INTERNET-DRAFT
draft-irtf-idrm-handle-system-def-01.txt

Sam X. Sun
Sean Reilly
Larry Lannom
CNRI
May 2001

Handle System Namespace and Service Definition

<draft-irtf-idrm-handle-system-def-01.txt>

Status of this Memo

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [KEYWORDS].

Abstract

The Handle System is a general-purpose global name service that allows secured name resolution and administration over the public Internet. The Handle System manages handles, which are unique names for digital objects and other Internet resources. This document provides a detailed description of the Handle System namespace, data and service model, as well as its operation and authentication protocol. It assumes that readers are familiar with the basic concepts of the Handle System as introduced in the overview document.

Table of Contents

1.   Introduction
2.   Handle System Namespace
3.   Handle System Data Model

3.1. Handle Value Set
3.2. Pre-defined Handle Data Types
3.2.1. Handle Administrator: HS_ADMIN
3.2.2. Service Site Information: HS_SITE
3.2.3. Service Handle: HS_SERV
3.2.4. Handle Alias: HS_ALIAS
3.2.5. Primary Site: HS_PRIMARY
3.2.6. Handle Value List: HS_VLIST
4. Handle System Service Model
4.1. Handle System Service Components
4.1.1. Global Handle Registry (GHR)
4.1.2. Local Handle Service (LHS)
4.2. Handle System Middle-Ware Components
4.2.1. Handle System Caching Service
4.2.2. Handle System Proxy Servers
4.3. Handle System Client Components
5. Handle System Operation Model
5.1. Handle System Service Request and Response
5.2. Handle System Authentication Protocol
6. Security Consideration
7. Acknowledgement
8. Authors' Address
9. References

1. Introduction

The Handle System(r) manages handles as globally unique names for Internet resources. It provides a general-purpose global name service that allows handles to be resolved and administrated securely over the Internet. The Handle System categorizes its service into two categories, that is, the handle resolution service and the handle administration service. Clients use handle resolution service to resolve handles into their values. The handle administration service deals with client requests to manage these handles, including adding and deleting handles, or updating their values. It also deals with authority administration via naming authority handles. The document "Handle System Overview" [1] provides an architectural overview of the Handle System, as well as its relationship to other Internet services such as DNS [2,3] and LDAP[4]. This document provides a detailed description of the Handle System namespace, data and service model, and operations. It assumes that readers are familiar with the basic concepts of the Handle System as introduced in the overview document.

The namespace definition specifies the handle syntax and its semantic structure. The data model defines the data structure used by the handle system protocol, and pre-defined data types for carrying out the handle service. The service model provides a definition of various components of the Handle System, and explains how they work together over the Internet. Finally, the handle system operation model describes its service operation in terms of messages transmitted between client and

server, as well as client authentication using the handle system
authentication protocol.

2. Handle System Namespace

Handles are character strings that may consist of a wide range of
characters. Every handle in the Handle System consists of two parts:
its naming authority, followed by a unique local name under the naming
authority. The naming authority and the local name are separated by the
ASCII character "/" (octet 0x2F). The following table provides the
handle syntax definition in ABNF [5] notation:

```
    <Handle>          = <NamingAuthority> "/" <LocalName>

    <NamingAuthority> = *(<NamingAuthority>  ".") <NAsegment>

    <NAsegment>       = 1*(%x00-2D  /  %x30-3F / %x41-FF )
                      ; any octets that map to UTF-8 encoded
                      ; Unicode 2.0 characters except
                      ; octets '0x2E' and '0x2F' (which
                      ; correspond to the ASCII characters '.',
                      ; and '/').

    <LocalName>       = *(%x00-FF)
                      ; any octets that map to UTF-8 encoded
                      ; Unicode 2.0 characters
```

Table 2.1   Handle syntax definition

As shown in Table 2.1, both <NamingAuthority> and <LocalName> are UTF-8
[6] encoded character strings. The handle system protocol mandates UTF-
8 **encoding for handles transferred over the wire. The <LocalName> may**
consist of any printable characters from the Unicode 2.0 standard [7].
The <NamingAuthority> may use any printable characters from the Unicode
2.0 **standard except the ASCII character '/' (0x2F), which is reserved**
to separate the <NamingAuthority> from the <LocalName>. A
<NamingAuthority> may consist of multiple non-empty <NAsegment>s, each
of which separated by the ASCII character '.' (octet 0x2E).

Naming authorities are defined in a hierarchical fashion resembling a
tree structure. Each node and leaf of the tree are given a label that
corresponds to a naming authority segment (<NAsegment>). The parent
node represents the parent naming authority. Naming authorities are
constructed left to right, concatenating the labels from the root of
the tree to the node that represents the naming authority. Each label
(or its <NAsegment>) is separated by the character '.' (octet 0x2E).
For example, the naming authority for the Digital Object Identifier
(DOI) project is "10". It is a root-level naming authority as it has
no parent naming authority for itself. It can, however, have many child

naming authorities, e.g., "10.1045" which is used as a naming authority for D-Lib Magazine.

By default, handles are case sensitive. However, a handle service, global or local, may implement its namespace so that ASCII characters under the namespace are treated as case insensitive. For example, the global handle service, formally known as the Global Handle Registry (GHR), is implemented such that ASCII characters are treated as case insensitive. Since GHR manages every handle system naming authority handle, ASCII characters in all naming authorities are treated as case insensitive.

3. Handle System Data Model

The Handle System provides a name to value binding service so that a handle can be used as a reference to any Internet resource. Each handle may have a set of values assigned to it. The Handle System maintains the value set of each handle and will return it in response to the handle resolution request. The handle system data model defines the conceptual data structure for these values. The data model used by the protocol may not correspond to the physical data model used for storage in any specific implementation. Rather, it is the data model followed by the handle system protocol as specified in the "Handle System Protocol Specification" [8].

3.1. Handle Value Set

Each handle may have a set of values assigned to it. Each value has a unique index number that distinguishes it from the other values of the set. Each value also has a specific data type that defines the syntax and semantics of its data, and each value has associated administrative information such as TTL and permissions. Each of these complex handle value records, which are also referred to simply as handle values, is bound to a given handle and any given handle can be bound with one or more handle values. Figure 3.1 shows the handle "10.1045/may99-payette" with a set of three handle values assigned to it, one of which (index number 1) is shown in detail. (Note that the encoding of the length for each field is not shown in Figure 3.1.)

```
                Handle "10.1045/may99-payette"

                              |
                              |
                              V

        ---------------------------------------------------------------
       |        <index>:             3                                 |
    -----------------------------------------------------------------  |
   |        <index>:             2                                   | |
 ------------------------------------------------------------------  | |
|                                                                  | | |
|  <index>:           1                                            | | |
|  <type>:            URL                                          | | |
|  <data>:            http://www.dlib.org/dlib...                  | | |
|  <TTL>:             {Relative: 24 hours}                         | | |
|  <permission>:      public-read, authorized-write                | | |
|  <timestamp>:       927314334000                                 | | |
|  <reference>:       {empty}                                      | |-
|                                                                  |-
 -----------------------------------------------------------------
```

Figure 3.1. Handle "10.1045/may99-payette" and its set of values.

The value record whose index is 1 has a URL data type. The URL data as specified in the <data> field is "http://www.dlib.org/dlib/may99/payette/05payette.html". The TTL (time to live) entry suggests that the value record should be cached no more than 24 hours before the source of the information is consulted again. The <permission> field grants anyone permission to read, but only the administrator may modify (or delete) the value record. The <reference> field is empty. It could contain a list of references to other handle values in order to provide additional credentials for this value record.

Thus, a handle value may be thought of as a record that consists of a group of fields, e.g., the <index> field, the <type> field, and so on. Each of these fields is defined below:

<index>
An unsigned 32-bit integer that uniquely identifies the value from the value set for that handle.

