

## Identification Protocol

### Status of this Memo

This RFC specifies an IAB standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

### 1. INTRODUCTION

The Identification Protocol (a.k.a., "ident", a.k.a., "the Ident Protocol") provides a means to determine the identity of a user of a particular TCP connection. Given a TCP port number pair, it returns a character string which identifies the owner of that connection on the server's system.

The Identification Protocol was formerly called the Authentication Server Protocol. It has been renamed to better reflect its function. This document is a product of the TCP Client Identity Protocol Working Group of the Internet Engineering Task Force (IETF).

### 2. OVERVIEW

This is a connection based application on TCP. A server listens for TCP connections on TCP port 113 (decimal). Once a connection is established, the server reads a line of data which specifies the connection of interest. If it exists, the system dependent user identifier of the connection of interest is sent as the reply. The server may then either shut the connection down or it may continue to read/respond to multiple queries.

The server should close the connection down after a configurable amount of time with no queries - a 60-180 second idle timeout is recommended. The client may close the connection down at any time; however to allow for network delays the client should wait at least 30 seconds (or longer) after a query before abandoning the query and closing the connection.

### 3. RESTRICTIONS

Queries are permitted only for fully specified connections. The query contains the local/foreign port pair -- the local/foreign address pair used to fully specify the connection is taken from the local and foreign address of query connection. This means a user on address A may only query the server on address B about connections between A and B.

### 4. QUERY/RESPONSE FORMAT

The server accepts simple text query requests of the form:

`<port-on-server> , <port-on-client>`

where `<port-on-server>` is the TCP port (decimal) on the target (where the "ident" server is running) system, and `<port-on-client>` is the TCP port (decimal) on the source (client) system.

N.B - If a client on host A wants to ask a server on host B about a connection specified locally (on the client's machine) as 23, 6191 (an inbound TELNET connection), the client must actually ask about 6191, 23 - which is how the connection would be specified on host B.

For example:

`6191, 23`

The response is of the form

`<port-on-server> , <port-on-client> : <resp-type> : <add-info>`

where `<port-on-server>`, `<port-on-client>` are the same pair as the query, `<resp-type>` is a keyword identifying the type of response, and `<add-info>` is context dependent.

The information returned is that associated with the fully specified TCP connection identified by `<server-address>`, `<client-address>`, `<port-on-server>`, `<port-on-client>`, where `<server-address>` and `<client-address>` are the local and foreign IP addresses of the querying connection -- i.e., the TCP connection to the Identification Protocol Server. (`<port-on-server>` and `<port-on-client>` are taken from the query.)

For example:

`6193, 23 : USERID : UNIX : stjohns`  
`6195, 23 : ERROR : NO-USER`

## 5. RESPONSE TYPES

A response can be one of two types:

### USERID

In this case, <add-info> is a string consisting of an operating system name (with an optional character set identifier), followed by ":", followed by an identification string.

The character set (if present) is separated from the operating system name by ",". The character set identifier is used to indicate the character set of the identification string. The character set identifier, if omitted, defaults to "US-ASCII" (see below).

Permitted operating system names and character set names are specified in RFC 1340, "Assigned Numbers" or its successors.

In addition to those operating system and character set names specified in "Assigned Numbers" there is one special case operating system identifier - "OTHER".

Unless "OTHER" is specified as the operating system type, the server is expected to return the "normal" user identification of the owner of this connection. "Normal" in this context may be taken to mean a string of characters which uniquely identifies the connection owner such as a user identifier assigned by the system administrator and used by such user as a mail identifier, or as the "user" part of a user/password pair used to gain access to system resources. When an operating system is specified (e.g., anything but "OTHER"), the user identifier is expected to be in a more or less immediately useful form - e.g., something that could be used as an argument to "finger" or as a mail address.

"OTHER" indicates the identifier is an unformatted character string consisting of printable characters in the specified character set. "OTHER" should be specified if the user identifier does not meet the constraints of the previous paragraph. Sending an encrypted audit token, or returning other non-userid information about a user (such as the real name and phone number of a user from a UNIX passwd file) are

both examples of when "OTHER" should be used.

Returned user identifiers are expected to be printable in the character set indicated.

The identifier is an unformatted octet string - - all octets are permissible EXCEPT octal 000 (NUL), 012 (LF) and 015 (CR). N.B. - space characters (040) following the colon separator ARE part of the identifier string and may not be ignored. A response string is still terminated normally by a CR/LF. N.B. A string may be printable, but is not \*necessarily\* printable.

#### ERROR

For some reason the port owner could not be determined, <add-info> tells why. The following are the permitted values of <add-info> and their meanings:

##### INVALID-PORT

Either the local or foreign port was improperly specified. This should be returned if either or both of the port ids were out of range (TCP port numbers are from 1-65535), negative integers, reals or in any fashion not recognized as a non-negative integer.

##### NO-USER

The connection specified by the port pair is not currently in use or currently not owned by an identifiable entity.

##### HIDDEN-USER

The server was able to identify the user of this port, but the information was not returned at the request of the user.

##### UNKNOWN-ERROR

Can't determine connection owner; reason unknown. Any error not covered above should return this error code value. Optionally, this code MAY be returned in lieu of any other specific error code if, for example, the server desires to hide information implied by the return of that error

code, or for any other reason. If a server implements such a feature, it MUST be configurable and it MUST default to returning the proper error message.

Other values may eventually be specified and defined in future revisions to this document. If an implementer has a need to specify a non-standard error code, that code must begin with "X".

