Hyperbase Server Interface

Rosemary Andrada, CamMoore
Collaborative Software Development Laboratory
Department of Information and Computer Sciences
University of Hawaii
Honolulu, HI 96822
(808) 956-3489
CSDL-TR-93-18

June 14, 1994

1 Motivation

This interface document is intended for the use of CSDL to aid in our understanding of the relationship between the hyperbase server (HBS) and the client program. It is to act as the base for discussing improvements to the hyperbase server. This document is broken up into several sections. The first gives an overview of HBS. Then we discus how to connect to HBS followed by information about operations available in vers 2.0 and 2.1 respectively. The event mechanisms is explained. Then we preser on all possible function return values, node field values and operation end with some proposed changes.

2 Overview of the Hyperbase Server

Hyperbase is a multiuser, database manager for HyperText app base manages the nodes and links of the HyperText artifact a small set of generic operations. The HyperText clients tions for their own HyperText applications. The sr the bottleneck at the server.

We have expanded on the original set of have added an Append operation to append to show all users currently connected to gracefully brings down the server, a Name, and a Message passing facility now when a client disconnects onl DISCONNECT now contains the c

3 Connecting to HBS

The HBS connects to new clients through a pipe, a 'unique socket' whose default value is 10008. The request to connect is made by calling the Unix system call connect(). The HBS sends three, four-byte integers through the unique socket. These numbers are necessary to establish more pipes for specific types of communication. Herein, these pipes will be referred to as the server's write, read and event sockets. After each four-byte integer the server waits for the client to connect on tha first four-byte integer is the server's write socket. Information t send to the client will be sent through this socket. The secon HBS's read socket. All requests and information must be third four-byte integer is the event socket. Event on this socket. Once these three sockets are s user's name and then the name itself. The u socket and the user's name must end with a of the message to be sent with the e Next it reads that many characte

Figure 1: Pac

Figure 2: Packet Expe

[1] Note the only time a string must be terminated with a nam. All user names in HBS are 'null' terminated.

4 Available Operations: ver 3.0

4.1 Read

This operation reads the specified key value of a given entity number. If successful, it returns the key value through its write socket.

The HBS reads the operation READ (the four-byte integer 1) from its read sock
Next it reads the entity number (a four-byte integer) from its read socket.
reads the key number (a four-byte integer). Next it reads the length of t
be sent with the event (a four-byte integer). Next it reads that man
the read socket. The READ operation then sends out the return
integer) on the HBS's write socket. Then if the return valu
351 (Locked by other), it sends the length of the data (a f
socket. Next it sends that many characters to its

Figure 3: **Packet Expected**

Figure 4: **Packet Sent Out by Re**

4.2 Write

This operation writes information to a specific key of a given entity number.

The HBS reads the operation WRITE (the four-byte integer 2) from its read socket. Next it reads the entity number (a four-byte integer) from its read socket. Next it reads the key number (a four-byte integer). Next it reads from the read socket the length of the data to be written in the node (a four-byte integer). Next it reads that characters from the read socket. Next it reads the length of the message t with the event (a four-byte integer). Next it reads that many characters socket. The WRITE operation then sends out the return value (a fo on the HBS's write socket.

Figure 5: **Packet Expected by Write**

Figure 6: **Packet Returned by Write**

4.3 Create Node

This operation creates a new node. If successful, it returns the ID number of the newly created node. These node IDs are always even numbered and are reused after deletion.

The HBS reads the operation CREATE NODE (the four-byte integer 5) from its read socket. Next it reads the length of the message to be sent with the event (a four-byte integer). Next it reads that many characters from the read socket. The CREATE NODE operation then sends out the return value (a four-byte integer) on the HBS's write socket. If the return value is equal to 0 it sends the new entity num (a four-byte integer) to its write socket.

Figure 7: **Packet Expected by Create Node**

Figure 8: **Packet Sent Out by CreateNode**

4.4 Delete

This operation deletes the given entity (node or link). Deletion will fail for nodes if it
is linked with another node. Deletion will fail for links if its 'From' entity still p
to it. This operation ignores the fact that a node or link may be subscrib
the event mechanism or that a client may own a lock on it.
The HBS reads the operation DELETE (the four-byte integer 6) f
socket. Next it reads the entity number (a four-byte integer) fro
Next it reads the length of the message to be sent with the even
Next it reads that many characters from the read socket. T
sends out the return value (a four-byte integer) on the H

Figure 9: **Packet Expected by Del**

Figure 10: **Packet Returned by Delete**

4.5 Link

This operation is a remnant from the original hyperbase. A more complete version will
be available in HBS 2.1.x. This operation will do one of two things. First, if the 'To'
entity number is a node, then a link will be created and the ID number of the newly
created link will be returned. Note that this link points to the given node, but
node points to this link yet. To establish this connection, one would need to
operation again in the following context. If the 'From' entity number
the 'To' entity number is a link, then this operation establishes
the two. That is, the node has as one of its outgoing links t
The HBS reads the operation LINK (the four-byte in
Next it reads the 'From' entity number (a four-byte i
it reads the 'To' entity number (a four-byte int
the length of the message to be sent with
that many characters from the read
return value (a four-byte integer
is equal to 0 it sends the er
then a new **Link** is crea
returned.

