Open Network Handles

—

Definitions and Specifications *(DRAFT 0.5)*

Michael J. O'Donnell

13 August, 2002

Abstract

I propose a system of *Open Network Handles* to provide permanent primitive network handles promiscuously to all who request them. Handles provide an intermediate level of service between IP numbers and domain names. While assignment of IP numbers is constrained by routing considerations, the owner of a handle may reassign it to different addresses over time for mobility or changes in configuration of resources. Unlike domain names, handles carry no significance in natural language, so they should not have high commercial value, nor should they attract disputes based on assertions of rights in significant names.

This document lays out the basic definitions underlying handle systems, and the specifications toward which a network handle system should be designed.

0.2 I think that the definitions have basically the right structure, and should be critiqued for details. The requirements are a mess, and need to be reorganized carefully, possibly with guidance from the RFC archives. For now, I'm just writing down requirements, and questions about requirements, as I think of them, to avoid forgetting them.

0.3 I have used the Internet Society's `rfc` style, but it doesn't appear to work as advertised, and several commands don't work. I have tried to use the modal words "should," "must," etc. according to the standards for RFCs. The word "requirements" seems to carry a lot of special baggage in the RFC literature that I don't understand, so I switched to "specifications." I reorganized the categories of requirements/specifications.

0.4 I expanded the discussion, and replaced "central registrant" with "global system."

0.5 I added a discussion of the Uniform Resource Names project.

1 Definitions

1.1 Abstract vs. Concrete Elements of Definitions

I am not aware of an accepted standard for giving precise definitions of systems of this sort. Useful and precise discussion of the specifications for a handle system depends on a strange combination of concrete and abstract thinking. The sort of network handle system proposed here is intended to become part of the nearly invisible infrastructure of the global Internet, not part of the user interface. The evaluation of such a proposal depends on its concrete value to users of a network, based on their concrete behavior. But the conceptual objects involved in the infrastructure are all freely created abstractions inside the computers and connections constituting the network.

It is very difficult to discuss computer/network systems without referring to their abstractions as if they were material objects. But the definitions of such abstractions are not really definitions of the composition of objects, but only of their relations to other objects, and the operations that may and may not be performed on them. A few of those relations and operations involve concrete objects in the "real" world outside of computer/network systems, and all value in the system derives from those external connections. But most of the technical issues that must be resolved are usually completely internal. So we must find a way to discuss the internal relations and operations.

The right essential style for such discussions seems to be the style of definitions of *abstract data types*, which was imported into a lot of design discussions in *object-oriented programming*. In the following definitions, I try to capture that style with somewhat intuitive language. But each of these definitions is intended to be the basis for a very precise understanding of what is and is not a successful implementation of a *network handle system*.

1.2 The Definitions Themselves

Definition 1 *An* agent *is any abstract identity that we recognize as being capable of owning some sort of authority.*

For most purposes, we may think of an agent as a human being. But an agent may also be a corporation, a competent being of another species, a department within a corporation, a role played by a sequence of human beings, or a computer system. I definitely want to leave the scope of possible agents open for now.

Definition 2 *A* token *is an object that can be copied, transmitted and tested for equivalence against other copies of tokens. Two token copies are* equivalent *if they are joined by a chain of copying and transmission.*

For most purposes, we may think of a token as a string of bits, and the test for equivalence is just an equality test. But a token that is copied between radically different underlying computing/networking infrastructures, either because

of contemporaneous diversity or because of change over the lifetime of the token (which in some cases may go into decades and centuries), may need to be translated into a new format while retaining its abstract identity. In practice we will almost surely refer to tokens as integers, but without using the structure imposed by the arithmetic operations and inequalities.

Now, we consider three different types of tokens: *addresses*, *handles*, and *names*. The distinction is tricky to express and grasp, and the particular sorts of addresses, handles, and names discussed here do not line up well with the objects discussed elsewhere under those terms. The problem is that the classification of tokens into addresses, handles, and names has little to do with the tokens themselves, but mostly with our intended uses of them. Furthermore, each type of token can in principle be used for the purposes intended for the other two, because each type is only useful along with maps between the three types. Any one alone of these types of token is sufficient in principle for routing messages in a network. The distinction has to do with which sorts of operations and combinations of operations each type is designed to support particularly efficiently and/or reliably.

