

Network Working Group
Request for Comments: 3651
Category: Informational

S. Sun
S. Reilly
L. Lannom
CNRI
November 2003

Handle System Namespace and Service Definition

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.

IESG Note

Several groups within the IETF and IRTF have discussed the Handle System and its relationship to existing systems of identifiers. The IESG wishes to point out that these discussions have not resulted in IETF consensus on the described Handle System nor on how it might fit into the IETF architecture for identifiers. Though there has been discussion of handles as a form of URI, specifically as a URN, these documents describe an alternate view of how namespaces and identifiers might work on the Internet and include characterizations of existing systems which may not match the IETF consensus view.

Abstract

The Handle System is a general-purpose global name service that allows secured name resolution and administration over the public Internet. This document provides a detailed description of the Handle System namespace, and its data, service, and operation models. The namespace definition specifies the handle syntax and its semantic structure. The data model defines the data structures used by the Handle System protocol and any pre-defined data types for carrying out the handle service. The service model provides definitions of various Handle System components and explains how they work together over the network. Finally, the Handle System operation model describes its service operation in terms of messages transmitted between client and server, and the client authentication process based on the Handle System authentication protocol.

Table of Contents

1.	Introduction	2
2.	Handle System Namespace.	3
3.	Handle System Data Model	4
3.1.	Handle Value Set	4
3.2.	Pre-defined Handle Data Types.	9
3.2.1.	Handle Administrator: HS_ADMIN	10
3.2.2.	Service Site Information: HS_SITE.	14
3.2.3.	Naming Authority Delegation Service: HS_NA_DELEGATE	19
3.2.4.	Service Handle: HS_SERV.	20
3.2.5.	Alias Handle: HS_ALIAS	21
3.2.6.	Primary Site: HS_PRIMARY	21
3.2.7.	Handle Value List: HS_VLIST.	22
4.	Handle System Service Model.	22
4.1.	Handle System Service Components	23
4.1.1.	Global Handle Registry (GHR)	23
4.1.2.	Local Handle Service (LHS)	26
4.2.	Handle System Middle-Ware Components	27
4.2.1.	Handle System Caching Service.	27
4.2.2.	Handle System Proxy Server	28
4.3.	Handle System Client Components.	28
5.	Handle System Operation Model.	29
5.1.	Handle System Service Request and Response	30
5.2.	Handle System Authentication Protocol.	32
6.	Security Considerations.	37
7.	Acknowledgements	38
8.	References and Bibliography.	38
9.	Authors' Addresses	40
10.	Full Copyright Statement	41

1. Introduction

The Handle System manages handles as globally unique names for Internet resources. It was originally conceived and described in a paper by Robert Kahn and Robert Wilensky [22] in 1995. The Handle System provides a general-purpose global name service that allows handles to be resolved and administrated securely over the public Internet. The Handle System categorizes its service into two categories: 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, and updating handle values.

The document "Handle System Overview" [1] provides an architectural overview of the Handle System, and 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, its data and service model, and its operation model. It assumes that readers are familiar with the basic concepts of the Handle System as described in the overview document.

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

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 characters from the Unicode 2.0 standard [7]. The <NamingAuthority> may use any 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 is 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. For example, "10.1045" is a child naming authority of "10" for the 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 the GHR manages all handles for naming authorities, ASCII characters in naming authorities are treated as case insensitive.

3. Handle System Data Model

The Handle System provides a name-to-value binding service over the public Internet. 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 any 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 be the exact 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. These handle values use a common data structure for its data. For example, each handle value has a unique index number that distinguishes it from other values in the value set. It also has a specific data type that defines the syntax and semantics of the data in its data field. Besides these, each handle value contains a set of administrative information such as TTL and permissions. Figure 3.1 shows the handle

"10.1045/may99-payette" with a set of three handle values. One of these values (with index number set to 1) is shown in detail. (Note that the encoding of the length for each field is not shown in Figure 3.1. Also, the empty <reference> field consists of a 4-byte integer whose value is zero.)

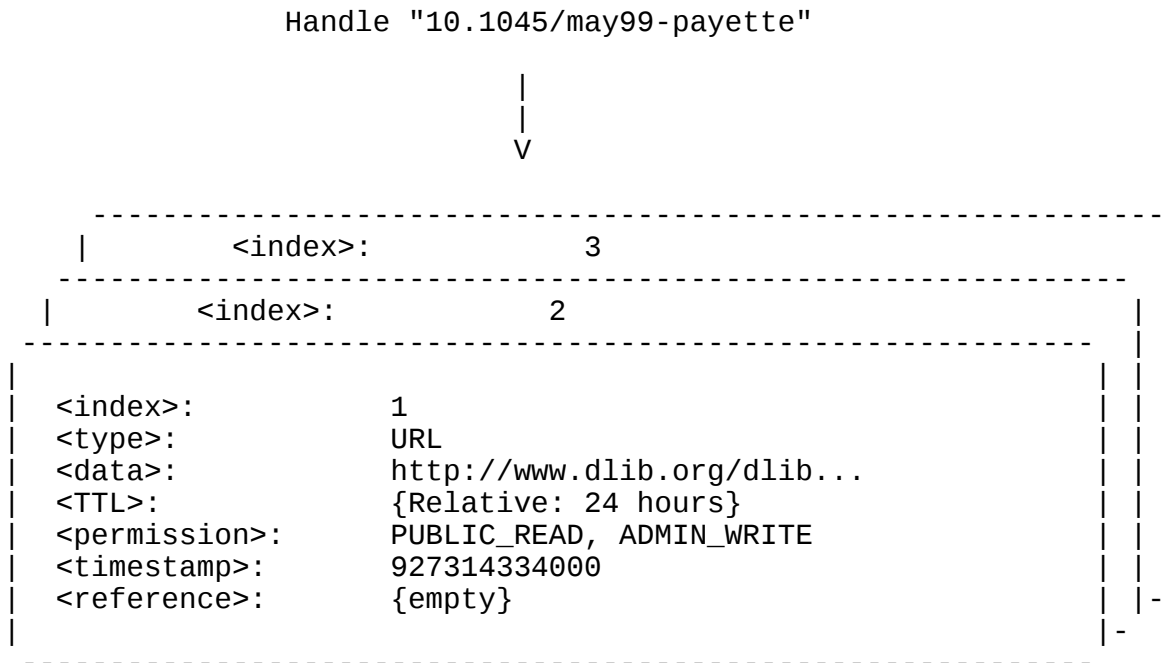


Figure 3.1: Handle "10.1045/may99-payette" and its set of values

In Figure 3.1, it shows a handle value whose its index is set to 1. The data type for the handle value is URL. The URL data as stated in the <data> field is "http://www.dlib.org/dlib...". 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 to be consulted again. The <permission> field grants anyone permission to read, but only the administrator to update the value. The <reference> field is empty. It may contain a list of references to other handle values as credentials for this handle value.