<type>
A UTF8-string that identifies the data type for the value record. Note that throughout this document, a UTF8-string is defined as a data structure that consists of a 4-byte unsigned integer followed by an UTF-8 encoded character string. The integer specifies the number of octets in the character string. The <type> field identifies the data type that defines the syntax and semantics for data in the next <data> field. The data type should be registered with the Handle System to avoid potential conflicts. The Handle System has a reserved naming authority "0.TYPE" for registered data types. For example, "URL" (as shown in Figure 3.1) is a registered data type. It is registered as the handle "0.TYPE/URL". The handle may have a string value (e.g., referening to RFC1738 [9]) that explains the syntax and semantics of the data type.

<data>
A sequence of octets (preceded by its length in a 4-byte unsigned integer) that describes the resource identified by the handle. The syntax and semantics of these octets are identified by the <type>

field.

<permission>
An eight-bit bit mask that defines the access control of the value record. Access control is defined in terms of read or write permission. Read permission is categorized as public-read, authorized-read, or no-read. Write access is similarly categorized as public-write, authorized-write, or no-write. Public-read (or public-write) permission grants read (or write) access to any user. Authorized-read (or authorized-write) limits read (or write) access to handle administrators only. A handle value with no-read permission cannot leave the server and so may be used, for example, to store secret keys for authentication. The no-write permission makes the value immutable and prevents it from being deleted (via the protocol). By default, any handle value allows read access without authentication (i.e., public-read), but requires authentication for write access (i.e., authorized-write). The administrator for a given handle must specify the permission for each handle value. Handle servers must check permissions before fulfilling any client request.

<TTL>
An octet followed by a 4-byte integer that specifies the Time-To-Live of the value record. The TTL describes how long the value record can be cached before the source of the information should again be consulted. A zero value for a TTL indicates that the value record should only be used for the transaction in progress, and should not be cached. If the TTL field is empty, a 24-hour default value is assumed. Any non-zero TTL is defined in terms of a TTL type (specified in the first octet), followed by the TTL value as a 32-bit unsigned integer. The TTL type indicates whether the TTL value is absolute or relative. The absolute TTL value defines the time to live in terms of seconds since 00:00:00 UTC, January 1st 1970. A relative TTL specifies the time to live in terms of the number of seconds elapsed since the value was obtained by the client from the Handle System.

<timestamp>
An 8-byte (long) integer that records the last time the value was updated at the primary server that manages the handle value. The field contains elapsed time since 00:00:00 UTC, January 1970 in milliseconds. The choice of milliseconds is to avoid potential collision when updating the value.

<reference>
A 4-byte integer followed by a list of references to other handle values. The integer specifies the number of references in the list. Each reference in the list refers to another handle value in terms of a UTF8-string and a 4-byte integer (where the UTF8-string is the handle name and the integer is the value index). References are generally used to add credentials to the current handle value. For example, a handle value may make itself more trust-worthy by providing a reference to a digital signature signed by a commonly trusted party.

By default, for any handle resolution request, the Handle System returns all the handle values with public-read permission. It is possible for a client to ask for a subset of those values with specific types (e.g. all URLs assigned to the handle). The client may also ask for a specific handle value based on the value index.

Each handle value may be uniquely referenced by the combination of the handle and the value index. Care must be taken when changing the value index, as it may break an existing reference to the handle value. For example, suppose handle X/Y has a value whose index is 1. That value may be referred to as X/Y:1. If the handle administrator changes the value index from 1 to 2, the value reference X/Y:1 will become obsolete, and the value will have to be referred to as X/Y:2.

Value records assigned to any handle may or may not have continuous index numbers. Nor can it be assumed that the index will start with 0 or 1. A handle administrator may assign values with any index as long as each index is unique within the value set.

A handle value may be "privatized" or "disabled" by setting its <permission> field as "authorized-read". This limits read-access to the handle administrator. The "privatized" value can then be used to keep any historical data (on behalf of the handle administrator) without making it public. This approach may also be used to keep any obsolete handle or naming authority from being reused accidentally.

3.2. Pre-defined Handle Data Types

Every handle value must have a data type specified in its <type> field. The Handle System provides a type registration service that allows organizations to register new data types for their applications. Data types are registered as handles under the naming authority "0.TYPE". For example, the handle "0.TYPE/URL" (uniquely) identifies the URL data type used by the Handle System. The associated values then describe, or point to descriptions, of the data type syntax and semantics, e.g., values associated with "0.TYPE/URL" could point to the RFCs which define URL. The Handle System pre-defines a set of data types to carry out the service. For example, HS_ADMIN is a pre-defined data type used to identify handle administrators or administrator groups. HS_SITE is a pre-defined data type used to describe the service interface of any handle system service components. The following sections describe the pre-defined data types used by the Handle System.

3.2.1. Handle Administrator: HS_ADMIN

Each handle has one or more administrators. Any administrative operation (e.g., add, delete or modify handle values) can only be performed by its authorized handle administrator(s). Handle administrators are defined by HS_ADMIN values. Every handle must have at least one HS_ ADMIN value. Handles with more than one administrator will have multiple HS_ADMIN values. HS_ADMIN values are used by the Handle System to authenticate any client as the handle administrator before fulfilling any administration requests.

Naming authorities, as described above, are themselves registered as handles under the pre-defined and reserved naming authority "0.NA". These handles are referred to as naming authority handles. Administrators for any naming authority are so defined by being specified as the administrators of the corresponding naming authority handle. For example, "0.NA/10" is the naming authority handle for the naming authority "10". Hence any administrator for the naming authority handle "0.NA/10" becomes the administrator for the naming authority "10". Naming authority administrators are the only ones who can create handles or sub-naming authorities under the naming authority. A sub-naming authority may define their own administrators to create handles and further levels of sub-naming authorities. Thus the naming authority "10.1045" may have a totally separate group of administrators from its parent naming authority "10".

A handle value of type HS_ADMIN is a handle value whose <type> field is HS_ADMIN and whose <data> field consists of the following entries:

<AdminRef>
A reference to a handle value, the reference consisting of the handle name (a UTF8-string) followed by a 4-byte integer providing the index of the handle value.

<AdminPermission>
A 16-bit bit-mask that defines the permissions granted to the administrator.

An <AdminRef> entry can provide a reference to another handle value that may be used to authenticate the administrator. That referenced handle value may contain the secret key or the public key (or its X.509 certificate [10]) used by the administrator. Alternatively, that referenced value may contain a list of references to yet other handle values, each of which eventually contains the secret key or the public key (or its X.509 certificate) used by the administrator. For example, the <AdminRef> entry may refer to a handle value whose <type> field is DSS_WITH_DES_CBC_SHA and whose <data> field contains a DES secret key[11], to be used in the Cipher Block Chaining (CBC) mode of operation [12, 13]. That handle value can then be used by the handle server to authenticate the client via the handle system authentication protocol, which is explained in section 5.2.

A single handle may contain both the HS_Admin value and the secret or public key value referenced by the <AdminRef> piece of that same HS_Admin value. That is, a handle may hold its own key(s) as well as its own administrator(s). Alternatively, and perhaps more commonly, the

<AdminRef> entry may reference a different handle that contains the secret or public key used by the administrator. This allows a single key to be referenced from the administrative records of many handles. In either case, the handle value containing the secret key should be protected with no-read permission to prevent it from being exposed.

The handle value referred to by the <AdminRef> entry may be of type HS_VLIST, the <data> field of which contains a list of references to other handle values. An HS_VLIST value defines an administrator group in which each reference is a member of the group. Each reference is defined in terms of a <handle>:<index> pair. An administrator group may also contain other administrator groups as its members. This allows administrator groups to be defined in a hierarchical fashion. Care must be taken, however, to avoid cyclic definition of administrators or administrator groups. Multiple levels of administrator groups should be avoided due to their lack of efficiency, but will not be signaled as an error. Client software should be prepared to detect any potential cyclic definition of administrators or <AdminRef> entries that point to non-existent handle values and report the error back to the user.

