Open Network Handles—Requirements
(DRAFT 0.2)

Michael J. O’Donnell
4 August, 2002

Abstract
I propose a system of Open Network Handles to provide permanent primitive network handles promiscuously to all who request them.

Draft status: In draft 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.

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 requirements 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.

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.

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 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 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.

For most purposes, the owner of a handle is the only agent who knows a par-
ticular 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 par-
ticular 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 as-
sociate 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 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 employ handles.
1.3 Discussion of the Definitions

1.3.1 Handles Provide Continuity, Not Identity

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.

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.

Changes in address structure, such as the change from IP v4 to IP v6, 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 IP v4 address is translated to a generally known IP v6 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 IP v4 addresses, which leads to temporary assignments of addresses. IP v6 is intended to remedy that scarcity, and eliminate the need to reassign addresses to accommodate scarcity. But IP v6 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 IP v6 addresses, they would constitute at best very inefficient handles, since many agents are voluntarily mobile. A mobile agent, using an IP v6 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.

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.

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 English or another natural languages 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 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 resolution of disputes, and reassignment of undefended names to recover their inherent value, 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.

2 Abstract Requirements

2.1 Technical Requirements

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. The system should allow all agents to create their own handles at will, possibly after an initial central registration of their first handle.
3. The system should support at least one trillion centrally 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.

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

5. The system should minimize the administrative work required by a central agency, and allow as much of that work as possible to be completely automated.

6. The system should respond to a handle 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.

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

8. The system, and its individual handles, should be capable of adapting for operation for 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.

9. 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 central agency, whenever it can be achieved by direct communication between querier and owner.

10. The system should provide protocols that allow it to be used uniformly, transparently, and seamlessly through centrally assigned vs. independently assigned handles. 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.

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

2.2 Social and Economical Requirements

1. A registrant should be able to assign handles to agents on request without taking any responsibility for the behavior of the agents.

2. The commercial value of top-level handles should be as low as possible. In particular, it should be very close to the inherent cost of registration. Ideally, it should be so low that a benevolent authority will perform registration at no charge.

3. 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.

4. The system should minimize the incentives for a single agent to hoard a large number of centrally assigned handles, and maximize the independent assignment of handles by agents. The main point is not to conserve storage by the central agency, 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.

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

6. 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.
2.3 Detailed Functional Requirements

These requirements are intended to be necessary consequences of the more abstractly expressed requirements above.