle Registry, add the prefix to the handle storage using the DBTool (See Chapter 5.1, *DBTool*).

(3) Once a prefix has been homed, create a new admin handle for it. (The default admin handle is the prefix itself. This default value cannot be used because it requires communication with the Global Handle Registry.) Create the new admin handle using the DBTool, and associate a secret key (password) with it at index 300. For example, if your prefix is 1234, add 0.NA/1234 to the homed prefixes using the DBTool, then create the new admin handle 1234/ADMIN (use upper case when using the DBTool) with a secret key at index 300.

(4) Edit the config.dct file to change the "server_admins" entry to the new admin handle.

(5) Restart the server.

11. Template Handles

A single template handle can be created as a base that will allow any number of extensions to that base to be resolved as full handles, according to a pattern, without each such handle being individually registered. This would allow, for example, the use of handles to reference an unlimited number of ranges within a video without each potential range having to be registered with a separate handle. If the pattern needs to be changed, e.g., the video moves or a different kind of server is used to deliver the video clips, only the single base handle needs to be changed to allow an unlimited number of previously constructed extensions to continue to resolve properly.

When a server receives a resolution request for a handle which is not in its database, it tries to determines if there is template for constructing the handle values.

11.1 The Template Delimiter

First, it looks for a *template delimiter*, which is a string dividing the original handle into a *base* and an *extension*. The delimiter is generally defined in an HS_NAMESPACE value of the prefix handle 0.NA/prefix.

An example:

```
<namespace>
<template delimiter="@" />
</namespace>
```

If there is no namespace info, the server will use the "namespace" value in the "server_config" section of its config.dct configuration file. If there is no value, or if no template delimiter is defined in the namespace (either from the prefix or the config file) the server will use the "template_delimiter" value in the "server_config" section of config.dct.

If there is namespace layering due to delegation, the innermost defined delimiter is used.

If a delimiter is found, the server looks up the base handle (i.e., the part before the delimiter). For example, in cnri.test.1/weather@foo, with delimiter @, the base handle is cnri.test.1/weather and the extension is foo.

A delimiter of "/" will enable an entire prefix to be templated. In this case the base handle is considered to have no values.

11.2 Template construction

Any HS_NAMESPACE value in the base handle will override any prefix namespace info — in particular, templates can be put directly into the base handle.

For backward compatibility, if there is no namespace info or no <template> in the namespace info, and there are values in the base handle, the template handle is considered to have the same values as the base handle. The same result can be obtained with <template><foreach><value/></foreach></template>.

If there is a namespace, it is used to construct the values of the template handle. Each <template> element within the namespace is applied in order. If no template is found, enclosing parent namespaces are tried. If no template is found at all, the server returns "handle not found".

(1) The template XML itself will contain <value> tags defining the handle values of the eventual result. The <value> tags can specify index, type, and data using attributes. The values of these attributes can refer to the parameters "${handle}", "${base}" and "${extension}". Data can also be specified as the contents of the <value> tag, instead of as an attribute.

(2) Some <value> tags may only be conditionally part of the constructed handle; these are enclosed in <if> tags. The only tests useable in an <if> are string equality and regular-expression matching. With a regular-expression match, various submatches can be referred to in enclosed values with syntax like "${extension[2]}". There is also an <else> tag.

- Details of <if> syntax:
    - value attribute is some parameter name (e.g., handle, base, extension; inside of a <foreach>, index, type, or data). The syntax value="extension[2]" works inside a nested if.
    - parameter attribute is the name of the parameter used to refer to submatches. Default is same as value.
    - test attribute is "equals" or "matches"
    - negate="true" negates the test
    - expression is the string used for equality comparison or RE matching.

(3) Any <notfound/> tag will cause the handle server to return "handle not found".

(4) With <def parameter="param"> ... </def> a new parameter ${param} can be defined. The value of the parameter is obtained by processing the contents of the <def> tag as a template. The data of any constructed handle value if the definition of the parameter. The simplest example is <def parameter="param"><value data="foo"/></def> which defines ${param} to be foo.

(5) Finally, it is possible to produce a value or values for each value already in the base handle. This is useful for having template handles which are identical to the base handle except perhaps for transforming the data of some particular handle value (e.g. a URL). Any values enclosed in a <foreach> tag will be constructed for each value of the base handle. Within the <foreach>, the parameters "${index}", "${type}", and "${data}" can be used to refer to the original handle value from the base handle. Within a <foreach>, <value> tags can omit type or data attributes, in which case the type or data from the original value will be used unchanged.

Here is an example.

```
<namespace>
  <template delimiter="@">
    <foreach>
      <if value="type" test="equals" expression="URL">
        <if value="extension" test="matches" expression="box\(([^,]*),([^,]*),([^,]*),([^,]*)\)"
          parameter="x">
           <value data= "${data}?wh=${x[4]}&amp;ww=${x[3]}&amp;wy=${x[2]}&amp;wx=${x[1]}" />
        </if>
```

```
                        <else>
                           <value data="${data}?${x}" />
                        </else>
                     </if>
                     <else>
                        <value />
                     </else>
                  </foreach>
               </template>
            </namespace>
```

This example produces exactly one value for each value in the base handle. If the type of the original value is "URL", we produce a new value with changed data; the new data depends on the format of the extension. An extension of the form "box(#,#,#,#)" produces a new URL with the four values to be used as query parameters; any other extension is appended as written onto the original URL. If the type of the original value is not "URL", the original value is used unchanged.