A handle can have multiple HS_ADMIN values, each of which defines a separate handle administrator. Different administrators can play different roles or be granted different permissions. For example, the naming authority handle "0.NA /10" may have two administrators, one of which may only have permission to create new handles under the naming authority, while the other may have permission to create new sub-naming authorities (e.g. "10.1045"). The permission types are as follows:

Add_NA
This permission allows the naming authority administrator to create new sub-naming authorities.

Delete_NA
This permission allows naming authority administrator to delete an existing sub-naming authority.

Add_Handle
This permission allows naming authority administrator to create new handles under the given naming authority.

Delete_Handle
This permission allows naming authority administrator to delete handles under the given naming authority.

Add_Value
This permission allows handle administrator to add handle values other than HS_ADMIN values. HS_ADMIN values are used to define handle administrators and are managed by a different set of permissions.

Delete_Value
This permission allows handle administrator to delete any handle

values other than HS_ADMIN values.

Modify_Value
This permission allows handle administrator to modify any handle
values other than HS_ADMIN values.

Add_Admin
This permission allows handle administrator to add new administrators,
i.e., HS_ADMIN values.

Delete_Admin
This permission allows handle administrator to delete handle
administrators, i.e., HS_ADMIN values.

Modify_Admin
This permission allows handle administrator to modify handle
administrator in terms of HS_ADMIN values.

LIST_Handle
This permission allows naming authority administrator to list
handles under the naming authority.

LIST_NA
This permission allows naming authority administrator to list
sub-naming authorities under the naming authority.

Authorized_Read
This permission grants handle administrator read-access to all those
handle values with "auth-read" permission. Adminitrator without this
permission will not have access to handle values that require
authentication for read access.

Administrator permissions are encoded in the <AdminPermission> part of
the <data> field of any HS_ADMIN value. Each permission is encoded as a
bit flag. The permission is granted if the flag is set to 1, otherwise
it is set to 0.

Figure 3.2.1 shows an example of HS_ADMIN value that defines an
administrator for the naming authority handle "0.NA/10".

```
      ---------------------------------------------------------------
   -----------------------------------------------------------------  |
 -----------------------------------------------------------------  | |
|                                                                 | | |
|  <index>:       2                                               | | |
|  <type>:        HS_ADMIN                                        | | |
|  <data>:                                                        | | |
|    <AdminRef>:        "0.NA/10": 3                              | | |
|    <AdminPermission>: Add_NA,     Delete_NA,                    | | |
|                       Add_Handle, Delete_Handle,                | | |
|                       Add_Value,  Delete_Value,  Modify_Value,  | | |
|                       Authorized_Read, List_Handle, List_NA     | | |
|                                                                 | | |
|  <TTL>:         24 hours                                        | | |
|  <permission>:  read by all, write by administrator             | | |
|  <reference>:   {Empty}                                         | |-
|                                                                 |-
 -----------------------------------------------------------------
```

Figure 3.2.1. Administrator for the naming authority handle "0.NA/10"

As shown in figure 3.2.1, a naming authority administrator for "10" is identified by an HS_ADMIN value assigned to the naming authority handle "0.NA/10". This administrator may be authenticated based on the handle value "0.NA/10:3", which is another value assigned to this same naming authority handle. This administrator is granted permission to add, delete, or modify sub-naming authorities under "10", as well as add or delete handles directly under the naming authority "10". This administrator may also add, delete, or modify any values assigned to the naming authority handle except for HS_ADMIN values, thus preventing this administrator from adding, deleting, or modifying any other administrators for the naming authority.

HS_ADMIN values are used by handle servers to authenticate clients as handle administrators before fulfilling any administrative requests. The server authenticates a client as the administrator by testing whether the client has possession of the secret key (or the private key) as referenced to by the <AdminRef> entry. The authentication is carried out according to the handle system authentication protocol, as specified later in this document.

HS_ADMIN values may require authentication for read access in order to limit public exposure of the data. Alternatively, the handle value that contains the secret key (as referenced to by the <AdminRef> entry) may have no-read permission set to prevent the key from leaving the server and so exposed during transmission.

3.2.2. Service Site Information: HS_SITE

The Handle System consists of a single distributed global handle service, also known as the Global Handle Registry (GHR), along with many local handle services (LHS), each of which may also be distributed. Every handle service provides the same set of functions for resolving and administering collections of handles. Handle services differ primarily in that each is responsible for a unique set of handles. They are also likely to differ in the selection, number, and configuration of components such as servers used to provide handle resolution and administration, and they are likely to be created and managed by different organizations, each with their own goals and policies.

Each handle service, global or local, may consist of one or more sites and each site may consist of one or more handle servers. Each site is a full functional replication, at least in terms of handle resolution, of all other sites within the service, although each site may have differing numbers of servers and computers. Having multiple sites allows the handle service to distribute load among these sites and avoid a single point of failure.

A site typically consists of a cluster of server computers residing within a local Internet domain. These computers work together to distribute the data storage and processing load at the site. It is possible, although it is not recommended, to compose a site from servers at widely different locations. Further, it is even possible to compose two different sites from the same set of servers.

Each service site is defined by an HS_SITE value. HS_SITE is a pre-defined handle system data type. An HS_SITE value defines a service site by identifying the server computers (e.g., IP addresses) that comprise the site along with their service configurations (e.g., port numbers). HS_SITE values are typically part of naming authority handles. The set of HS_SITE values assigned to a naming authority handle is also called the service information for the naming authority. These values are used by naming authority administrators to manage service configurations. Note that an additional layer of indirection, called a service handle, can be used to allow multiple naming authorities to reference a single set of HS_SITE values. This is described in the Service Handle section below. Clients of the Handle System depend on service information to locate the responsible handle server(s) for their requests. The service information can also be used by clients to authenticate responses from those servers.

An HS_SITE value is a handle value whose <type> field is HS_SITE and whose <data> field consists of the following entries:

<Version>
A 2-byte value that identifies the version number of the HS_SITE data format, to allow backward compatibility over time. The first byte contains the major version number, and the second byte contains the minor version number. This document specifies the version number 1.1.

<ProtocolVersion>
A 2-byte integer that identifies the handle protocol version number. The higher byte of the value identifies the major version number and the lower byte the minor version.

<SerialNumber>
A 2-byte integer that increases by 1 (and eventually returns to 0) each time the value gets changed. It is used in the handle system protocol to synchronize the HS_SITE values between client and server.

<PrimaryMask>

An 8-bit mask that identifies the primary site(s) of the handle service. The first bit of the octet is the <MultiPrimary> bit. It indicates whether the handle service has multiple primary sites. The second bit of the octet is the <PrimarySite> bit. It indicates whether the HS_SITE value defines a primary site. A primary site is one that supports administrative operations for its handles. A <MultiPrimary> entry with zero value indicates that the handle service has a single primary site and all handle administration has to be done at that site. A non-zero <MultiPrimary> entry indicates that the handle service has multiple primary sites. Each primary site may be used to administrate a subset of handles managed by the handle service. Any handle under such service must identify its primary site using an HS_PRIMARY value. The HS_PRIMARY value is a handle value whose <type> is HS_PRIMARY and whose <data> contains a reference to the HS_SITE value that defines the primary site for that handle.

<HashOption>
An 8-bit octet that identifies the hash option used by the service site to distribute handles among its servers. Valid options include HASH_BY_HANDLE, HASH_BY_NA, or HASH_BY_LOCAL, which indicate whether the hash operation should be applied to the entire handle, the naming authority portion of the handle, or the local name portion of the handle, respectively. The standard MD5 hashing algorithm [14] is used by each service site to distribute handles among its servers.