Thus a handle value may be thought of as a record that consists of a group of data fields. Each of these data fields is defined as follows:

<index>

An unsigned 32-bit integer that uniquely identifies a handle value from other handle values.

<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 of data in the next <data> field. The data type may 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 value that explains the syntax and semantics of the data type.

Data types under the Handle System may be hierarchical. Each level of the hierarchy may be named in terms of a UTF8-String with no '.' (0x2E) characters. The '.' character is used to mark the boundary between hierarchy levels. For example, the Handle System data type "a.b" may be considered as a sub-type "b" under the type "a". Similarly, handle values of <type> "a.b.x", "a.b.y" and "a.b.z" may be considered as handle values under the common type hierarchy "a.b".

For any handle values, the UTF8-string in the <type> field may not end with the '.' character. In other words, no Handle System data type should end with the '.' character. However, the '.' character may appear in the end of the <type> parameter in a handle query. This is used to query for all handle values under a common type hierarchy. For example, one may query for all handle values under the type hierarchy "a.b" (e.g., handle values of <type> "a.b.x", "a.b.y" and "a.b.z") by setting the <type> parameter to "a.b.". Note here that the <type> parameter ends with the '.' character. Details of the handle query operation can be found in the Handle System protocol specification [8].

<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 for access control of the handle value. Access control is defined in terms of read, write, and execute

permissions, applicable to either general public or handle administrator(s). Each handle value can have its permission field specified as any combination of the following bits:

PUBLIC_WRITE	(0x01)	permission that allows anyone to modify or delete the handle value.
PUBLIC_READ	(0x02)	permission that allows anyone to read the handle value.
ADMIN_WRITE	(0x04)	permission that allows any handle administrator to update or delete the handle value.
ADMIN_READ	(0x08)_	permission that allows the handle value to be read by any handle administrator with AUTHORITATIVE_READ privilege.
PUBLIC_EXECUTE	(0x10)	permission that allows anyone to execute the program identified by the handle value on the handle host as anonymous user. Because of the security risks this may have brought up, implementations may choose not to support such permission, or provide options so that it can be disabled at deployment.
ADMIN_EXECUTE	(0x20)	permission that allows handle administrator(s) to run the program identified by the handle value on the handle server. The handle server must authenticate the handle administrator before executing the program. The handle administrator must have an established account on the handle server. The execution of the handle value should assume the same privilege as the one given to the account for the handle administrator. Because of the security risks this may have brought up, implementations may choose not to support such permission, or provide options so that it can be disabled at deployment.

Note that a handle value with no `PUBLIC_READ` nor `ADMIN_READ` permission can not leave the handle server. It may be used, for example, to store secret keys for authentication purposes. A handle value with neither `PUBLIC_WRITE` nor `ADMIN_WRITE` permission makes the handle value immutable and cannot be deleted by any handle administrator (via the Handle System protocol).

The administrator for a given handle must specify the permission for each handle value. Implementations may choose `PUBLIC_READ` and `ADMIN_WRITE` as the default 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. It is used to describe 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. Any non-zero TTL is defined in terms of a TTL type (specified in the first octet), followed by the TTL value (the 32-bit unsigned integer that follows the TTL type). 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 any handle server.

<timestamp>

An 8-byte (long) integer that records the last time the value was updated at the server. 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 referring to a digital signature issued by a commonly trusted entity.

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

Each handle value can be uniquely referenced by the combination of the handle and its 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 the 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 reference to X/Y:1 will become obsolete. Any reference to the handle value will have to change to 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 a handle value 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 only. The "privatized" value can then be used to keep any historical data (on behalf of the handle administrator) without exposing it to public. Such 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 can be registered as handles under the naming authority "0.TYPE". For example, the URL data type is registered under the Handle System as the handle "0.TYPE/URL". The handle may have a handle value that refers to RFC1738 [9], an IETF standard document that defines the syntax and semantics of URL.

The Handle System pre-defines a set of data types to carry out the handle service. For example, HS_ADMIN is a pre-defined data type used to describe handle administrators or administrator groups. HS_SITE is a pre-defined data type to describe the service interface of any Handle System service component. The following sections provide detailed descriptions of these pre-defined data types under 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 the handle administrator with adequate privilege. Handle administrators are defined in terms of HS_ADMIN values. Every handle must have at least one HS_ADMIN value that defines its administrator. Each HS_ADMIN value can be used to define a set of handle administrators sharing the same administration privilege. Handles with multiple administrators of different privileges may have multiple HS_ADMIN values. HS_ADMIN values are used by the Handle System to authenticate handle administrators before fulfilling any handle administration request.

Naming authorities, as described above, are themselves registered as handles under the reserved naming authority "0.NA". These handles are referred to as naming authority handles. Administrators for any naming authority are defined 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" is also 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 its own set of administrators to create handles or further levels of sub-naming authorities. For example, the naming authority "10.1045" may have a totally different group of administrators from its parent naming authority "10".

An HS_ADMIN value 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 consists of the handle name (a UTF8-string) followed by a 4-byte unsigned integer for the handle value index. The handle value identifies the set of administrators for the handle.

<AdminPermission>

A 16-bit bit-mask that defines the administration privilege of the set of handle administrators identified by the HS_ADMIN value.

The <AdminRef> entry refers to a handle value that can be used to authenticate the handle administrator. Such handle value is called the handle administrator reference. The handle administrator reference may contain the secret key, public key, or X.509 certificate [10] provided by the handle administrator. For example, the <AdminRef> entry may contain a handle administrator reference

whose <type> field is DSS_WITH_DES_CBC_SHA and whose <data> field contains a DES secret key [11], for use in the Cipher Block Chaining (CBC) mode of operation [12, 13]. The secret key can be used by the handle server to authenticate the handle administrator. For stronger cryptographic algorithm, the handle administrator reference may contain a set of Triple-DES keys [23] and set its <type> to be DES-EDE3-WITH-CBC.

A single handle may be assigned with both the HS_ADMIN value and the handle administrator reference. In other words, the <AdminRef> entry may refer to a handle value assigned to the same handle that has the HS_ADMIN value. In this case, authentication of the handle administrator does not rely on any other handles. Alternatively, the handle administrator reference may be a handle value under a different handle. Thus HS_ADMIN values from different handles may share a common handle administrator reference. This feature allows sharing of handle administrators among different handles. The handle administrator reference contains the secret key, public key, or X.509 certificate provided by the administrator of these handles.

Handle administrator reference may be of type HS_VLIST and has its <data> field contain a list of references to other handle values. Each of these handle values defines a handle administrator reference. The HS_VLIST value defines an administrator group. Each handle administrator reference from the HS_VLIST is a member of the administrator group. Each handle value 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 treat them as an error.