Figure 12: **Packet S**

4.6 Move Link

This operation changes the destination node for an existing link.

The HBS reads the operation MOVE LINK (the four-byte integer 8) from its read socket. Next it reads the **Link** number (a four-byte integer) from its read socket. Next it reads the 'To' **Node** number (a four-byte integer) from its read socket. Next it reads the length of the message to be sent with the event (a four-byte integer). Next it re that many characters from the read socket. The MOVE LINK operation then send out the return value (a four-byte integer) on the HBS's write socket.

Figure 13: **Packet Expected by MoveLink**

Figure 14: **Packet Returned by MoveLink**

4.7 Remove Link

This operation destroys the conceptual connection between a given node and one of its outgoing links. Note that no nodes or links are deleted. Use Delete for deleting nodes and links.

The HBS reads the operation REMOVE_LINK (the four-byte integer 9) from its read socket. Next it reads the **Node** number (a four-byte integer) from its read socket. Next it reads the **Link** number (a four-byte integer) from its read socket. Next it reads the length of the message to be sent with the event (a four-byte integer). Next it that many characters from the read socket. The REMOVE_LINK operation then se out the return value (a four-byte integer) on the HBS's write socket.

Figure 15: **Packet Expected by ReMoveLink**

Figure 16: **Packet Returned by RemoveLink**

4.8 Event

This operation allows the client to subscribe to a particular event involving an entity
and the operation on that entity at a given key. Each combination of an event must
be subscribed to separately. However, it is possible to subscribe to mass entities
operations or keys using the 'ALL' code represented by the number 0.
The HBS reads the operation EVENT (the four-byte integer 10) from its
socket. Next it reads the entity number (a four-byte integer) from its read so
it reads the operation number (a four-byte integer). Next it reads the
integer). Next it reads the length of the message to be sent with
byte integer). Next it reads that many characters from the re
operation then sends out the return value (a four-byte int
socket.

Figure 17: **Packet Expected by Event**

Figure 18: **Packet Returned by Event**

4.9 UnEvent

This operation allows the client to unsubscribe himself from a particular event involving
an entity and the operation on that entity at a given key. The one restriction here is
that once a client subscribes to all nodes for any combination of operation or key,
can only unsubscribe himself from all nodes. Unsubscribing in this fashion
cause an error. It will simply be ignored.
The HBS reads the operation UNEVENT (the four-byte integer 11) fr
socket. Next it reads the entity number (a four-byte integer) from its r
it reads the operation number (a four-byte integer). Next it rea
integer). Next it reads the length of the message to be sent v
integer). Next it reads that many characters from the
operation then sends out the return value (a four-byt
socket.

Figure 19: Packet Expected by U

Figure 20: Packet Returned by UnEvent

4.10 Show Event

This operation displays a list of clients subscribing to a particular event involving an entity and the operation on that entity at a given key.

The HBS reads the operation SHOWEVENT (the four-byte integer 12) from its read socket. Next it reads the entity number (a four-byte integer) from its read socket. Next it reads the operation number (a four-byte integer). Next it reads the key nu (an integer). Next it reads the length of the message to be sent with the four-byte integer). Next it reads that many characters from the read SHOWEVENT operation then sends out the return value (an integer) o write socket. Then if the return value is equal to 0 it sends the len containing a list of users subscribed to the event (a four-byt that many characters to the write socket. Note that the more names delimited by a newline character.

Figure 21: **Packet Expected by Sh**

Figure 22: **Packet Sent Out by ShowEvent**

4.11 Lock

This operation allows the client to lock any writable key at the given entity. Once a client obtains a lock, no other client may perform the write operation on the locked key.

The HBS reads the operation LOCK (the four-byte integer 13) from its read socket. Next it reads the entity number (a four-byte integer) from its read socket. Next it rea the key number (a four-byte integer). Next it reads the length of the message sent with the event (a four-byte integer). Next it reads that many chara the read socket. The LOCK operation then sends out the return value integer) on the HBS's write socket.

Figure 23: **Packet Expected by Lock**

Figure 24: **Packet Returned by Lock**

4.12 UnLock

This operation allows the client to unlock a particular key at the given entity.