The RE language is Java's (http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html).

11.3 Template handles by reference

XML of the form

```
<template ref="10:abc/def" />
```

will be taken to refer to the handle value of index 10 of handle "abc/def". The data of the handle is parsed as XML and interpreted as above. Too much recursion, or a "handle not found" result, any resolution error, or failure to parse the referenced tag, all lead to a "handle not found" result.

11. Template Handles

A single template handle can be created as a base that will allow any number of extensions to that base to be resolved as full handles, according to a pattern, without each such handle being individually registered. This would allow, for example, the use of handles to reference an unlimited number of ranges within a video without each potential range having to be registered with a separate handle. If the pattern needs to be changed, e.g., the video moves or a different kind of server is used to deliver the video clips, only the single base handle needs to be changed to allow an unlimited number of previously constructed extensions to continue to resolve properly.

When a server receives a resolution request for a handle which is not in its database, it tries to determines if there is template for constructing the handle values.

11.1 The Template Delimiter

First, it looks for a *template delimiter*, which is a string dividing the original handle into a *base* and an *extension*. The delimiter is generally defined in an HS_NAMESPACE value of the prefix handle 0.NA/prefix.

An example:

```
<namespace>
<template delimiter="@" />
</namespace>
```

If there is no namespace info, the server will use the "namespace" value in the "server_config" section of its config.dct configuration file. If there is no value, or if no template delimiter is defined in the namespace (either from the prefix or the config file) the server will use the "template_delimiter" value in the "server_config" section of config.dct.

If a delimiter is found, the server looks up the base handle (i.e., the part before the delimiter). For example, in cnri.test.1/weather@foo, with delimiter @, the base handle is cnri.test.1/weather and the extension is foo.

A delimiter of "/" will enable an entire prefix to be templated. In this case the base handle is considered to have no values.

11.2 Template construction

Any HS_NAMESPACE value in the base handle will override any prefix namespace info — in particular, templates can be put directly into the base handle.

If there is a namespace, it is used to construct the values of the template handle. Each <template> element within the namespace is applied in order. If no template is found, enclosing parent namespaces are tried. If no template is found at all, the server returns "handle not found".

(1) The template XML itself will contain <value> tags defining the handle values of the eventual result. The <value> tags can specify index, type, and data using attributes. The values of these attributes can refer to the parameters "${handle}", "${base}" and "${extension}". Data can also be specified as the contents of the <value> tag, instead of as an attribute.

(2) Some <value> tags may only be conditionally part of the constructed handle; these are enclosed in <if> tags. The only tests useable in an <if> are string equality and regular-expression matching. With a regular-expression match, various submatches can be referred to in enclosed values with syntax like "${extension[2]}". There is also an <else> tag.

Details of <if> syntax:

- value attribute is some parameter name (e.g., handle, base, extension; inside of a <foreach>, index, type, or data). The syntax value="extension[2]" works inside a nested if.

- parameter attribute is the name of the parameter used to refer to submatches. Default is same as value.
- test attribute is "equals" or "matches"
- negate="true" negates the test
- expression is the string used for equality comparison or RE matching.

(3) Any <notfound/> tag will cause the handle server to return "handle not found".

(4) With <def parameter="param"> ... </def> a new parameter ${param} can be defined. The value of the parameter is obtained by processing the contents of the <def> tag as a template. The data of any constructed handle value if the definition of the parameter. The simplest example is <def parameter="param"><value data="foo"/></def> which defines ${param} to be foo.

(5) Finally, it is possible to produce a value or values for each value already in the base handle. This is useful for having template handles which are identical to the base handle except perhaps for transforming the data of some particular handle value (e.g. a URL). Any values enclosed in a <foreach> tag will be constructed for each value of the base handle. Within the <foreach>, the parameters "${index}", "${type}", and "${data}" can be used to refer to the original handle value from the base handle. Within a <foreach>, <value> tags can omit type or data attributes, in which case the type or data from the original value will be used unchanged.

Here is an example.

```
<namespace>
  <template delimiter="@">
    <foreach>
      <if value="type" test="equals" expression="URL">
        <if value="extension" test="matches" expression="box\(([^,]*),([^,]*),([^,]*),([^,]*)\)"
          parameter="x">
           <value data= "${data}?wh=${x[4]}&amp;ww=${x[3]}&amp;wy=${x[2]}&amp;wx=${x[1]}" />
        </if>
        <else>
           <value data="${data}?${x}" />
        </else>
      </if>
      <else>
        <value />
      </else>
    </foreach>
  </template>
</namespace>
```

This example produces exactly one value for each value in the base handle. If the type of the original value is "URL", we produce a new value with changed data; the new data depends on the format of the extension. An extension of the form "box(#,#,#,#)" produces a new URL with the four values to be used as query parameters; any other extension is appended as written onto the original URL. If the type of the original value is not "URL", the original value is used unchanged.