A handle can have multiple HS_ADMIN values, each of which defines a different 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 set of possible permissions for a handle administrator is defined as follows:

Add_Handle (0x0001)

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

Delete_Handle (0x0002)

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

Add_NA (0x0004)

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

Delete_NA (0x0008)

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

Modify_Value (0x0010)

This permission allows handle administrator to modify any 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 (0x0020)

This permission allows handle administrator to delete any handle value other than the HS_ADMIN values.

Add_Value (0x0040)

This permission allows handle administrator to add handle values other than the HS_ADMIN values.

Modify_Admin (0x0080)

This permission allows handle administrator to modify HS_ADMIN values.

Remove_Admin (0x0100)

This permission allows handle administrator to remove HS_ADMIN values.

Add_Admin (0x0200)

This permission allows handle administrator to add new HS_ADMIN values.

Authorized_Read (0x0400)

This permission grants handle administrator read-access to handle values with the ADMIN_READ permission. Administrators without this permission will not have access to handle values that require authentication for read access.

LIST_Handle (0x0800)

This permission allows naming authority administrator to list handles under a given naming authority.

LIST_NA (0x1000)

This permission allows naming authority administrator to list immediate sub-naming authorities under a given naming authority.

Administrator permissions are encoded in the <AdminPermission> entry in 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". In figure 3.2.1, a naming authority administrator is identified by an HS_ADMIN value assigned to the naming authority handle "0.NA/10". The administrator can be authenticated based on the handle value "0.NA/10":3, which is the handle value assigned to the naming authority handle "0.NA/10" and has its index set to 3. The handle value "0.NA/10":3 may contain the secret or public key used by the administrator. The administrator is granted permission to add, delete, or modify sub-naming authorities under "10", and add or delete handles directly under the naming authority. The administrator may also add, delete, or modify any handle values assigned to the naming authority handle except those HS_ADMIN values. In other words, the administrator is not allowed to add, delete, or modify any administrators for the naming authority.

```

-----
-----
<index>:      2
<type>:      HS_ADMIN
<data>:
  <AdminRef>:  "0.NA/10": 3
  <AdminPerm>: Add_NA,      Delete_NA,
               Add_Handle, Delete_Handle,
               Add_Value,  Delete_Value, Modify_Value,
               Authorized_Read, List_Handle, List_NA

<TTL>:      24 hours
<permission>: PUBLIC_READ, ADMIN_WRITE
<reference>: {empty}
-----
-----

```

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

HS_ADMIN values are used by handle servers to authenticate the handle administrator before fulfilling any administrative requests. The server authenticates a client by checking whether the client has possession of the secret key (or the private key) that matches the one in any of the handle administrator references. The authentication is carried out via the Handle System authentication protocol as described later in this document.

HS_ADMIN values may require authentication for read access in order to prevent public exposure of the data. Additionally, the handle administrator reference that contains the administrator's secret key should have neither PUBLIC_READ nor ADMIN_READ permission to prevent the key from leaving the server.

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), and unlimited number of Local Handle Services (LHSs). Each handle service, global or local, may be replicated into multiple service sites. Each service site may consist of multiple server computers. Service requests targeted at any handle service can be distributed into different service sites, and into different server computers within any service site. Such architecture assures that each handle service could have the capacity to manage any large number of handles and handle requests. It also provides ways for each handle service to avoid any single point of failure.

Each handle service, global or local, may provide the same set of functions for resolving and administering its collection of handles. Handle services differ primarily in that each service is responsible for a distinct set of handles. They are also likely to differ in the selection, number, and configuration of their components such as the servers used to provide handle resolution and administration. Different handle services may be created and managed by different organizations. Each of them may have their own goals and policies.

A service 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 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 assigned to naming authority handles. The set of HS_SITE values assigned to a naming authority handle is called the service information for the naming authority.

The service information is managed by the naming authority administrator. It must reflect the configuration of the handle service for the naming authority. 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, as described later in this document (see section 3.2.3). Clients of the Handle System depend on the service information to locate the responsible handle server before they can send their service requests. The service information can also be used by clients to authenticate any service response from the handle server.

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. The version number identifies the data format used by the HS_SITE value. It is defined to allow backward compatibility over time. This document defines the HS_SITE with version number 0.

<ProtocolVersion>

A 2-byte integer value that identifies the handle protocol version. The higher byte of the value identifies the major version and the lower byte the minor version. Details of the Handle System protocol is specified in [8].

<SerialNumber>

A 2-byte integer value that increases by 1 (and may wrap around through 0) each time the HS_SITE 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 is a primary site. A primary site is the 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 handles managed under the handle service. Handles managed by such service may identify its primary sites using an HS_PRIMARY value, as described in section 3.2.5.

<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_NAME (0x00), HASH_BY_LOCAL (0x01), or HASH_BY_HANDLE (0x02). These options indicate whether the hash operation should only be applied to the naming authority portion of the handle, or only the local name portion of the handle, or the entire 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 reserved for future use.

<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. They are used to add literal explanations of the service site. For example, if the <attribute> is "Organization", the <value> should contain a description of the organization hosting the service site. Other <attribute>s may be defined to help distinguish the service sites from each other.

<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 <ServerRecord> defines a handle server that is part of the service site. Each <ServerRecord> consists of the following data fields:

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

where each field is defined as follows:

<ServerID>

A 4-byte unsigned integer that uniquely identifies a server process under the service site. <ServerID>s do not have to begin with 1 and they don't have to be consecutive numbers. They are used to distinguish servers under a service site from each other. Note that there can be multiple servers residing on any given computer, each with a different <ServerID>.

<Address>

The 16-byte IPv6 [15, 16] address of the handle server. Any IPv4 address should be presented as ::::FFFF:xxxx:xxxx (where xxxx:xxxx can be any 4-byte IPv4 address).

<PublicKeyRecord>

A 4-byte integer followed by a byte-array that contains the server's public key. The integer specifies the size of the byte-array. The byte-array (for the publickey) consists of three parts: a UTF8-string that describes the key type, a two-byte option field reserved for future use, and a byte-array that contains the public key itself. For example, the UTF8-String "DSA_PUB_KEY" indicates that the <PublicKeyRecord> contains a DSA public key. The storage format of the DSA key in the byte-array could then be found from the handle "0.type/DSA_PUB_KEY". Public key in the <PublicKeyRecord> can be used to authenticate any service response from the handle server.

The <PublicKeyRecord> may also contain an X.509 certificate. This happens if the key type field contains the UTF8-String "CERT.X509". In this case, "CERT.X509" will map to the handle "0.TYPE/CERT.X509". The handle may contain information that describes the syntax and semantics of the public key or its certificate. Additional key type may also be registered (as handles under "0.TYPE") to further distinguish different kinds of X.509 certificates. For example, "CERT.X509.DSA" may be used to denote X.509 certificates that contain DSA public keys. If the key type field of a <PublicKeyRecord> declares "CERT.X509.DSA", the <PublicKeyRecord> must contain a X.509 certificate with a DSA public key in it."