<HashFilter>
An UTF8-string entry that is reserved for future extension.

<AttributeList>
A 4-byte integer followed by a list of UTF8-string pairs. The integer indicates the number of UTF8-string pairs that follow. Each UTF8-string pair is an <attribute>:<value> pair that is used to describe the service site. For example, if the <attribute> is "Description", the <value> will contain a natural language description of the service site. Other <attribute> types may be defined in the future to further distinguish each service site.

<NumOfServer>
A 4-byte integer that defines the number of servers in the service site. The entry is followed by a list of <ServerRecord>s, each of which defines a handle server that is part of the site. The <ServerRecord> is defined as follows:

```
<ServerRecord> ::= <ServerID>
                   <Address>
                   <PublicKeyRecord>
                   <ServiceInterface>
```
where

<ServerID>
a 4-byte unsigned integer that uniquely identifies the server process

at the service site. <ServerID> does not have to begin with 1 and
might not be consecutive. It is defined for administration purpose
but not used in the hash-operation (described below) to locate a
handle server. Note that there can be multiple servers residenting on
any given computer, each with a different <ServerID>.

<Address>
the 16-byte IPv6 [15, 16] address of the handle server.

```
<PublicKeyRecord> ::= <RecordType>
                      <KeyRecord>
```

here <RecordType> is an UTF-8 String that identifies a registered
handle data type used to store the key in <KeyRecord>. <KeyRecord>
contains the public key of the handle server, which may be used
by clients to authenticate any server response.

```
<ServiceInterface> ::= <InterfaceCounter>
                       *[ <ServiceType>
                          <TransmissionProtocol>
                          <PortNumber> ]
```

a 4-byte integer followed by an array of triplets of <ServiceType,
TransmissionProtocol, PortNumber>. The 4-byte integer specifies the
number of triplets. Each triplet lists a service interface provided
by the handle server. For each triplet, the <ServiceType> is an octet
as a bit mask that specifies whether the interface is for handle
resolution (bit 0), handle administration (bit 1), or both. The
<TransmissionProtocol> is also an octet as a bit mask that specifies
the transmission protocol, including TCP (bit 0), UDP (bit 1), and
HTTP (bit 2). The <PortNumber> is a 4-byte unsigned integer that
specifies the port number used by the interface. The default port
number for handle resolution/administration is 2641.

The following figure gives an example of an HS_SITE value assigned to
the naming authority handle "0.NA/10". The value describes the primary
service site that manages all handles under the naming authority "10".

```
    ------------------------------------------------------------
  ------------------------------------------------------------  |
 ------------------------------------------------------------  | |
|                                                            | | |
| <index>:       2                                           | | |
| <type>:        HS_SITE                                     | | |
| <data>:                                                    | | |
|    Version:           1.1                                  | | |
|    ProtocolVersion:   4.0                                  | | |
|    SerialNumber:      1                                    | | |
|    PrimaryMask:                                            | | |
|        MultiPrimary:    FALSE                              | | |
|        PrimarySite:     TRUE                               | | |
|    HashOption:        HASH_BY_HANDLE                         | | |
|    HashFilter:        {Empty}                                | | |
|    AttributeList:     0    {followed by no attributes}       | | |
|    NumOfServer:       3                                      | | |
|         {followed by a list of <ServerRecord>}               | | |
|                                                              | | |
|        ------------------------------------------            | | |
|       ------------------------------------------ |           | | |
|      ------------------------------------------ ||           | | |
|     | ServerID:        1                       |||           | | |
|     | Address:         ::132.151.1.155         |||           | | |
|     | PublicKeyRecord: HS_DSAKEY, iQCuR2R...   |||           | | |
|     | ServiceInterface                         |||           | | |
|     |    ServiceType:          Resolution_Only |||           | | |
|     |    TransmissionProtocol: TCP & UDP       |||           | | |
|     |    PortNumber:           2641            |||           | | |
|     |                                          |||           | | |
|     |    ServiceType:          Admin only      |||           | | |
|     |    TransmissionProtocol: TCP             ||            | | |
|     |    PortNumber:           2642            |             | | |
|      ------------------------------------------              | | |
|                                                              | | |
|  <TTL>:        24 hours                                      | | |
|  <permission>: read by all, write by administrator           | | |
|  <reference>:  {empty}                                       | |-
|                                                              |-
 -----------------------------------------------------------
```

Fig. 3.2.2. The primary service site for the naming authority "10"

Figure 3.2.2 defines a handle system service site in terms of an HS_SITE value. The value is assigned to the naming authority handle "0.NA/10". According to the <PrimaryMask> entry, it is the only primary site for the handle service and so manages all handles under the naming authority. The site consists of three handle servers, as indicated by the <NumOfServer> entry. These servers provide handle resolution and administration service for all handles under "10". The first server record (ServerID 0) shows two service interfaces, one for handle resolution and the other for handle administration, each with its own port.

Each server of a service site is responsible for a subset of handles managed by that site. Clients can find the responsible server by performing a hash-operation that first convert all of the ASCII characters in the handle to upper-case, then apply the MD5 hashing on the converted handle string (according to the <HashOption> specified in the HS_SITE value). The result is a 16-byte integer that will be divided by the number of servers (as specified in the <NumOfServer> of the HS_SITE value). The remainder is the sequence number (starting with zero) of the <ServerRecord> as listed in the HS_SITE value. From the server record, clients can find the IP address of the responsible server, and select their preferred transmission protocol for their request.

3.2.3. Service Handle: HS_SERV

The configuration of any given handle service is characterized by the collection of HS_SITE values which define the service, as described above. These HS_SITE values may be assigned directly to the relevant naming authority handle(s), or an additional level of indirection may be introduced through the use of service handles, which may be thought of as names for handle services. Service handles are intended to contain a list of HS_SITE values and may be referenced from naming authority handles via the HS_SERV pre-defined data type. This mechanism allows changes to handle service configurations, e.g., adding a new site, to be made in one place, the service handle, rather than in each naming authority handle, an administrative advantage in those cases in which multiple naming authorities share a single handle service. The mechanism may also be generalized to provide referral from one handle service to another for whatever reason.

An HS_SERV value is a handle value whose <type> field is HS_SERV and whose <data> field contains the service handle. Service handles can be registered under the reserved naming authority "0.SERV" that is managed by the Global Handle Registry. For example, a service handle "0.SERV/123" may be created as the service handle that maintains the service information for the handle service that is responsible for handles under the naming authority "123" and its sub-naming authorities. HS_SERV values are typically assigned to naming authority handles to refer clients to the responsible handle service. A naming authority handle may have no more than one HS_SERV value assigned to it, otherwise it is an error. If a naming authority handle has both a list of HS_SITE values and an HS_SERV value, the HS_SITE values should be used as the service information for the naming authority.

The use of service handles raises several special considerations. Multiple levels of service handle redirection should be avoided due to their lack of efficiency, but will not be signaled as an error. Client software should be prepared to detect any looped reference of service handles or HS_SERV values that point to non-existent service handles and return an error condition back to the user.

3.2.4. Handle Alias: HS_ALIAS

To support multiple names for the same object, the Handle System provides the pre-defined data type HS_ALIAS. An HS_ALIAS value simply provides a reference to another handle. A handle that has an HS_ALIAS value should not have any additional values other than HS_ADMIN (for administration). This is necessary to prevent any inconsistencies between a handle and its aliases.