Definition 3 *An* address *is a token that an agent can use to specify the delivery of a message to another agent. At any given time, every message to a given address is routed to a particular agent, called the* owner *of the address. But address ownership may change for many reasons, not all of them under the control of the owner. At any time, an agent may discover which addresses it owns.*

By calling a token an address*, we indicate that it is important that it be sufficient to determine the routing of individual small packets of information efficiently. This requirement for efficient routing normally induces constraints on the assignment of addresses to owners based on network topology.*

For most purposes, we may think of an address as an IP number. But UDP already uses addresses consisting of an IP number and a port number. More complex sorts of addresses are often useful. For instance, a list of IP numbers used as a hunt list is a type of complex address, routing messages to an agent that owns all of the IP numbers in the list. Normally this agent will be an individual human being, a corporation, or a department, but we may always conceive a new abstract agent corresponding to any weird list of addresses, just to let the discussion continue.

Definition 4 *A* handle *is a token with which we can perform the following operations.*

1. *An agent may acquire authority over the meaning of a newly created handle. This agent becomes the* owner *of the handle.*

2. *The owner of a handle may at any time associate that handle with an address. The owner may change the associated address at will.*

3. *An agent (not normally the owner) with a copy of a handle may use that copy to construct a query to discover the associated address. In this case, the given handle* resolves to *the given address.*

4. *When a handle resolves to an address, then with high reliability the association of that handle with that address has been performed through some chain of authorizations commencing with the owner of the handle.*

By calling a token a handle*, we indicate that it is important that we be able to process handle/address associations/reassociations and resolution of queries according to the most recently processed (re)association of the handle. We must process these transactions efficiently, but at a substantially lower rate than the rate at which an address is used to route individual packets. In principle, a handle may serve as an address (by resolving it for each routing action), and an address may serve as a handle (by letting it follow the handle owner around the network), but the efficiency requirements for the two types of token are incompatible.*

For most networking purposes, the owner of a handle is the only agent who knows a particular secret token, such as a password or a cryptographic key, associated with the handle. If several people know the same secret token associated with a particular handle, we consider them to constitute one collective agent owning the handle. The definition of handle is not intended to entail any particular level of civility among agents. The "chain of authorizations" leading to an association of an address with the handle may include authorizations acquired through coercion, deceit, and accident.

Some discussions of handles, and many applications of handles, may associate them with objects that are not normally thought of as agents. For example, we might want to associate a handle with a text, such as particular version of *Hamlet*. We may accommodate such uses of handles in this discussion, by conceiving of these objects as relatively passive agents. For the purposes of implementing a network handle system, some agent must take responsibility for maintaining the appropriateness of the address associated with the handle. For my purposes in this proposal, it seems best to think of that agent, along with her voluntary agreement to be steward of a particular object, as a sort of abstract agent.

Definition 5 *A* transferable handle *is a handle whose owner has the additional power to completely transfer authority over the handle (including the authority to perform further transfers) to another agent at will, independently of all other operations. The* owner *of a transferable handle at any time is the final holder of authority after all previous transfers.*

When handle ownership is determined by knowledge of a secret token, the owner may always reveal that secret. Such a revelation, by itself, constitutes a grant of shared authority, but *not* a transfer of ownership of the handle. Transfer of ownership requires that all future authority be invested in the new owner, and

not shared with the previous owner nor others who have been granted shared authority. In the case of ownership by knowledge of a *changeable* secret token, an owner may transfer by granting shared authority to a new owner, who then changes the secret token.

Notice that transfer of ownership includes a grant of authority, plus a relinquishing of authority. So, we do not need to modify the line in the definition of handle referring to a chain of authorizations—such a chain may include transfers of ownership as well as shared grants of authority.

The phrase, "independently of all other operations" is crucial to the definition of transferable handle. We might design a handle system in which certain handles are transferable in a block, but not individually. A transferable handle, according to my definition, must be transferable individually, while the original owner retains all other handles.

Definition 6 *A* network handle system *is a system of protocols, software, and other resources that allows agents communicating over a particular network to own and employee handles.*

In discussing the design and implementation of a network handle system, we use the word *handle* to specify a particular technical data format in the network and its host computers. For example, we might use integers as handles. The network handle system is only directly responsible for insuring the integrity of the use of each individual integer as a handle in communications over the network. We will particularly try to design a minimal network handle system, that passes off all relationships between handles and things outside of the system itself to other external systems—for example the relationship between the agent owning a handle and a particular human being is external to a minimal network handle system. But a useful minimal network handle system should provide hooks for external relationships to be established by other systems. For example when a particular network handle system becomes obsolete, it should provide hooks for mapping its handles to handles in a new system. To a user, there is really a single conceptual handle, which is represented differently inside the network during different periods of time. It is too much to expect one technical system design to last forever, but a good design should accommodate external efforts to connect it to its successor.