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

A 4-byte integer followed by an array of triplets consisting 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 (0x01), handle administration (0x02), or both. The <TransmissionProtocol> is also an octet (as a bit mask) that specifies the transmission protocol. Possible transmission protocols include TCP (0x01), UDP (0x02), and HTTP (0x04). The

<PortNumber> is a 4-byte unsigned integer that specifies the port number used by the interface. The default port number is 2641.

Figure 3.2.2 shows an example of handle service site in terms of a HS_SITE value. The HS_SITE value is assigned to the naming authority handle "0.NA/10". The <PrimaryMask> indicates that it is the only primary site of the handle service. The site consists of three handle servers, as indicated in the <NumOfServer>. These servers provide handle resolution and administration service for every handle under the naming authority "10". The first server record (ServerID 0) shows two service interfaces, one for handle resolution and the other for handle administration. Each interface has its own port.

Each server within a service site is responsible for a subset of handles managed by the handle service. Clients can find the responsible server by performing a common hash-operation. The hash-operation will first convert all ASCII characters in the handle into upper-case. It then applies the MD5 hashing upon the portion of the converted handle string (according to the <HashOption> entry). The result is a 16-byte integer. The absolute value of the integer will be divided by the number of servers (specified in the <NumOfServer> entry). The remainder is the sequence number (starting with zero) of the <ServerRecord> listed in the HS_SITE value. From the <ServerRecord>, clients can find the IP address of the handle server for their handle requests.

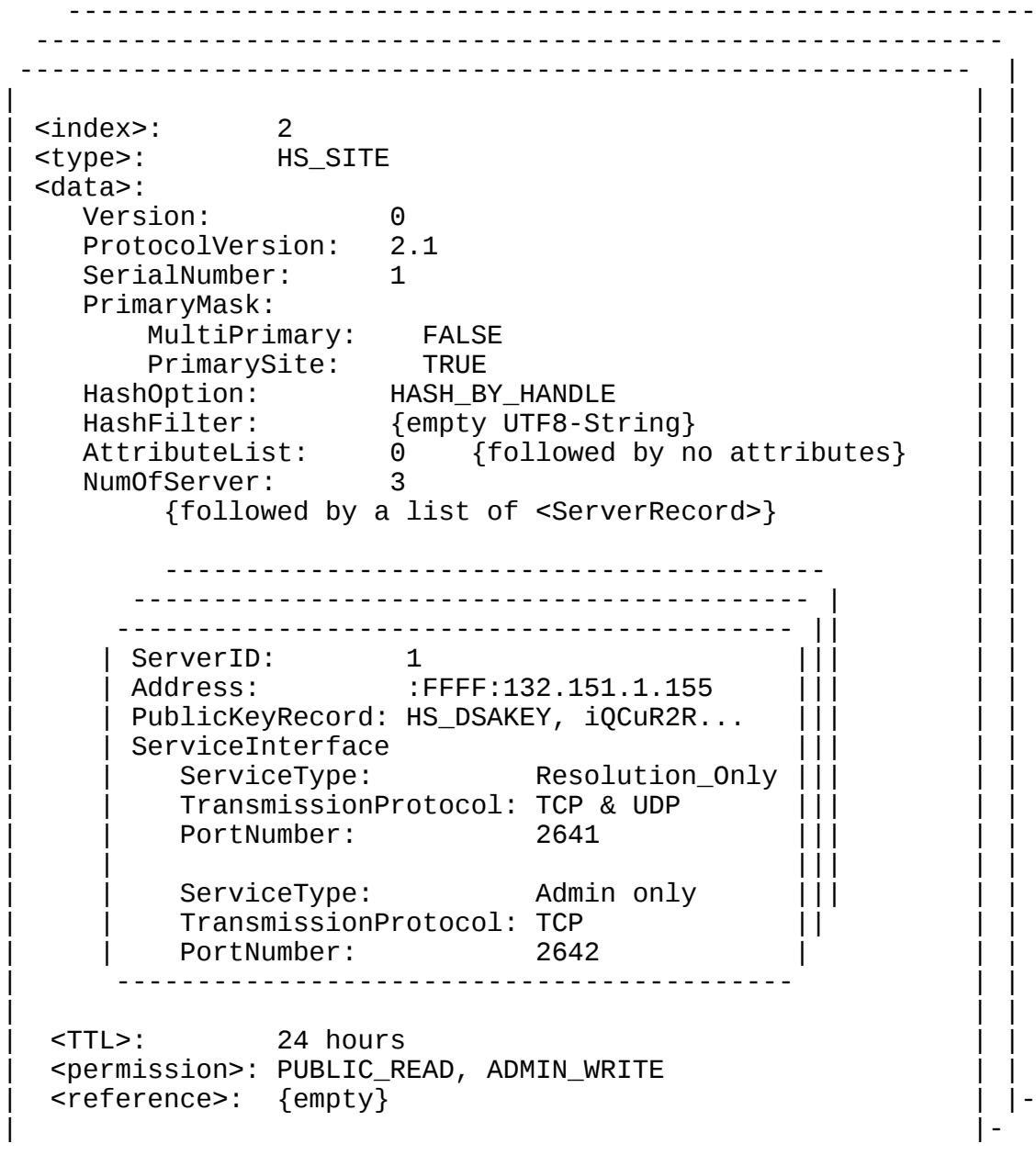


Fig. 3.2.2: The primary service site for the naming authority "10"

3.2.3. Naming Authority Delegation Service: HS_NA_DELEGATE

The HS_NA_DELEGATE is a pre-defined Handle System data type. It has the exact same format as the HS_SITE value. Like HS_SITE values, HS_NA_DELEGATE values are used to describe service sites of a LHS.

HS_NA_DELEGATE values may be assigned to naming authority handles to designate naming authority administration to a LHS. A naming authority handle with a set of HS_NA_DELEGATE values indicates that all child naming authorities of the naming authority are managed by the LHS described by the HS_NA_DELEGATE values.

For example, suppose the naming authority "foo.bar" decides to have its child naming authorities delegated to a LHS. To achieve this, one may assign the naming authority handle "0.NA/foo.bar" with a set of HS_NA_DELEGATE values that describes the LHS. The set of HS_NA_DELEGATE values indicate that the service information of any child naming authority of the "foo.bar", such as "foo.bar.baz", can be found by querying the naming authority handle "0.NA/foo.bar.baz" from the LHS.

3.2.4. Service Handle: HS_SERV

Any handle service, global or local, can be defined in terms of a set of HS_SITE values. These HS_SITE values may be assigned directly to the relevant naming authority handle, or an additional level of indirection may be introduced through the use of service handles. A service handle may be thought of as a name for a handle service. It may be used to maintain the HS_SITE values for the handle service and referenced from a naming authority handle via a HS_SERV value. A HS_SERV value is a handle value whose <type> field is HS_SERV and whose <data> field contains the reference to the service handle. HS_SERV values are typically assigned to naming authority handles to refer clients to the responsible handle service.