In addition, the server is allowed to drop the query connection without responding. Any premature close (i.e., one where the client does not receive the EOL, whether graceful or an abort should be considered to have the same meaning as "ERROR : UNKNOWN-ERROR".

#### FORMAL SYNTAX

```

<request> ::= <port-pair> <EOL>

<port-pair> ::= <integer> "," <integer>

<reply> ::= <reply-text> <EOL>

<EOL> ::= "015 012" ; CR-LF End of Line Indicator

<reply-text> ::= <error-reply> | <ident-reply>

<error-reply> ::= <port-pair> ":" "ERROR" ":" <error-type>

<ident-reply> ::= <port-pair> ":" "USERID" ":" <opsys-field>
                  ":" <user-id>

<error-type> ::= "INVALID-PORT" | "NO-USER" | "UNKNOWN-ERROR"
                  | "HIDDEN-USER" | <error-token>

<opsys-field> ::= <opsys> [ "," <charset> ]

<opsys> ::= "OTHER" | "UNIX" | <token> ...etc.
           ; (See "Assigned Numbers")

<charset> ::= "US-ASCII" | ...etc.
             ; (See "Assigned Numbers")

<user-id> ::= <octet-string>

<token> ::= 1*64<token-characters> ; 1-64 characters

<error-token> ::= "X"1*63<token-characters>
                  ; 2-64 chars beginning w/X

```

<integer> ::= 1\*5<digit> ; 1-5 digits.

<digit> ::= "0" | "1" ... "8" | "9" ; 0-9

<token-characters> ::=  
    <Any of these ASCII characters: a-z, A-Z,  
    - (dash), .!@#\$%^&\*()\_+=.,<>/?"'~`{ }[]; >  
        ; upper and lowercase a-z plus  
        ; printables minus the colon ":"  
        ; character.

<octet-string> ::= 1\*512<octet-characters>

<octet-characters> ::=  
    <any octet from 00 to 377 (octal) except for  
    ASCII NUL (000), CR (015) and LF (012)>

#### Notes on Syntax:

- 1) To promote interoperability among variant implementations, with respect to white space the above syntax is understood to embody the "be conservative in what you send and be liberal in what you accept" philosophy. Clients and servers should not generate unnecessary white space (space and tab characters) but should accept white space anywhere except within a token. In parsing responses, white space may occur anywhere, except within a token. Specifically, any amount of white space is permitted at the beginning or end of a line both for queries and responses. This does not apply for responses that contain a user ID because everything after the colon after the operating system type until the terminating CR/LF is taken as part of the user ID. The terminating CR/LF is NOT considered part of the user ID.
- 2) The above notwithstanding, servers should restrict the amount of inter-token white space they send to the smallest amount reasonable or useful. Clients should feel free to abort a connection if they receive 1000 characters without receiving an <EOL>.
- 3) The 512 character limit on user IDs and the 64 character limit on tokens should be understood to mean as follows: a) No new token (i.e., OPSYS or ERROR-TYPE) token will be defined that has a length greater than 64 and b) a server SHOULD NOT send more than 512 octets of user ID and a client MUST accept at least 512 octets of

user ID. Because of this limitation, a server MUST return the most significant portion of the user ID in the first 512 octets.

- 4) The character sets and character set identifiers should map directly to those defined in or referenced by RFC 1340, "Assigned Numbers" or its successors. Character set identifiers only apply to the user identification field - all other fields will be defined in and must be sent as US-ASCII.
- 5) Although <user-id> is defined as an <octet-string> above, it must follow the format and character set constraints implied by the <opsys-field>; see the discussion above.
- 6) The character set provides context for the client to print or store the returned user identification string. If the client does not recognize or implement the returned character set, it should handle the returned identification string as OCTET, but should in addition store or report the character set. An OCTET string should be printed, stored or handled in hex notation (0-9a-f) in addition to any other representation the client implements - this provides a standard representation among differing implementations.

## 6. Security Considerations

The information returned by this protocol is at most as trustworthy as the host providing it OR the organization operating the host. For example, a PC in an open lab has few if any controls on it to prevent a user from having this protocol return any identifier the user wants. Likewise, if the host has been compromised the information returned may be completely erroneous and misleading.

The Identification Protocol is not intended as an authorization or access control protocol. At best, it provides some additional auditing information with respect to TCP connections. At worst, it can provide misleading, incorrect, or maliciously incorrect information.

The use of the information returned by this protocol for other than auditing is strongly discouraged. Specifically, using Identification Protocol information to make access control decisions - either as the primary method (i.e., no other checks) or as an adjunct to other methods may result in a weakening of normal host security.

An Identification server may reveal information about users, entities, objects or processes which might normally be considered private. An Identification server provides service which is a rough analog of the CallerID services provided by some phone companies and many of the same privacy considerations and arguments that apply to the CallerID service apply to Identification. If you wouldn't run a "finger" server due to privacy considerations you may not want to run this protocol.

## 7. ACKNOWLEDGEMENTS

Acknowledgement is given to Dan Bernstein who is primarily responsible for renewing interest in this protocol and for pointing out some annoying errors in RFC 931.

## References

- [1] St. Johns, M., "Authentication Server", RFC 931, TPSC, January 1985.
- [2] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1340, USC/Information Sciences Institute, July 1992.

## Author's Address

Michael C. St. Johns  
DARPA/CSTO  
3701 N. Fairfax Dr  
Arlington, VA 22203  
  
Phone: (703) 696-2271  
EMail: stjohns@DARPA.MIL