For example, suppose we have the above namespace value in 0.NA/1234, and 1234/abc contains two handle values:

1	URL	http://example.org/data/abc
2	EMAIL	contact@example.org

Then 1234/abc@box(10,20,30,40) resolves with two handle values:

1	URL	http://example.org/data/abc?wh=40&ww=30&wy=20&wx=10
2	EMAIL	contact@example.org

For more on the RE language, see http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html.

11.3 Template handles by reference

XML of the form

```
<template ref="10:abc/def" />
```

will be taken to refer to the handle value of index 10 of handle "abc/def". The data of the handle value is parsed as XML and interpreted as above. Too much recursion, or a "handle not found" result, any resolution error, or failure to parse the referenced tag, all lead to a "handle not found" result.

12. The 10320/loc Handle Type

For handles with multiple URL values, the proxy server (or web browser plug-in) simply selects the first URL value in the list of values returned by the handle resolution. Because the order of that list is nondeterministic, there is no intelligent selection of a URL to which the client would be redirected. The 10320/loc handle value type was developed to improve the selection of specific resource URLs and to add features to the handle-to-URL resolution process. (Note that the prefix '10320' has been set aside by the Handle System administrator for handle application types.)

Type 10320/loc specifies an XML-formatted handle value that contains a list of locations. Each location has a set of associated attributes that help determine if or when that location is used. The overall list of locations can include hints for how the resolving client should select a location, including an ordered set of selection methods. Resolvers can apply each known selection method, in order, to choose a location based on the resolver's context (the HTTP request in the case of the proxy server) and the attributes of each location.

The attributes for the set of locations, as well as each location entry in the set, are open-ended to allow for future capabilities to be added in a backwards-compatible way. A small number of attributes have been defined as "standard" that all resolvers should understand.

At the top level of the XML structure are the following defined attributes:

chooseby (optional)

The chooseby attribute identifies a comma-delimited list of selection methods. If no chooseby attribute is specified then the default (currently "locatt,country,weighted") is assumed.

For each location the following attributes are defined:

href (required)

The URL for the location.

weight (optional)

The weight (from zero to one) that should apply to this location when performing a random selection. Setting the weight attribute to zero results in the location not being selected unless a) it is explicitly referenced by another attribute; b) there are no other suitable locations; or c) the location is selected based on one of the other selection methods, such as country or language. If a location has no weight attribute then it is assumed to have a weight of one.

The currently defined selection methods are:

locatt

Select only locations from an attribute passed in the proxy/handle-URI link. If someone constructs a link as hdl:123/456?locatt=id:1 then the resolver will return the locations that have an "id" attribute of 1 (i.e., the second location in the resolution example below).

country

> Selects only locations that have a 'country' attribute matching the country of the client. If no matching locations are found then this selects locations that have no country attribute (i.e., not a mismatch). The Proxy determines the country of the client using a GeoIP lookup.

weighted

> Selects a single location based on a random choice. The Proxy will observe the 'weight' attribute for each location, which should be a floating point non-negative number. The weighting allows for a very basic load balancing, but is also a way to ensure that some locations can only be addressed directly (for example by country or locatt/attributes). If the weighted selection method is applied to locations that all have non-positive weights, then this selects one of the remaining locations randomly while disregarding location weights.

The Proxy will iterate over the known selection methods, in order, until a single location has been selected. After each iteration the Proxy will take one of four steps:

- if there is only one remaining location element, it is returned as a redirect;
- if there are no remaining location elements, the Proxy reverts to the location elements as they were before the last method was applied;
- if there are multiple location elements the Proxy will apply the remaining selection methods to those locations;
- if there are no more selection methods to try, the weighted random selection method is applied, which is guaranteed to return a single location. In a sense, the weighted random is always the "fallback".

For handle 123/456, with a value type 10320/loc that has this list of location attributes:

```
<locations>
  <location id="0" href="http://uk.example.com/" country="gb" weight="0" />
  <location id="1" href="http://www1.example.com/" weight="1" />
  <location id="2" href="http://www2.example.com/" weight="1" />
</locations>
```

the following selections could be made:

Reference: 123/456 from a client located in the UK
Result: The "country" selection method selects the first location based on the 'country' attribute of the first location and the client's position.

Reference: 123/456 from a client located outside the UK
Result: The "country" selection method removes the first location from consideration based on its 'country' attribute and chooses one of the last two locations using the "weighted" random selection method.

Reference: 123/456?locatt=id:1
Result: The second location is used based on the "locatt" selection method and the 'id' attribute.

Reference: 123/456?locatt=id:0
Result: The first location is used based on the "locatt" selection method and the 'id' attribute. The resolver never gets to the "country" selection method as the "locatt" selection method resulted in only a single matching location.

Reference: 123/456?locatt=country:uk
Result: The first location is used based on the "locatt" selection method and the 'country' attribute.

Reference: 123/456?locatt=country:us
Result: The "country" selection method removes the first location from consideration based on its 'country' attribute, finds no US-specific location, and chooses one of the last two locations using the "weighted" random selection method.