1.3 Discussion of the Definitions

1.3.1 Handles Provide Continuity, Not Identity Nor Quality

When a group of agents follow the protocols of a handle system, the handle system by itself does not verify the identity or other quality of any agent. It only provides high reliability that all different addresses resolved from a given handle at different times are authorized by the owner. The handle provides a minimum continuity in a sequence of communications, so that the participants may accumulate information about their cocommunicants, and confidence in their identities or other qualities. But all such information and confidence must

originate in the content of the communication, or must be derived from other channels.

A handle system with respect to the definitions above does *not* take any responsibility for, nor depend on, the good behavior of handle owners in their communications with queriers. The system only tries to assure queriers that every use of the same handle returns results authorized by the handle owner—it makes no attempt to assure the reasonableness of the owner's authorizations or responses. For correct behavior, the system must guarantee a high probability that a query yields an authorized response at a reasonable cost in time and communication, and that if handle owner and querier have followed protocols, the querier can recognize an unauthorized response at a reasonable cost.

To repeat, a handle system does not identify agents, nor generate any sort of confidence in them. Rather, it allows information and confidence acquired by other means to accumulate over time, by providing highly reliable assurance that all addresses associated with a handle are authorized by the same agent.

1.3.2 Handles vs. Addresses

An address is not usually a handle. For example, an IP address is not a handle. As long as a single agent owns a particular IP address, that address behaves as a handle. But sometimes IP addresses must be assigned and reassigned to allow efficient routing. When an IP address transfers from one agent to another for routing purposes, it fails to satisfy the definition of a handle. To force an address to act as a handle, we destroy its efficiency as an address.

Changes in address *structure*, such as the change from IPv4 to IPv6, do not by themselves invalidate IP addresses as handles. The definition of handle, through the definition of token, allows for the translation of handle format, as long as the conceptual identity of a handle is preserved. So, as long as a given IPv4 address is translated to a generally known IPv6 address, its quality as a handle is preserved. It is only the transfer to a different agent that disqualifies IP addresses from being handles.

A lot of transfer of IP addresses from one agent to another derives from the scarcity of IPv4 addresses, which leads to temporary assignments of addresses. IPv6 is intended to remedy that scarcity, and eliminate the need to reassign addresses to accommodate scarcity. But IPv6 may still call for occasional reassignment of addresses to maintain routing efficiency, since efficient routing tables need to deal uniformly with subranges of addresses. Even if there were never any involuntary reassignment of IPv6 addresses, they would constitute at best very inefficient handles, since many agents are voluntarily mobile. A mobile agent, using an IPv6 address as a handle, must arrange forwarding from that address to her actual location. Even slow mobility, on a time scale of years, is a problem, since useful handles should often live for many years.

1.3.3 Handles vs. Names

Definition 7 *A* name *is any token that is somehow associated with an agent or other object. When we discover the object associated with the name, we* resolve *the name. When R is a particular method for resolving names, an R-*name *is one that is to be resolved by method R.*

*By calling a token an R-*name*, we indicate that it is relatively convenient and efficient to resolve it by method R, compared to the apparent alternatives. In principle, an R-name may be used as a handle or as an address, but normally the constraints imposed by R resolution are incompatible with the reliability and/or efficiency of the operations for which handles and addresses are designed.*

It is easy and natural to conflate names with handles in a discussion of methods for referring to objects. But for my purposes in this proposal, it is very important to distinguish them.

If *HR* is the handle-resolving method of a particular handle system, then every handle in that system is technically a *HR*-name. But we are usually concerned with names that resolve through more humanly meaningful methods, such as the semantics of a natural language.

On the other hand, a name is not always a handle. Many methods for resolving names are ambiguous, or change over time, so they do not provide the continuity required of a handle. Also, many name-resolving methods do not allow an agent to own a particular name, and reassign its association with an address. In order to use names as handles, we should force the name resolution method to follow the changes of address authorized by the owner. When natural language semantics are involved, we usually lack the social influence to accomplish this.

1.3.4 Handles in the current DNS

The DNS was designed essentially to be a system of handles. But, to make life easier in the absence of good user interfaces for manipulating humanly opaque handles, domain names are expressed as strings of characters, which often have meanings as names resolved through natural languages and/or local jargon. Notice that the domain name `mycompany.com` resolves in two different ways. DNS resolves it through formal tables stored at various network hosts into an IP address, while our human understanding of proper names resolves it to the particular corporation called "mycompany," perhaps with a slightly different capitalization or spelling or spacing.