Use of service handle allows sharing of service information among multiple naming authorities. It also allows changes to service configuration (e.g., adding a new site) to be made in one place rather than in every naming authority handle involved. The mechanism may also be used to support service referral from one handle service to another for whatever reason.

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.

Service handles can be registered under the reserved naming authority "0.SERV". Handles under "0.SERV" are managed by the GHR. For example, the service handle "0.SERV/123" may be created to maintain the service information for the handle service that manages handles under the naming authority "123" and any of its sub-naming authorities.

Similarly, a service handle "0.SERV/a.b.c" may be created to host the service information for the handle service that manages handles under the naming authority "a.b.c".

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 are not signaled as an error. Looped reference of service handles or HS_SERV values that point to non-existent service handles should be caught and error conditions passed back to the user.

3.2.5. Alias Handle: HS_ALIAS

In practice, it is very possible that a digital object may have multiple names that will identify the object. The Handle System supports such feature via the pre-defined data type HS_ALIAS. An HS_ALIAS value is a handle value whose <type> field is HS_ALIAS and whose <data> field contains a reference to another handle. A handle with a HS_ALIAS value is an alias handle to the handle referenced in the HS_ALIAS value. An alias handle should not have any additional handle values other than HS_ALIAS or HS_ADMIN (for administration) values. This is necessary to prevent any inconsistency between a handle and its aliases.

During a handle resolution, a client may get back an HS_ALIAS value. This indicates that the handle in question is an alias handle. The client may then retry the query against the handle specified in the HS_ALIAS value until final results are obtained.

The use of alias handle introduces a number of special considerations. For example, 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 user.

One potential use of alias handle would be to support the transfer of ownership of any named resource. 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 any broken reference, the handle used before the ownership transfer may be changed into an alias handle and point its HS_ALIAS value to the newly created handle.

3.2.6. Primary Site: HS_PRIMARY

HS_PRIMARY is a pre-defined data type used to designate the primary service sites for any given handle. A handle service with multiple primary service sites is called a multi-primary service. Otherwise

it is called a single-primary service. Each handle managed by a multi-primary handle service may specify its primary service sites in terms of an HS_PRIMARY value. A HS_PRIMARY value is a handle value whose <type> field is HS_PRIMARY and whose <data> field contains a list of references to HS_SITE values. Each of these HS_SITE defines a primary service site for the handle.

There can be at most one HS_PRIMARY value assigned to each handle. Otherwise it is an error. A handle with no HS_PRIMARY value but managed by a multi-primary handle service is not an error. In this case, every primary service site of the handle service will also be the primary site for the handle. Handles managed by a single-primary handle service do not need any HS_PRIMARY values and any such values should be ignored.

3.2.7. Handle Value List: HS_VLIST

HS_VLIST is a pre-defined data type that allows a handle value to be used as a 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 unsigned integer followed by a list of references to other handle values. The integer specifies the number of references in the list. The references may refer to handle values under the same handle or handle values from any other handles. Each reference is encoded as an UTF8-string followed by a 4-byte unsigned integer that identifies the referenced handle and its value index.

HS_VLIST values may be used to define administrator groups for handles. In this case, each reference in the HS_VLIST defines a member of the administrator group and the HS_VLIST value identifies the group as a whole. Client software must be careful, however, to avoid cyclic definition of value references.

4. Handle System Service Model

The Handle System is a distributed global name service. It consists of a single distributed Global Handle Registry (GHR) and unlimited number of Local Handle Services (LHS). 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 relationships 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 can be any number of Local Handle Services (LHSs). Each LHS must be registered with the GHR to manage handles under a distinct set of naming authorities. Naming authorities are managed by the GHR via naming authority handles (i.e., handles under the naming authority "0.NA"). A naming authority handle can also be used to locate the service information (in terms of HS_SITE values) that describes the handle service responsible for handles under the naming authority. From the service information, clients can choose a service site and locate the responsible server for their handle requests.

Handle System service components are scalable and extensible to accommodate any large amount of service load. A 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 any single point of failure and allows load balancing among these service sites. Using multiple servers at any service site distributes the service load into multiple server processes 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 mainly 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 and administration service to non-naming-authority handles. Unlike any LHS, which mostly manages handles under a few naming authorities, the GHR is primarily used to register naming authorities and provide service information for every LHS. In other words, the GHR is the single root service that registers every LHS and provides their service information via the use of naming authority handle(s). Every naming authority under the Handle System must be registered under the GHR as a naming authority handle. The naming authority handle provides the service information of the handle service that manages all the handles under the naming authority. The service information may be provided in terms of a set of HS_SITE values, or an HS_SERV value that refers to a service handle, as described earlier.

The GHR may consist of multiple service sites, each described in a HS_SITE value. These HS_SITE values are assigned to the designated naming authority handle "0.NA/0.NA", also called the root handle. The

root handle is the naming authority handle that maintains the service information for GHR. Top level naming authorities can only be created by administrators of the root handle.

In order to communicate with the GHR, client software needs the GHR service information beforehand. The service information may be distributed initially with the client software, or obtained from some other secure sources (e.g., postal mail, secure web site, etc.). Client software may keep the service information to communicate with the GHR until the service information becomes expired (according to its TTL). The GHR must update its service information (assigned to the root handle) every time it changes its configuration. Client software with out-dated service information will be notified of the update every time it communicates with the GHR. The GHR must be maintained in such a way that any client software with out-dated GHR service information can still query the root handle for the latest update.

Fig. 4.1.1 shows the GHR service information in terms of a set of HS_SITE values. The GHR may consist of a number of service sites, each described in a HS_SITE value. The figure shows a GHR service site located in US East Coast, as indicated in the <AttributeList>.

```

-----
<index>:      3
<type>:       HS_SITE
<data>:
  Version:      1
  ProtocolVersion: 2.1
  SerialNumber:  1
  PrimaryMask:
    MultiPrimary:  TRUE
    PrimarySite:   TRUE
  HashOption:    HASH_BY_HANDLE
  HashFilter:     {empty UTF8-String}
  AttributeList:  1
    Description:  Service site at US East Coast
  NumOfServer:   3

  -----
  -----
  -----
  ServerID:      1
  Address:        :FFFF:132.151.2.150
  PublicKeyRecord: HS_DSAKEY, iQCuR2Rnw...
  ServiceInterface
    ServiceType:  Resolution & Admin
    TransmissionProtocol: TCP & UDP
    PortNumber:   2641
  -----
  -----
<TTL>:          24 hours
<permission>:   PUBLIC_READ, ADMIN_WRITE
<reference>:     {empty}
-----

```

Figure 4.1.1: GHR service information

The GHR and its service information provide an entry point for any client software to communicate with the Handle System. For any given handle, client software can query the GHR for its naming authority handle. This will return the service information of the LHS that manages every handle under the naming authority. The service information will direct the client software to the handle server within the LHS that manages the handle.