The HBS reads the operation UNLOCK (the four-byte integer 14) from its read socket. Next it reads the entity number (a four-byte integer) from its read socket. Next it reads the key number (a four-byte integer). Next it reads the length of the message to be sent with the event (a four-byte integer). Next it reads that many charac from the read socket. The UNLOCK operation then sends out the return valu four-byte integer) on the HBS's write socket.

Figure 25: **Packet Expected by UnLock**

Figure 26: **Packet Returned by UnLock**

4.13 Show Lock

This operation displays the client that currently holds a lock on a particular key at a given entity.

The HBS reads the operation SHOWLOCK (the four-byte integer 15) from its read socket. Next it reads the entity number (a four-byte integer) from its read socket. Next it reads the key number (a four-byte integer). Next it reads the length of the message to be sent with the event (a four-byte integer). Next it reads that many chara from the read socket. The SHOWLOCK operation then sends out the return va (a four-byte integer) on the HBS's write socket. Then it sends the length of t name or an error message (a four-byte integer) to the write socket. Next many characters to the write socket.

Figure 27: **Packet Expected by ShowLock**

Figure 28: **Packet Sent Out by ShowLock**

4.14 Disconnect

This operation disconnects the client from the server process. In doing so, the client is automatically unsubscribed from all events and all the client's locks are released.

The HBS reads the operation DISCONNECT (the four-byte integer 17) from its read socket. Next it reads the length of the message to be sent with the event four-byte integer). Next it reads that many characters from the read sock DISCONNECT operation then sends out the return value (a four-byte integer HBS's write socket.

Figure 29: **Packet Expected by Disconnect**

Figure 30: **Packet Returned by Disconnect**

4.15 Browse

This operation allows the client to obtain a list of all node or link IDs currently in the database. The client specifies the node type (data or link) and a list of nodes or links is returned respectively.

The HBS reads the operation BROWSE (the four-byte integer 18) from its rea socket. Next it reads the type of entity (a four-byte integer, 0 for Node, 1 for Next it reads the length of the message to be sent with the event (a four-by Next it reads that many characters from the read socket. The BROWS then sends out the return value (a four-byte integer) on the HBS's wri if the return value is equal to 0 it sends the length of the string integer). Next it sends that many four-byte integers f the write socket.

Figure 31: Packet Expected by

Figure 32: Packet Sent Out by Browse

4.16 Shut Down Server

This operation allows a client to shut down the server. However, the server will only shut down if the requesting client is the last client connected to the server or send an error code if unsuccessful.

The HBS reads the operation SHUTDOWN (the four-byte integer 19) from its read socket. The SHUTDOWN operation then sends out the return value on the HBS's write socket.

- This operation is a new addition of version 2.0

Figure 33: **Packet Expected by ShutDown**

Figure 34: **Packet Returned by ShutDown**

4.17 Show Users

This operation returns the number of clients currently connected followed by a list of names of those clients.

The HBS reads the operation SHOWUSERS (the four-byte integer 20) from its read socket. Next it reads the length of the message to be sent with the event (a four-byte integer). Next it reads that many characters from the read socket. The SHOW USERS operation then sends out the return value (a four-byte integer) on the HBS's write socket. Next HBS sends the user count (a four-byte integer) to its write soc For each user, it sends the length of the user name (a four-byte integer) and t characters to its write socket.

- This operation is a new addition of version 2.0

Figure 35: **Packet Expected by ShowUsers**

Figure 36: **Packet Sent Out by ShowUsers**

4.18 Append

This operation appends text to the data field in a given entity.

The HBS reads the operation APPEND (the four-byte integer 21) from its read socket. Next it reads the entity number (a four-byte integer) from its read socket. Next it reads from the read socket the length of the data to be appended to the node (a four-byte integer). Next it reads that many characters from the read socket. Next reads the length of the message to be sent with the event (a four-byte integer). reads that many characters from the read socket. The APPEND operation the out the return value (a four-byte integer) on the HBS's write socket.

- This operation is a new addition of version 2.0

Figure 37: **Packet Expected by Append**

Figure 38: **Packet Returned by Append**

4.19 Create Node with Name

This operation creates a new node and writes a given name to the name field. If successful, it returns the even-numbered ID of the newly created node.

The HBS reads the operation CREATE_NODE_W_NAME (the four-byte integer 24) from its read socket. Next it reads the length of the name (a four-byte integer) to be written to the name field. Then it reads in that many characters from the read socket. Next it reads the length of the message to be sent with the event (a four-byte integer). Next it read that many characters from the read socket. The CREATE_NODE_W_NAME operation then sends out the return value (a four-byte integer) on the HBS's write socket. If the return value is equal to 0 it sends the new entity number (a four-byte integer) to its write socket.