In principle, DNS domain names can be perfectly good handles from the point of view of network technicalities, and the fact that they are also English language names is a bonus added value. Unfortunately, this added value can have adverse impact on the utility of the whole system.

- The added value of domain names as humanly meaningful tokens increases their commercial value. Along with its beneficial effects, this increase in value prices domain names too high for certain applications that do not

require the human meaning. The higher value also attracts disputes, which add a nonmonetary cost to the use of domain names as handles.

- Second, and perhaps worse, the reassignment of names due to resolution of disputes, or to recover the inherent value of underutilized or underdefended names, leads to administrative policies that violate the permanence of handles. When a domain name is reassigned either due to a challenge from another agent with a claim on the human meaning of the name, or due to failure by the original owner to renew her claim and pay for its continuance, there is a violation of the definition of handle, just as there is when an IP number is reassigned for routing efficiency.

So, while the current DNS constitutes a technically excellent implementation of network handles, the extra value of domain names as natural language names both inflates their price out of the reach of many agents who could afford and make productive use of a mere handle, and it requires administrative actions that violate the permanence required in the definition of handle.

1.3.5 Handles in Natural Language

2 Specifications

2.1 Parties

A network handle system is implemented by a number of parties who accept different responsibilities and rely on one another in different ways:

- global system
- handle owners
- handle queriers

Although the task of the global system may be distributed among different conventional individuals and organizations, I will treat it as a single party in these specifications.

2.2 Technical Specifications

2.2.1 Functional Specifications

1. The network handle system must adhere precisely to the definitions of handle operations above with high reliability, enough to generate widespread confidence in its use.

2. Participation in the system must be voluntary for handle owners and queriers. The system may require some committed support for the global system, but it should benefit as much as possible from volunteer effort in the global system as well.

3. The system must adhere reliably to the definitions above for those owners and queriers who participate voluntarily and follow the system's protocols, independently of those who do not participate or who do not follow the protocols.

4. Each party who participates voluntarily and follows system protocols must be able to discover whether she is relying on parties other than the global system for adherence to the definitions above.

5. When a querier is uncertain of the authenticity of a response, she should be able to query the global system (in the broad sense, including all deputies) directly to get a response that relies only on the global system and end-to-end verification. This verification through the global system is mainly important when the querier fails to make satisfactory contact with the handle owner.

6. The system should allow each agent to create her own handles at will, without any interaction with the global system, but perhaps only after an initial central registration of her first handle.

7. The system should minimize the administrative work required by a global system, and allow as much of that work as possible to be completely automated. A system that achieves reliability completely through communication between querier and owner, with no need for a global system, would be ideal. But such a system may not be feasible.

8. The system should respond to a query with just enough information to allow the querier to contact the owner of the handle. Essentially, that means to return an address, but the system should accommodate more general sorts of addresses than IP numbers when the added generality allows an improvement in generality of agents and/or in network efficiency and reliability that cannot be achieved by further communication between the querier and the owner.

9. **???** How much verification should be available for the *failure* to return an address in response to a query?

10. The system, and its individual handles, should be capable of adapting for operation over the longest conceivable length of time. We should certainly try to accommodate continuous operations for centuries. That doesn't mean that a particular handle format and/or system implementation must survive very long, but the system must have the potential to be upgraded in place, preserving the identities of handles through upgrades, probably by mapping old handles to new.

11. The system should provide for the greatest assurance of the authenticity of its responses that is affordable at a given time, under its other constraints. But that assurance need not, and should not, be actively supported by

the global system, whenever it can be achieved by direct communication between querier and owner.

12. The system should provide protocols that allow it to be used uniformly, transparently, and seamlessly through globally assigned vs. independently assigned handles. In particular, independently assigned handles should still be globally meaningful. But it should make few or no actual restrictions on the treatment of independently assigned handles, allowing for local specializations and experimentations in privately controlled portions of handle space.

13. If possible under all the other constraints, handles should be transferable. This might be feasible for globally assigned handles yet infeasible for independently assigned handles. If possible, both sorts of handles should be transferable.

2.2.2 Performance Specifications

1. The system should support at least one trillion globally assigned handles, fulfilling at least 100,000 (preferably one million) requests for handles per day, and a number of independently assigned handles limited only by the capacity of the agents who assign them.

2. The system should support queries at roughly the same rate as the current DNS system.

2.3 Social, Economical, and Administrative Specifications

1. The global system should assign handles to agents promiscuously on request without taking any responsibility for the behavior of the agents.

2. Registration should not require reliable identification of the agent who registers, although voluntary provision of contact instructions may be needed.