4.1.2. Local Handle Service (LHS)

A Local Handle Services (LHS) manages 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. Note that a LHS is not a "local" service in terms of any network topology. It is called a "Local" Handle Service because it typically manages a restricted (local) namespace.

A naming authority is "homed" at a LHS if all handles under the naming authority are managed by the LHS. A LHS may be home to multiple naming authorities. On the other hand, a naming authority may only be "homed" at one LHS. Note that a naming authority may also be homed at the GHR.

```

-----
-----
<index>:      3
<type>:       HS_SITE
<data>:
  Version:          1
  ProtocolVersion:  2.1
  SerialNumber:     1
  PrimaryMask:
    MultiPrimary:   FALSE
    PrimarySite:    TRUE
  HashOption:       HASH_BY_LOCALNAME
  HashFilter:       {empty UTF8-String}
  AttributeList:    1
    Description:    Local Service for "10"
  NumOfServer:      2
    -----
    -----
    ServerID:        1
    Address:          :FFFF:132.151.3.150
    PublicKeyRecord:  HS_DSAKEY, iQCuR2R...
    ServiceInterface:
      ServiceType:    Resolution & Admin
      TransmissionProtocol: TCP & UDP
      PortNumber:     2641
    -----
  <TTL>:            24 hours
  <permission>:     PUBLIC_READ, ADMIN_WRITE
  <reference>:      {empty}
-----
-----

```

Figure 4.1.2: LHS service information

Like the GHR, a LHS may also 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 service handle or to the relevant naming authority handle(s). Fig. 4.1.2 shows an example of HS_SITE values for a LHS. These HS_SITE values are assigned to the naming authority handle "0.NA/10". This suggests that the naming authority "10" is "homed" at the LHS specified in these HS_SITE values. Clients may query the GHR to obtain the service information in order to communicate with the LHS. Administrators of the naming authority handle are responsible for maintaining the service information and keeping it up to date.

Note that a LHS may refer its clients to another LHS in response to a service request. This allows the LHS to further distribute its service in a hierarchical fashion.

4.2. Handle System Middle-Ware Components

Handle System middle-ware components currently include Handle System caching servers and Handle System proxy servers. These Handle System middle-ware components are clients to Handle System service components, but servers to Handle System client software. Handle System middle-ware components are used to provide additional interfaces to the basic handle service. For example, a Handle System caching server may be used to share resolution results within a local community. Additionally, a Handle System proxy server can be used to bypass any organizational firewall via HTTP tunneling.

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 the service information of any LHS, allows re-use of information obtained from earlier queries.

Each handle value contains a <TTL> (Time to Live) field that tells a caching service how long the cached value may be regarded as valid. A zero-value TTL indicates that the value can only be used for the transaction in progress and should not be cached. A caching service may obtain its data directly from a handle service, or from another caching service that eventually gets its data from the handle service.

A caching service may be defined in terms of an HS_SITE value and may consist of multiple caching servers. For any given handle, clients can find the responsible caching server within the caching service by using the same hashing algorithm as used in locating the handle server within any handle service.

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. Clients are responsible to set up their trust relationship with the caching service that they select. They will also rely on the caching service to properly authenticate any response from any handle server.

4.2.2. Handle System Proxy Server

Handle System proxy servers can be used to enable handle resolution via other Internet protocols. For example, CNRI has built and made available a Handle System HTTP Proxy Server that will process any handle resolution in terms of HTTP protocol. The current DNS address for the proxy server is at "hdl.handle.net". The proxy server allows any handle to be resolved via a HTTP URL. The URL can be constructed as "http://hdl.handle.net/<handle>", where <handle> can be any handle from the Handle System. For example, the handle "ncstrl.vatech_cs/tr-93-35" can be resolved via the HTTP 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 in terms of a HTTP request. The proxy server will query the Handle System for the handle data and return the results in terms of HTTP response.

Using HTTP URLs allows handles to be resolved from standard web browsers without any additional client software. However, such reference to the handle also ties itself to the proxy server. If the 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 server 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. Clients are responsible to set up their trust relationship with the proxy server that they select. They will also rely on the proxy server to properly authenticate any response from any handle server.

4.3. Handle System Client Components

Handle System client components are client software that communicates with the Handle System service components. Client software may speak the Handle System protocol and send its request directly to a service

component. The response from the service component may be the final answer to the request, or a referral to another service component. The client software will have to follow the referral in order to complete the transaction.

Client software may also be configured to tunnel its request via a middle-ware component. The middle-ware component will thus be responsible for obtaining the final result and returning it to the client. Unlike service components, middle-ware components will only return final results of client's request. No service referral will be returned from middle-ware components.

Various Handle System client components may be developed for various applications. The CNRI Handle System Resolver [17] is one such component. The resolver extends web browsers (e.g., Netscape or Microsoft Internet Explorer) in such a way that handles can be resolved directly in terms of "hdl:" Uniform Resource Identifiers (URIs). The Grail web browser [18], a freely downloadable software developed in Python [19], 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. Details of the handle URI syntax will be specified in a separate document.

5. Handle System Operation Model

Handle System operations can be categorized into resolution and administration. Clients use the handle resolution service to query for any handle values. Handle administration allows clients to manage handles, including adding and deleting handles, and updating their values. It also deals with naming authority administration via naming authority handles. This section explains how various Handle System components work together to accomplish these service operations.

Both resolution and administration may require authentication of the client. The authentication can be done via the Handle System authentication protocol described later in this section. Whether authentication is required or not depends on the kind of operation involved and the permissions assigned to the relevant handle value, and policies deployed by the relevant service components.

The Handle System protocol specifies the syntax and semantics of each message exchanged between Handle System clients and its server components. This section provides a high level overview of the

protocol used to accomplish any service operation. The exact programmatic detail of each message (i.e., their byte layout or syntax) is specified in a separate document [8].

5.1. Handle System Service Request and Response

The Handle System provides its service in response to client 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 service information (in terms of HS_SERV or HS_SITE values) of the handle service that manages all handles under the naming authority. Any handle request must be directed to the home service of the handle in question. Clients may find the home service by querying the corresponding naming authority handle against the GHR. Alternatively, this information may be found in a local cache or even be part of a local client configuration. Given the service information, clients may select a service site and locate the responsible handle server within the site.

To resolve the handle "ncstr1.vatech_cs/te-93-35", for example, client software needs to know the home service for the naming authority "ncstr1.vatech_cs". The home service can be obtained by querying the naming authority handle "0.NA/ncstr1.vatech_cs" against the GHR. The GHR will return the service information in terms of the HS_SITE values assigned to the naming authority handle. From the service information, clients can pick a service site, find the responsible handle server within the site, and send the resolution request to the handle server.