Thus, when resolving a handle, a client may get back an HS_ALIAS value. This indicates that the handle in question is a handle alias. The client may then retry the query against the handle specified in the HS_ALIAS value until final results are obtained. The use of aliases introduce a number of special considerations, e.g., multiple levels of aliases should be avoided, for the sake of efficiency, but are not signaled as an error. Alias loops and aliases that point to non-existent handles should be caught and error conditions passed back to the client.

One potential use of handle aliases would be in the transfer of ownership of the underlying resources. When a resource identified by a handle transfers from one organization to another, a new handle for the resource may be created. To avoid inconsistency and broken references, the handle used before the ownership transfer may be changed to a handle alias and its HS_ALIAS value pointed to the newly created handle.

3.2.5. Primary Site: HS_PRIMARY

HS_PRIMARY is a pre-defined data type used to designate a primary service site. If a handle service supports multiple primary service sites, each handle managed by the handle service must specify its primary service site via an HS_PRIMARY value. An HS_PRIMARY value is a handle value whose <type> is HS_PRIMARY and whose <data> contains a reference to the HS_SITE value that defines the primary service site.

There can be at most one HS_PRIMARY value assigned to any handle, otherwise it is an error. A handle with no HS_PRIMARY value but managed by a handle service with multiple primary service sites is an error. Handles managed by a handle service with a single primary service site do not need HS_PRIMARY values and any such values are ignored.

3.2.6. Handle Value List: HS_VLIST

HS_VLIST is a pre-defined data type that allows a handle value to make reference to a list of other handle values. An HS_VLIST value is a handle value whose <type> is HS_VLIST and whose <data> consists of a 4-byte integer followed by a list of references to other handle values. The integer specifies the number of references in the list. The references may point to handle values of the same handle or to handle values of one or more other handles. Each reference is encoded as a UTF8-string followed by a 4-byte integer, which identifies the handle name and index value respectively.

HS_VLIST values may be used to define administrator groups for handles. In that case, each reference in the HS_VLIST defines a member of the administrator group and the HS_VLIST value can be used to make reference to the group as a whole. Client software must be careful, however, to avoid cyclic definition of value references. Applications must keep track of the levels of reference to prevent infinite loops.

4. Handle System Service Model

The Handle System provides a distributed global name service via its service components, which consist of a single distributed Global Handle Registry (GHR) and many local handle services (registered underneath the GHR). These service components provide the name service (both resolution and administration) on behalf of handle system client components. Handle system client components may also choose to use handle system middle-ware components (e.g., the handle system caching service) for efficiency. This section describes these components and their relationship to each other.

4.1. Handle System Service Components

The Handle System defines a hierarchical service model. At the top level is the single distributed global handle service, also known as the Global Handle Registry (GHR). Underneath the GHR, there are many local handle services (LHS). Each handle service, global or local, may be registered with the GHR to manage handles under a set of naming authorities and will provide resolution and administration service for these handles. The GHR manages all the naming authorities through the management of naming authority handles (i.e., handles under the naming authority "0.NA"). Each naming authority handle maintains the service information (in terms of HS_SITE values) that describes the local handle service (or the GHR) that is responsible for handles under that naming authority. The service information is defined in terms of a list of HS_SITE values, as described in section 3.2, and enables clients to interact with the relevant handle server in order to resolve or administer those handles. Note that a local handle service may further refer to other local handle services in response to any service requests. This allows the local handle service to distribute its service in a hierarchical fashion underneath the GHR.

Handle system service components are scalable and extensible to accommodate their service loads. Any handle service, global or local, may consist of multiple service sites, replicating each other. Each service site may also consist of a cluster of computers working together to serve its respective namespace. Having multiple service sites avoids having a single point of failure, and allows load balancing among these sites. Using multiple computers at any service site distributes the workload and allows less powerful computers to be utilized for the name service.

4.1.1. Global Handle Registry (GHR)

The Global Handle Registry (GHR) is used to manage naming authority handles and to provide service information for every naming authority under the Handle System. The GHR may also be used to manage and provide resolution service to non naming authority handles. Unlike any local handle service, which is typically responsible for handles under one or a few naming authorities, the GHR is used to register and provide service information for the entire handle system namespace. Every naming authority handle under the Handle System is managed by the GHR. A naming authority handle provides the service information for the handle service that manages all handles under the naming authority. The service information may be provided in terms of a set of HS_SITE values, or a HS_SERV value as described above. The GHR may consist of multiple service sites, each described in terms of an HS_SITE value. In the case of GHR these HS_SITE values are specifically assigned to the designated naming authority handle "0.NA/0.NA", or root handle. Recall that every handle under the naming authority "0.NA" is a naming authority handle, The handle "0.NA/0.NA" is the naming authority handle that manages all the root-level naming authorities and also provides service configuration information for the Global Handle Registry.

The GHR can be configured to consist of multiple primary service sites. In this case, root-level naming authorities may be created or managed at any one of these sites. The site where a naming authority is created is called the primary service site for the naming authority. All other service sites are secondary service sites for the naming authority. Sub-naming authorities may only be created at the primary service site of its parent naming authority. Any naming authority administration (e.g., add/delete a sub-naming authority) can only be done at the primary service site for the naming authority handle. The primary service site is responsible to inform every secondary service sites for any updates that need to be replicated. Conflicts should be caught when multiple service sites attempt to create the same root-level naming authority.

In order to communicate with the GHR, handle clients need its service information (i.e., the set of HS_SITE values that describes the GHR service sites). This may be distributed with the client software, or obtained from some other secure source (e.g., postal mail, secure web site, etc.). When clients find that their version of GHR service information has expired (according to its TTL), they may update their GHR service information by resolving the naming authority handle "0.NA/0.NA". The GHR must be managed in such a way that clients with expired GHR service information can still access the service and get the update.

Fig. 4.1.1 shows the GHR service information in terms of a set of HS_SITE values, one of which is shown in detail. These values are assigned to the naming authority handle "0.NA/0.NA" whose administrator also manages all the root level naming authorities.

```
        --------------------------------------------------------
    ----------------------------------------------------------  |
 -----------------------------------------------------------  | |
|                                                           | | |
|  <index>:      3                                            | | |
|  <type>:       HS_SITE                                      | | |
|  <data>:                                                    | | |
|    Version:          1                                      | | |
|    ProtocolVersion:  4.0                                    | | |
|    SerialNumber:     1                                      | | |
|    PrimaryMask:                                             | | |
|            MultiPrimary:    TRUE                            | | |
|            PrimarySite:     TRUE                            | | |
|    HashOption:       HASH_BY_HANDLE                         | | |
|    HashFilter:       {Empty}                                | | |
|    AttributeList:    1                                      | | |
|        Description:  Service site at US East Coast          | | |
|    NumOfServer:      3                                      | | |
|                                                             | | |
|        ------------------------------------------           | | |
|       ------------------------------------------ |          | | |
|      ------------------------------------------ ||          | | |
|     | ServerID:        1                        |||          | | |
|     | Address:         ::132.151.2.150          |||          | | |
|     | PublicKeyRecord: HS_DSAKEY, iQCuR2Rnw... |||          | | |
|     | ServiceInterface                          |||          | | |
|     |    ServiceType:       Resolution & Admin |||          | | |
|     |    TransmissionProtocol: TCP & UDP       ||           | | |
|     |    PortNumber:           2641            |            | | |
|      ------------------------------------------             | | |
|                                                             | | |
|  <TTL>:        24 hours                                     | | |
|  <permission>: read by all, write by administrator          | | |
|  <reference>:  {empty}                                      | |-
|                                                             |-
 -----------------------------------------------------------
```

Figure 4.1.1.   GHR service information

The GHR and its service information provide an entry point for any handle system service request. Since the GHR manages all naming authority handles, it can provide clients with the service information for any naming authority upon request. If, for example, a client trying to resolve the handle "10.1045" does not know which handle service to query, it can begin by querying the GHR for the naming authority handle, i.e., "0.NA/10". The GHR will return the service information for the naming authority "10", enabling the client to choose a service site and send the handle query to the responsible server at that site.