3. The commercial value of globally assigned handles should be as low as possible. In particular, it should be very close to the inherent cost of registration, and very close to (or ideally below) the value of independently assigned handles. Ideally, it should be so low that a benevolent authority will perform registration at no charge.

4. In order to keep the commercial value low, and to minimize externally-driven disputes, the tokens used as handles should have no explicit meaning to the general public, and minimal potential for accidental meaning outside of the handle system.

5. The system should minimize the incentives for a single agent to hoard a large number of globally assigned handles, and maximize the independent assignment of handles by agents. The main point is not to conserve storage by the global system, although that could become important, but to avoid

a flood of requests for multiple handles from the same agent, leading to a delay or denial of service to other agents.

6. Handle ownership should be practically accessible, possibly at lower levels of reliable security, to essentially all users of the Internet, including the very naive.

7. The use of handles to generate network addresses in widely used applications, such as Web browsing and email, should be transparent to the most naive users.

8. **???** What administrative responsibility, if any, should the global system accept for the *address* owner's authorization to associate a handle with her address?

9. The global system should be supported by

   - the smallest possible commitment from one or more highly trusted institutions to guarantee minimally useful operation;
   - voluntary efforts by whomever volunteers, with no need to trust the volunteers;
   - contractors who boost system performance—efficiency and/or reliability—on behalf of particular handle owners and/or queriers.

3 Related Projects

3.1 Uniform Resource Names

An IETF working group has been working on *Uniform Resource Names* (URN) since 1997 or earlier. The basic specifications for URNs look a lot like my specifications for handles. From RFC 1737:

> These are the requirements for URNs' functional capabilities:
>
> - Global scope: A URN is a name with global scope which does not imply a location. It has the same meaning everywhere.
> - Global uniqueness: The same URN will never be assigned to two different resources.
> - Persistence: It is intended that the lifetime of a URN be permanent. That is, the URN will be globally unique forever, and may well be used as a reference to a resource well beyond the lifetime of the resource it identifies or of any naming authority involved in the assignment of its name.
> - Scalability: URNs can be assigned to any resource that might conceivably be available on the network, for hundreds of years.

- Legacy support: The scheme must permit the support of existing legacy naming systems, insofar as they satisfy the other requirements described here. For example, ISBN numbers, ISO public identifiers, and UPC product codes seem to satisfy the functional requirements, and allow an embedding that satisfies the syntactic requirements described here.
- Extensibility: Any scheme for URNs must permit future extensions to the scheme.
- Independence: It is solely the responsibility of a name issuing authority to determine the conditions under which it will issue a name.
- Resolution: A URN will not impede resolution (translation into a URL, q.v.). To be more specific, for URNs that have corresponding URLs, there must be some feasible mechanism to translate a URN to a URL.

Global scope, global uniqueness, persistence, scalability, extensibility, and independence all appear to be equivalent to specifications below for a network handle system, mixing functional specifications with performance specifications somewhat.

I would not choose to mention legacy support as a key requirement. But, with my untutored interpretation of "permit," I would think of this as a very weak requirement, and I would expect it to be satisfied by any sensible design, letting the actual mapping from legacy names to keys be done by tables external to the handle system. However, the URN group seems to have a much stronger interpretation of "permit" than mine, and they seem to have invested a lot of effort to make sure that legacy naming systems can be incorporated as hierarchical subsystems of URNs. The URN group also seems to be very concerned with the structure of particular objects to which URNs refer.

I also would not focus on resolving into URLs. URLs are one sort of extended address, but I am most concerned with resolving into IP numbers, and then adding a few carefully designed generalizations.

For my purposes, to specify the requirements for a desirable network handle system, I would delete the legacy support item, but add

- Minimality: A network handle system should provide just enough service so that a querier with a handle may establish meaningful contact with the owner of the handle. All authentication, and all interpretation of the object referred to by the handle, should be established by direct communication between querier and owner. A minimal network handle system abstracts from IP addresses just enough to provide for continuity in communication with the same agent, and leaves all other concerns to other systems.

My minimality requirement is incompatible with the strong interpretation of the legacy support requirement, and with any accommodation of specific structural information about the objects referred to by handles.

I have chosen to design a minimal network handle system because the continuity service that it provides appears to be usable by a variety of more structured, specialized, and/or meaningful systems without much inefficiency due to the modularization. In particular, I expect that a URN system could be implemented using an existing minimal network handle system to provide global scope, global uniqueness, persistence, scalability, extensibility, and independence, adding other structural properties through separately managed tables.

3.2 CNRI Handle System