Clients may require digital signatures from a handle server in order to authenticate any response from the server. The signature can be generated using the server's private key. Clients may verify the signature using the public key available from the service information (refer to the <PublicKeyRecord> entry discussed in 3.2.2).

A communication session may also be established between any client and handle server. Each session is identified by a unique session ID managed by the server. A session may be used to manage requests that require multiple interactions. It may also be used to share any TCP connection or authentication information among multiple service transactions. Each session may establish a session key and use it to

authenticate any message exchanged within the session. It may also be used to encrypt any message between the client and the server to achieve data confidentiality.

The following diagram shows a handle resolution process in terms of messages exchanged between client software and Handle System service components. In this case, the client is trying to resolve the handle "ncstr1.vatech_cs/tr-93-35". It assumes that the client has yet obtained the service information of the LHS "homed" by the naming authority "ncstr1.vatech.cs". The client has to get the service information from the naming authority handle managed by the GHR. The service information allows the client to locate the responsible LHS and query for the handle value.

```
[HS Client] -----> [Global Handle Registry]
    1. ask for the service
       information from the
       naming authority handle
       "0.NA/ncstr1.vatech_cs"

[HS Client] <----- [Global Handle Registry]
    2. service information for
       the naming authority
       "ncstr1.vatech_cs"

[HS Client] -----> [Local Handle Service]
    3. query the handle
       "ncstr1.vatech_cs/tr-93-35"
       against the responsible
       handle server

\... ...

(optional client authentication, depending on the service request)

\... ...

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

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 queries the GHR to find the home service for the handle's naming authority. The

GHR returns the service information of the LHS that manages every handle under the naming authority. From the service information, the client can find the responsible handle server and query the server for the handle. The server may set up a session to authenticate the client if any of the handle value requires authentication. Otherwise, the server will simply return the handle value to the client. The server may send a digital signature as part of its response if required by the client.

The above procedure assumes that the client software already has the GHR service information. That information was likely obtained from the client software distribution. The GHR will notify the client software if it learns that the service information used by the client software is out of date. Client software may retrieve the latest service information from the root handle "0.NA/0.NA". The root handle also maintains the public key that may be used to authenticate the service information.

Note that a client may cache the service information of any naming authority 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. Client software may also be configured to query a caching or proxy server directly for any handle. In this case, the caching or proxy server will act as the [HS Client] in Figure 5.1 before returning the query result to the client.

Client software under certain organization may also elect to bypass the GHR and communicate directly with a LHS managed by the organization. Doing so may achieve quicker response for handles managed under the LHS. The client software will be referred to the GHR for handles not managed by the LHS.

5.2. Handle System Authentication Protocol

The Handle System supports handle administration over the public Internet. Access controls can be defined on each handle value. The Handle System authentication protocol is the protocol used by any handle server to authenticate handle administrator upon any administration request. The authentication is also necessary when clients query for handle values that are read-only by the handle administrator. Handle administration include adding, deleting or modifying handle values, and adding or deleting handles. Naming authority administrations are carried out as handle administrations over the corresponding naming authority handles.

The Handle System authentication protocol does not perform any server authentication. However, a client may authenticate any server response by asking the server to sign its response with digital signature.

By default, the Handle System authenticates clients via a challenge-response protocol. That is, after receiving a client's request, the server issues a challenge to the client if authentication is necessary. To be authenticated as the administrator, the client has to return a challenge-response, a message that demonstrates possession of the administrator's secret. The secret may be the private key or the secret key of the administrator. This challenge-response allows the server to authenticate the client as the handle administrator. Upon successful authentication, the server will fulfill the client's request if the administrator is given sufficient permission.

For example, suppose a client sends a request to the handle server to add a new handle value. The server will issue a challenge to the client in order to authenticate the client as one of the handle administrators. If the client possesses the private key of the administrator, she can use it to sign the server's challenge and return the signature as part of her challenge-response. The server will validate the signature in order to authenticate the client. The client will be notified if the validation fails. Otherwise, the server will further check if the administrator has the permission to add the handle value. If so, the server will add the handle value and report success to the client. Otherwise, a permission-denied message will be returned.

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

```

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

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

[Client] -----> [Handle Server]
      3. reference to handle administrator
      + challenge-response from client

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

Figure 5.2: Handle System authentication process

In Figure 5.2, the client sends an administration request to the handle server (along with optional credential discussed later). The server decides that client authentication is required and issues a challenge to the client. The client identifies itself as a handle administrator and returns the challenge-response to the server. The server authenticates the client as the administrator based on the challenge-response. It also checks to see if the administrator is authorized for the administration request. If so, the server will fulfill the request and acknowledge the client.

Handle servers must authenticate the client before fulfilling any request that requires administrator privilege. The exact authentication process varies depending on whether public key or secret key is used by the administrator. It also depends on whether the handle used to store the administrator's key is managed by the same handle server or not.

When public key is used, the challenge-response from the client contains its digital signature over the server's challenge. The server can authenticate the client by verifying the digital signature based on the administrator's public key. If secret key is used, the challenge-response from the client carries the Message Authenticate Code (MAC) generated using the secret key. The server may authenticate the client by generating the same MAC using the administrator's secret key and comparing it against the challenge-response.

The reference to handle administrator in Fig 5.2 is also called a key-reference. It refers to a handle value that contains the key used by the administrator. If the key-reference is managed by the same handle server (e.g., a handle value assigned to the same handle), the server may use the key directly to do the authentication. If the key-reference is managed by some other handle server (whether or not within the same handle 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 carries both the server's challenge and the client's challenge-response. The key-server will return a verification-response, signed using the key-server's private key. The content of the verification-response will depend on the handle value referenced by the key-reference. If the key-reference refers to a public key used by the administrator, the verification-response will contain the public key of the administrator. Otherwise, the key-server will verify the challenge-response on behalf of the requesting server and return the result in the verification-response. The following diagram shows the control flow of the authentication process where the key-reference refers to a handle value that contains the administrator's public (or secret) key and the key-server is some other handle server.