4.1.2. Local Handle Service (LHS)

Local Handle Services (LHS) manage handles under given sets of naming authorities. Each naming authority defines a "local" namespace that consists of all of the handles under the naming authority. A naming

authority is "homed" at an LHS if all handles under the naming
authority are managed by that LHS. An LHS may be home to multiple
naming authorities, but a naming authority may only be "homed" at one
LHS. A naming authority may be homed at the GHR. Note that a local
handle service is not a "local" service in terms of any network
topology, but is called a local handle service because it is typically
home to a restricted, or local, namespace.

As with the GHR, any LHS may consist of many service sites with each
site described by an HS_SITE value. The set of HS_SITE values for any
LHS may be assigned to a single service handle or to the relevant
naming authority handle(s). Fig. 4.1.2 shows an example of HS_SITE
values for an LHS. These HS_SITE values are assigned to the naming
authority handle "0.NA/10", indicating that the naming authority "10" is
homed at the local handle service described by these HS_SITE values.
Clients may query the GHR to obtain this service information in order
to communicate with this local handle service. Administrators for the
naming authority handle are responsible for managing the service
information and keeping it up to date.

```
      ----------------------------------------------------------
    ----------------------------------------------------------  |
 ----------------------------------------------------------  | |
|                                                           | | |
|  <index>:      3                                          | | |
|  <type>:       HS_SITE                                    | | |
|  <data>:                                                  | | |
|    Version:          1                                    | | |
|    ProtocolVersion:  4.0                                  | | |
|    SerialNumber:     1                                    | | |
|    PrimaryMask:                                           | | |
|            MultiPrimary:   FALSE                          | | |
|            PrimarySite:    TRUE                           | | |
|    HashOption:       HASH_BY_LOCALNAME                    | | |
|    HashFilter:       {Empty}                              | | |
|    AttributeList:    1                                    | | |
|        Description:  Local Service for "10"               | | |
|    NumOfServer:      2                                    | | |
|                                                           | | |
|        -----------------------------------------          | | |
|       ----------------------------------------- |         | | |
|     | ServerID:        1                       ||         | | |
|     | Address:         ::132.151.3.150         ||         | | |
|     | PublicKeyRecord: HS_DSAKEY, iQCuR2R...   ||         | | |
|     | ServiceInteface:                         ||         | | |
|     |    ServiceType:     Resolution & Admin   ||         | | |
|     |    TransmissionProtocol:     TCP & UDP   ||         | | |
|     |    PortNumber:               2641        |'         | | |
|      -----------------------------------------'           | | |
|                                                           | | |
|  <TTL>:        24 hours                                 | | |
|  <permission>: read by all, write by administrator      | | |
|  <reference>:  {empty}                                  | |-
|                                                         |-
 ---------------------------------------------------------

           Figure 4.1.2. LHS service information
```

4.2. Handle System Middle-Ware Components

Handle system middle-ware components currently include handle system caching services and handle system proxy servers. These handle system middle-ware components are clients to global or local handle services, but servers to handle system clients. Handle system middle-ware components may be used to provide additional service interfaces to the basic handle service. For example, a handle system caching service may be utilized to share resolution results within a local community. A handle system proxy server could be used to bypass organizational firewalls, and accept service requests in terms of the HTTP protocol.

4.2.1. Handle System Caching Service

Handle system caching service can be used to reduce the network traffic between handle system clients and servers. Caching handle data, including service information for any local handle service allows subsequent requests going through the cache to be answered using the information obtained from earlier queries.

Every handle value contains a <TTL> (Time to Live) field that tells a caching server how long the value should be cached before an update is required. A zero-value TTL indicates that the value can only be used for the transaction in progress, and should not be cached. A caching server may obtain its data directly from the responsible handle service component, or from another caching service which eventually gets its data from the handle service.

A caching service is defined in terms of an HS_SITE value that consists of multiple caching servers. Clients can direct their request to the responsible caching server within the caching service by using the hashing algorithm defined in the HS_SITE value.

Caching services are not part of any handle system administration or authentication hierarchy. The handle system protocol does not authenticate any response from a caching service. Use of a caching service is a client option, and the client may have to rely on the caching service to authenticate any service response from handle system service components. Clients are responsible to set up their own trust relationship with the caching service they select.

4.2.2. Handle System Proxy Servers

Handle system proxy servers can be used to enable handle resolution via other Internet application protocols. For example, CNRI has built and made available a handle system to HTTP proxy server that will listen to handle resolution requests in terms of HTTP. The current DNS address for the proxy server is "hdl.handle.net". Thus, the handle " ncstrl.vatech_cs/tr-93-35" can be resolved as the URL "http://hdl.handle.net/ncstrl.vatech_cs/tr-93-35" from any web browser. In this case, the URL is sent to the proxy server as an HTTP request. The proxy server will query the handle system for the handle data and return the results to the client as an HTTP response.

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 the selection or use of proxy servers should be carefully evaluated.

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 to set up their own trust relationship with the proxy server they select.

4.3. Handle System Client Components

Handle system client components are end-user software applications that receive service from the Handle System. Depending on configuration, a handle system client component may talk to handle system service components directly, or obtain service via handle system middle-ware components, such as a handle system caching service.

When a client component sends a request to a handle system service component directly, the response from the component may be the final answer to the request, or it may be a referral to another handle system component. Service referrals are returned in terms of service information (i.e., HS_SITE values) or service names (i.e., HS_SERV values). If a service referral is returned, the client component will need to follow the referral in order to complete the transaction.

Client components may also be configured to fill their requests from handle system middle-ware components. The middle-ware component will then be responsible for getting the final result of any client request and returning it to the client. Unlike the handle system service components, middle-ware components will only return final result regarding their client's request. No service referral should be returned from handle system middle-ware components.

Handle system client components should be developed according to the "Handle System Protocol Specification" [8] or the "Handle System Application Programming Interface (API) Specification" [17], which are specified in separate documents. The Handle System API defines a high-level application programming interface on top of the basic Handle System protocol, and allows common client modules to be developed and shared among various applications.

Various handle system client components may be developed for various applications. The CNRI Handle System Resolver [18], which can be integrated with web browsers, is one such client component. It extends web browsers (e.g. Netscape or Microsoft Internet Explorer) such that handles can be resolved directly as Uniform Resource Identifiers (URIs) under the "hdl:" scheme. The Grail web browser [21], a freely downloadable software developed in Python [22], also supports the "hdl:" URI scheme and will resolve handles accordingly. For example, the handle "10.1045/july95-arms" may be resolved by entering its handle URI as "hdl:10.1045/july95-arms" into any of these resolver enabled browsers. The "Handle System URI Syntax" [19] will specify the syntax and semantics for handles expressed under the "hdl:" URI scheme.

5. Handle System Operation Model

Handle System operations can be categorized into resolution and administration. Handle resolution is achieved by client submitting a handle query to a handle service and receiving back the set of handle values from the responsible handle server. Handle administration allows clients to manage handles, including adding and deleting handles, or updating their values, over the public Internet. It also deals with naming authority administration via naming authority handles. Both types of operations may require authentication of the client via the handle system authentication protocol, described below. Whether authentication is required or not depends on the kind of operation involved and the permission settings of the relevant handle value(s), as well as the policies deployed by the responsible service components.

The handle system protocol governs the syntax and semantics of each message transmitted between handle system clients and server components. This section provides a high level overview of the protocol and the role that each message plays during a service operation. The exact programmatic details of these messages (i.e. their byte layout or syntax) are specified in a separate document [2].

5.1. Handle System Service Request and Response

The Handle System provides its service in response to client service requests. A client may send a request to any handle server to provoke a response. The response either provides an answer to the request, or a status code with associated information that either refers the request to another service component, asks for client authentication, or signals some error status.

Each handle under the Handle System is managed by its home service. The naming authority handle provides the home service information (in terms of HS_SERV or HS_SITE values) for all the handles under the naming authority. Any handle service request must be directed to its home service. Clients may find the home service by querying the Global Handle Registry (GHR) for the corresponding naming authority handle. Alternatively, this information may be found in a local cache or even be part of a local client configuration. Given the service information, clients select a service site and communicate with the responsible handle server at the site.

To resolve the handle "ncstrl.vatech_cs/te-93-35", for example, a client would need to know the home service for the naming authority "ncstrl.vatech_cs". The home service can be obtained by querying the GHR for the corresponding naming authority handle. In this case, the naming authority handle is "0.NA/ncstrl.vatech_cs". GHR will return the service information in terms of the HS_SITE values assigned to the naming authority handle. From that service information, clients can pick a service site, find the responsible server within the site, and send the resolution request to that server.

Clients may require digital signatures attached to any response from a handle server in order to authenticate the response. The signature is generated using the server's private key. Clients may verify the signature using the public key available in the service information (refer to the <PublicKey> element shown in Figure 3.2.2).

A communication session may be established between the client and server to manage a request that requires multiple interactions. The session may also be used to share a TCP connection or client authentication among multiple service requests. Each session is identified by a session ID managed by the server. The client may provide a session key which would be a secret key shared by the client and server. The session key can be a nonce (i.e., a random array of octets) generated by the client and sent to the handle server, encrypted by the server's public key. Using the session key, the session may provide data integrity service by attaching the Message Authentication Code (MAC) to each message. It may also provide data confidentiality service by encrypting each message sent back and forth between the client and server.

The following diagram shows a handle resolution process in terms of the messages transmitted between a handle system client and handle system service components. In this case, the client is trying to resolve the handle "ncstrl.vatech_cs/tr-93-35", does not know the home service, and so must begin by querying the GHR.

```
[HS Client]  ----------------------------> [Global Handle Registry]
```

```
              1. request for service
                 information for the
                 naming authority handle
                 "0.NA/ncstrl.vatech_cs"


[HS Client]  ---------------------------> [Global Handle Registry]
              2. service information for
                 the naming authority
                 handle "0.NA/ncstrl.vatech_cs"



[HS Client]  -------------------------------> [Local Handle Service]
              3. query for the handle
                 "ncstrl.vatech_cs/tr-93-35"
                 against the responsible
                 local handle server


  ... ...
  (optional client authentication, depending on the service request)
  ... ...



[HS Client]  ------------------------------> [Local Handle Service]
               4. query result from the server
                + signature from the server (optional)
```

Figure 5.1   Handle resolution example

In Figure 5.1, the client is configured to communicate with the GHR for any handle service. In this case, the client first sends a request to GHR to find the home service for the handle in question. GHR returns the service information assigned to the corresponding naming authority handle. The service information typically directs the client to the LHS that manages all the handles under that naming authority. From the service information, a client can find the responsible server at their choice of service site and send the handle query to that server. That server may set up a session to authenticate the client if the queried data requires authentication. Otherwise, the server will simply send the data back to the client. The server may provide a signature as part of its response if so requested by the client.

The above procedure assumes that the client already has the GHR service information. That information was likely obtained from the client software distribution and kept up to date by queries against the naming authority handle "0.NA/0.NA", which is the naming authority handle for all the root level naming authorities. "0.NA/0.NA" also maintains the public key that may be used to authenticate the GHR service information.

Note that a client may cache the service information for any naming authority handle so that subsequent queries for handles under the same naming authority may reuse the service information and bypass the first two steps shown in Figure 5.1. Clients may also elect to communicate directly with a caching service or proxy server for any handle resolution operations. In that case, clients will not need to query for any home service information and instead will query the caching or proxy server directly for the handle. The caching or proxy server will then act as the [HS Client] in Figure 5.1 before finally returning the end query result to the client.

It is also possible for clients to communicate with any given LHS directly with their requests. This will give them quick response for handles homed at the local handle service. For handles not homed at the LHS, clients are referred to the GHR and will continue from there.

5.2. Handle System Authentication Protocol

The Handle System supports handle administration and access control of handle values over the Internet. The handle system authentication protocol is used by handle server to authenticate clients as authorized administrators by both administration and resolution purposes. The protocol does not perform server authentication. However, client may authenticate any response from the server by asking the server to sign any response from the server with server's digital signature.

Client authentication is generally required for any handle administration requests, including adding, deleting or modifying handle values. Adding or deleting handles or sub-naming authorities are considered administration of the corresponding naming authority handle and can only be performed by the authorized naming authority (handle) administrator(s).

By default, the Handle System authenticates clients using a challenge response protocol. That is, after receiving a client's request, the server issues a challenge to the client if authentication is necessary. The challenge consists of a nonce generated by the handle server, concatenated to the MD5 hash of the client's request. To be authenticated as the administrator, the client has to send back a response message that demonstrates procession of the administrator's secret. The secret may be the secret key or the private key of the administrator. When secret key is used by the administrator, the response message will consist of the Message Authentication Code (MAC) over the server's challenge, generated using the administrator's secret key. If public key is used, the response message will contain the digital signature over the challenge, generated using the administrator's private key. This challenge-response message allows the server to authenticate the client as the handle administrator as claimed by the client. Upon successful authentication, the server will carry out the client's request as long as the claimed administrator is granted with sufficient permission.

For example, suppose a client sends a request to the responsible handle server to add a new handle value. If authentication is required, the server will return a challenge for the client to authenticate himself as an administrator. If the client processes the private key of some administrator, he can use it to sign the server's challenge and send the signature back to the server. The server will verify the signature in order to authenticate the client as the handle administrator claimed by the client. If the verification fails, a message indicating authentication failure will be sent to the client. Otherwise, the server will further check if the claimed administrator has the permission to add new handle value(s). If so, the server will add the handle value and report success to the client. Otherwise, a permission denied message will be sent instead.

The following control diagram shows a typical authentication process in terms of the messages transmitted between the client and the handle server.

```
  [Client]  --------------------------------->    [Handle Server]
              1. client request
               + client credential (optional)


  [Client]  <---------------------------------  [Handle Server]
              2. session ID
               + server's challenge to client
                 (i.e., nonce + MD5 of client request)


  [Client]  -------------------------------------->   [Handle Server]
              3. session ID
               + response to server's challenge
               + administrator reference
               + encrypted session key (optional)


  [Client]  <--------------------------------------   [Handle Server]
              4. server acknowledgement


        Figure 5.2. Handle system authentication process
```

In Figure 5.2, the client sends a request to the handle server (along with optional credential discussed later). The server decides that client authentication is required and issues a challenge back to the client, along with the session ID. The client authenticates itself as an administrator by answering the challenge using the secret key or the private key of the administrator. The server verifies client's response using the administrator reference from the client. The administrator reference specifies the HS_ADMIN value claimed by the client, as well as the key reference that refers to the handle value that contains the key used by the administrator. The key reference allows the handle server to verify the response (i.e., the MAC or signature over the challenge) from the client. From the HS_ADMIN value, the server can check whether the key reference is one of the administrators defined by the HS_ADMIN value, and confirm that the administrator is granted sufficient permission. A client may send back its administrator reference without specifying the specific HS_ADMIN value. In this case, the handle server should look up all the HS_ADMIN values with sufficient permission and check if the key reference belongs to any of the HS_ADMIN values.

The handle server authenticates the client before fulfilling any client request that requires authentication. Authentication is based on the administrator reference along with the response from the client. The process of authenticating the client varies depending on whether public key or secret key is used by the administrator. It also depends on whether the key reference (as part of the administrator reference) is managed by the same handle server or not.