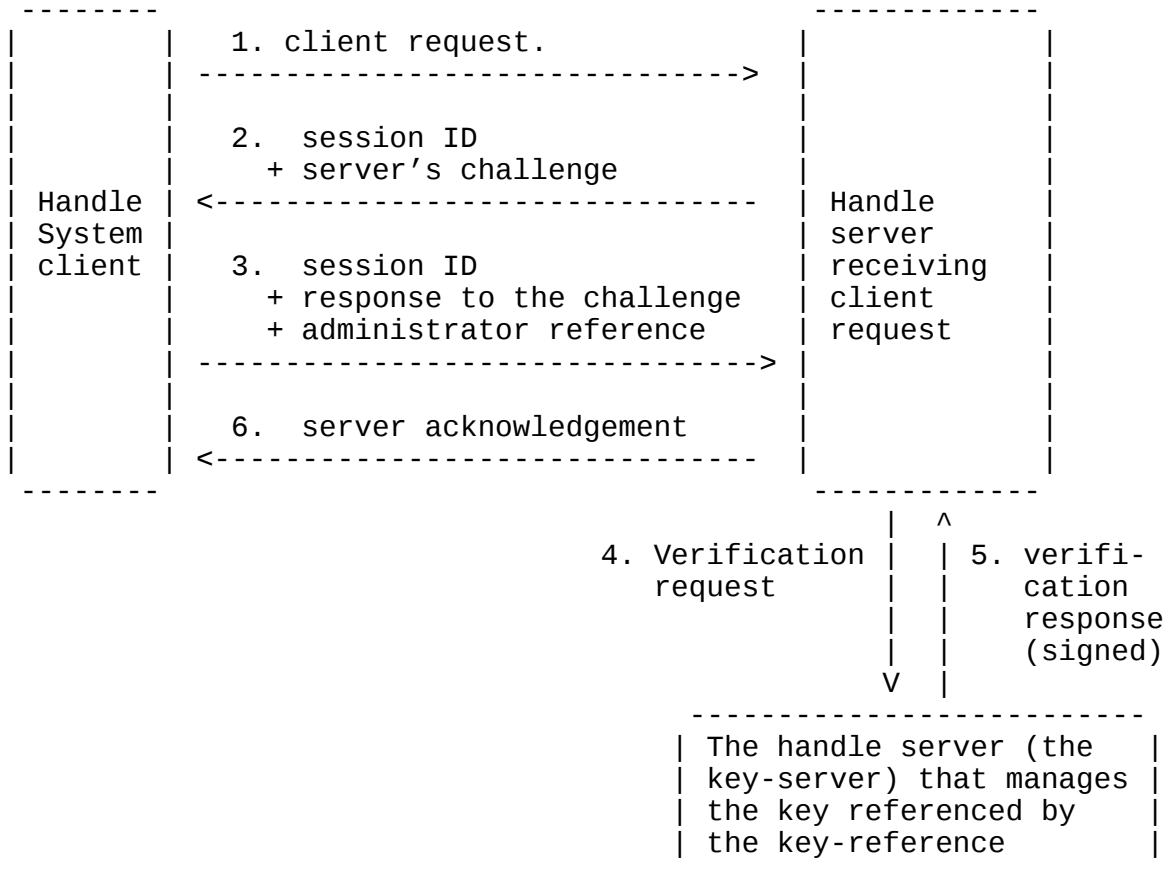


Figure 5.3: Authentication process requiring verification from a second handle server

Secret key based authentication via a second handle server, i.e., the key server, provides a convenient way to share a common secret key (e.g., pass phrase) among handles managed by different handle servers. 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 cannot determine whether the verification-request comes from a legitimate handle server. A handle service may set its local policy so that secret key based authentication can only be carried out if the handle server (receiving the client request) is also the key-server.

Local handle services may define additional local policies for authentication and/or authorization. Handle System service components may also choose to use other Internet authentication mechanisms such as Kerberos [20] or some Transport Layer Security protocol [21]. Details of these will be addressed in a separate document.

6. Security Considerations

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

The Handle System delegates handle administration to each handle administrator who may or may not be the server administrator. Handle administrators are allowed to choose their own public/secret keys used for authentication. The security of Handle System authentication depends on the proper key selection and its maintenance by the handle administrator. Handle administrators must choose and protect their authentication keys carefully in order to protect the handle data. Handle server implementations may deploy policies that regulate the selection of public/secret keys used for authentication. For example, a handle server may require that any authentication key must be no less than certain number of bits. It may also prohibit the use of secret keys because of the potential dictionary attack.

The Handle System data model supports execution permission (PUBLIC_EXECUTE, ADMIN_EXECUTE) for each handle value. While this allows better sharing of network resources, it also raises many security considerations. Execution privilege should be restricted within the permissions of certain user account (corresponding to the handle administrator) on the server to prevent system-wide disruption. Switching between computing platforms for the server should also be careful to avoid any unexpected behavior. Implementations may choose not to support the execution permission, or provide options so that it can be disabled.

To protect against any irresponsible use of system resource, handle servers may implement quota control. The quota control can be used to put limits on the number of handles under a naming authority, the number of handle values allowed for any given handle, the maximum size of any handle value, and the number of sub-naming authorities under a naming authority. Handle servers must report error if the result of a handle administration violates any of these limits.

7. Acknowledgements

This work is derived from the earlier versions of the Handle System implementation. The overall digital object architecture, including the Handle System, was described in a paper by Robert Kahn and Robert Wilensky [22] in 1995. Development continued at CNRI as part of the Computer Science Technical Reports (CSTR) project, funded by the Defense Advanced Projects Agency (DARPA) under Grant Number MDA-972-92-J-1029 and MDA-972-99-1-0018. Design ideas are based on those discussed within the Handle System development team, including David Ely, Charles Orth, Allison Yu, Sean Reilly, Jane Euler, Catherine Rey, Stephanie Nguyen, Jason Petrone, and Helen She. Their contributions to this work are gratefully acknowledged.

The authors also thank Russ Housley (housley@vigilsec.com), Ted Hardie (hardie@qualcomm.com), and Mark Baugher (mbaugher@cisco.com) for their extensive review and comments, as well as recommendations received from other members of the IETF/IRTF community.

8. References and Bibliography

- [1] Sun, S. and L. Lannom, "Handle System Overview", RFC 3650, November 2003.
- [2] Mockapetris, P., "Domain Names - Concepts and Facilities," STD 13, RFC 1034, November 1987.
- [3] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, November 1987.
- [4] Wahl, M., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.
- [5] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.
- [6] Yergeau, F., "UTF-8, A Transform Format for Unicode and ISO10646", RFC 2279, January 1998.
- [7] The Unicode Consortium, "The Unicode Standard, Version 2.0", Addison-Wesley Developers Press, 1996. ISBN 0-201-48345-9
- [8] Sun, S., Reilly, S. and L. Lannom, "Handle System Protocol (ver 2.1) Specification", RFC 3652, November 2003.
- [9] Berners-Lee, T., Masinter, L. and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

- [10] Housley, R., Polk, W. Ford, W. and D. Solo, "Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.
- [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] Balenson, D., "Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers", RFC 1423, February 1993.
- [14] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
- [15] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, December 1995.
- [16] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.
- [17] CNRI Handle System Resolver, <http://www.handle.net/resolver>
- [18] Grail browser home page, <http://grail.sourceforge.net/>
- [19] Python language website, <http://www.python.org/>
- [20] Kohl, J. and C. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.
- [21] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.
- [22] R. Kahn, R. Wilensky, "A Framework for Distributed Digital Object Services, May 1995, <http://www.cnri.reston.va.us/k-w.html>
- [23] American National Standards Institute. ANSI X9.52-1998, Triple Data Encryption Algorithm Modes of Operation. 1998.

9. Authors' Addresses

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

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

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

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

10. Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