When a public key is used by the administrator, the response from the client contains the digital signature over the challenge from the server. The server can authenticate the client by verifying the digital signature using the administrator's public key. If secret key is used, the response from the client carries the Message Authenticate Code (MAC) generated using the secret key. The MAC is generated by applying the MD5 hashing over the block of data that is made of the server's challenge concatenated with the secret key. The server may authenticate the client by generating the same MAC using the administrator's secret key and comparing it with the response from the client.

The key reference refers to a handle value that contains the key used by the administrator. If the handle value is managed by the same handle server (e.g., assigned to the same handle referenced by the client's request), the server may acquire the key securely from its local database. If the handle value is managed by another handle server (whether or not within the same service), the server will have to send a verification request to this other handle server, call it the key server, in order to authenticate the client. The verification request to the key server includes both the server's challenge and the client's response. The key server is requested to send back its verification response, signed using the key server's private key. The content of the verification response will depend on the handle value referred to by the key reference. If the key reference refers to a public key used by the administrator, the key server will send back the public key as the verification response. Otherwise, the key server will verify the

client's response on behalf of the requesting server and send back a
verification response indicating whether or not the client's response
matches the server's challenge. The following diagram shows the control
flow of the authentication process where the key reference refers to a
handle value, containing the administrator's public or secret key,
residing on another handle server.

```
  ---------                                    -------------
 |        |   1. client request.              |             |
 |        | -------------------------------->  |             |
 |        |                                   |             |
 |        |   2.  session ID                  |             |
 |        |     + server's challenge          |             |
 | Handle | <-------------------------------  | Handle      |
 | system |                                   | server      |
 | client |   3.  session ID                  | responsible |
 |        |     + response to the challenge   | for client  |
 |        |     + administrator reference     | request     |
 |        |     + encrypted session key       |             |
 |        | --------------------------------> |             |
 |        |                                   |             |
 |        |                                   |             |
 |        |                                   |             |
 |        |   6.  server acknowledgement      |             |
 |        | <-------------------------------  |             |
 |        |                                   |             |
  --------                                     -------------
                                                   |  ^
                                                   |  |
                                   4. Verification |  | 5. verifi-
                                      request      |  |    cation
                                                   |  |    response
                                                   |  |    (signed)
                                                   |  |
                                                   V  |
                                         -------------------------
                                        |                         |
                                        | The handle server (the  |
                                        | key server) referred    |
                                        | to by the administrator |
                                        | reference               |
                                        |                         |
                                         -------------------------
```

Figure 5.3. Authentication process requiring verification
from a second handle server.

Secret key based authentication via a second handle server, i.e., the
key server, provides a convenient way to share a single administrator

password for multiple handles. However, it should not be used to manage highly sensitive handles or handle data. The authentication process itself is expensive and relies on a third party, i.e., the key server, for proper operation. Additionally, the secret key itself is subject to dictionary attack since the key server can not determine whether any verification request is from a legitimate handle server. A handle service may set local policy such that secret key based authentication can only be used if the key reference refers to a handle value managed by the same handle server.

When sending a challenge-response to the server, a client has the option to send a randomly generated session key, encrypted using the server's public key. The session key is a secret key that may be used for subsequent message exchange between the client and the handle server. The client may also ask that all the messages within the communication session to be encrypted using the session key. The session key may also be used to authenticate any message between the client and server by attaching the MAC, generated using the session key, to each message.

Local handle services may define additional local policies for authentication and/or authorization. A handle system client and its service components may also elect to use other Internet authentication mechanisms such as Kerberos [20] or Transport Layer Security protocol[23]. Details of these will be addressed in a separate document.

6. Security Consideration

Handle System security considerations are discussed in the "Handle System Overview" document [1] and that discussion applies equally to this document.

7. Acknowledgement

This work is derived from earlier versions of the Handle System. Design ideas are based on those discussed within the handle system development team, including David Ely, Allison Yu, Jane Euler, Catherine Rey, and Stephanie Nguyen. Their contributions to this work are gratefully acknowledged.

8. Authors' Address

Sam X. Sun
Corporation for National Research Initiatives (CNRI)
1895 Preston White Dr.   Suite 100
Reston, VA 20191
USA

Phone: 703-262-5316
Email: ssun@cnri.reston.va.us

Sean Reilly
Corporation for National Research Initiatives (CNRI)
1895 **Preston White Dr.**   Suite 100
Reston, VA 20191
USA

Phone: 703-620-8990
Email: sreilly@cnri.reston.va.us

Larry Lannom
Corporation for National Research Initiatives (CNRI)
1895 **Preston White Dr.**   Suite 100
Reston, VA 20191
USA

Phone: 703-620-8990
Email: llannom@cnri.reston.va.us

9. References

[1] S. Sun, L. Lannom, "Handle System Overview", IETF draft, http://www.ietf.org/internet-drafts/draft-sun-handle-system-06.txt, work in progress.
[2] P. Mockapetris, "DOMAIN NAMES - CONCEPTS AND FACILITIES", RFC1034, November 1987. http://www.ietf.org/rfc/rfc1034.txt
[3] P. Mockapetris, "DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION", RFC1035, November 1987. http://www.ietf.org/rfc/rfc1035.txt
[4] M. Wahl, T. Howes, and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, http://www.ietf.org/rfc/rfc2251.txt
[5] D. Crocker, Ed., P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, http://www.ietf.org/rfc/rfc2234.txt
[6] F. Yergeau, "UTF-8, A Transform Format for Unicode and ISO10646", RFC2044, October 1996. http://www.ietf.org/rfc/rfc2044.txt
[7] The Unicode Consortium, "The Unicode Standard, Version 2.0", Addison-Wesley Developers Press, 1996. ISBN 0-201-48345-9
[8] S. Sun, S. Reilly, L. Lannom, J. She, "Handle System Protocol Specification", IETF draft, http://www.ietf.org/internet-drafts/draft-sun-handle-system-protocol-01.txt, work in progress.
[9] T. Berners-Lee, L. Masinter, M. McCahill, et al., "Uniform Resource Locators (URL)", RFC1738, http://www.ietf.org/rfc/rfc1738.txt
[10] ITU-T Rec. X.509, "The Directory: Authentication Framework", 1993.
[11] Federal Information Processing Standards Publication (FIPS PUB) 46-1, Data Encryption Standard, Reaffirmed 1988 January 22 (supersedes FIPS PUB 46, 1977 January 15).
[12] Federal Information Processing Standards Publication (FIPS PUB) 81, DES Modes of Operation, 1980 December 2.
[13] D. Balenson, "Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers", RFC 1423, February 1993, http://www.ietf.org/rfc/rfc1423.txt
[14] R. Rivest, " The MD5 Message-Digest Algorithm", RFC 1321, April

1992, http://www.ietf.org/rfc/rfc1321.txt
[15] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, http://www.ietf.org/rfc/rfc1883.txt
[16] R. Hinden, S. Deering, "IP Version 6 Addressing Architecture", RFC1884, http://www.ietf.org/rfc/rfc1884.txt
[17] "Handle System Application Programming Interface (API) Specification", work in progress.
[18] CNRI Handle System Resolver, http://www.handle.net/resolver
[19] "Handle System URI Syntax", work in progress.
[20] J. Kohl, and C. Neuman, "The Kerberos Network Authentication Service (V5)", RFC1510, http://www.ietf.org/rfc/rfc1510.txt
[21] Grail browser home page, http://grail.cnri.reston.va.us/grail/
[22] Python language website, http://www.python.org/
[23] T. Dierks, C. Allen, "The TLS Protocol Version 1.0", RFC2246, http://www.ietf.org/rfc/rfc2246.txt