Figure 39: **Packet Expected by CreateNodeWithName**

Figure 40: **Packet Returned by CreateNodeWithName**

4.20 Create Link with Name

This operation creates a new link, writes a given name to its name field and establishes a connection with its 'From' node and its 'To' node. If successful, it returns the odd-numbered ID of the newly created link.

The HBS reads the operation CREATE_LINK_WNAME (the four-byte integer 25) from its read socket. Next it reads the 'From' entity number (a four-byte integer) from its read socket. Next it reads the 'To' entity number (a four-byte integer) from it read socket. Next it reads the length of the name (a four-byte integer) to be v to the name field. Next it reads that many characters from the read socket reads the length of the message to be sent with the event (a four-byte i it reads that many characters from the read socket. The CREATE_LI operation then sends out the return value (a four-byte integer) on th socket. Then if the return value is equal to 0 it sends the entity n created link.

Figure 41: **Packet Expected by Create Link with Name**

Figure 42: **Packet Returned by Create Link with Name**

4.21 Message

This operation allows a client to send a message to all clients that have subscri the events for the particular node and key.

The HBS reads the operation MESSAGE (the four-byte integer 26) fro socket. Next it reads the entity number (a four-byte integer), the k integer), the length of the message (a four-byte integer) and fi bytes (the message) from the read socket. The HBS creates a

sends it to all clients that have subscribed to events for the given node and key. The HBS returns OK through the write socket.

4.22 Connected

This operation tells the HBS to listen to other clients and end the special connect mode.

The HBS reads the operation CONNECTED (the four-byte integer 27) from the read socket. It then reads in the length of the message (a four-byte integer) and that many bytes (the message) from the read socket. The HBS returns OK on the write socket.

4.23 Create Node with Name and Data

This operation creates a new node, writes a given name to the name field and writes the given data to the data field. If successful, it returns the even-numbered ID of newly created node.

The HBS reads the operation CREATE_NAME_DATA (the four-byte integer 28) from its read socket. Next it reads the length of the name (a four-byte integer) written to the name field. Then it reads in that many characters from the read
Next it reads the length of the data to be placed in the node (a four-b
Next it read that many characters from the read socket. Next it re
the message to be sent with the event (a four-byte integer). N
characters from the read socket. The CREATE_NAME_DATA oper
out the return value (a four-byte integer) on the HBS's write sock
is equal to 0 it sends the new entity number (a four-byte i

4.24 Create Node with Name, Data and

This operation creates a new node, writes a given nam
given data to the data field and acquires the lock f
returns the even-numbered ID of the newly cr
The HBS reads the operation CREATE_NAME
ger 29) from its read socket. Next it reads the len
to be written to the name field. Then it reads
socket. Next it reads the length of the
integer). Next it read that many cha
length of the message to be sent w
many characters from the rea
then sends out the return value (a
acquires the lock on the data
entity number (a four-byt

4.25 Delete Link

This operation deletes a link and removes the reference to that link from the data node that points to the link.

The HBS reads in the operation DELETE_LINK (the four-byte integer 31) from the read socket. It then reads in the source node number (a four-byte integer), the link node number (a four-byte integer), the length of the message (a four-byte integer then message length bytes (the message) from the read socket. The HBS sends out return value (a four-byte integer) out the write socket.

4.26 Create a Link to a New Node

This operation creates a new node and then creates a new link to that new the supplied source node.

The HBS reads in the operation LINK_NEW_NODE (the four-byte inte from the read socket. It then read in the source node number (a four-by the length of the new node's name (a four-byte integer), that many node's name), the length of the new link's name (a four-byte intege (the new link's name), the length of the message (a four-byte i bytes (the message) all from the read socket. The HBS sen four-byte integer) out the write socket. If the retur successful) then the HBS sends out the new link node ID (a four-byte integer) to the write so

4.27 Wite and Unlock the No

This operation writes out to disk the field unlocks that field of the node.

The HBS reads the operation WRITE read socket. Next it reads the entity nu Next it reads the key number (a fo the length of the data to be that many characters fr be sent with the ev the read socket integer) on t

5 Events

Once a client subscribes to Events, by using the EVENT operation, the HBS will automatically send the client events. The HBS screens all activity in the Hyperbase and sends the client all events that the client has subscribed to. The HBS sends the events to the event socket for that client. The client then just has to read fromth socket to get the event. If the HBS sends more than one event to the client, the e just forma queue in the socket. The format of each event is as follows: name (a 'null' terminated string), the entity number on which the eve four-byte integer), the operation number (a four-byte integer), t operation occurred (a four-byte integer), the length of a mes the message fromthe user who performed the operation

Figure 43: **Event Packet sent out b**

6 Operations

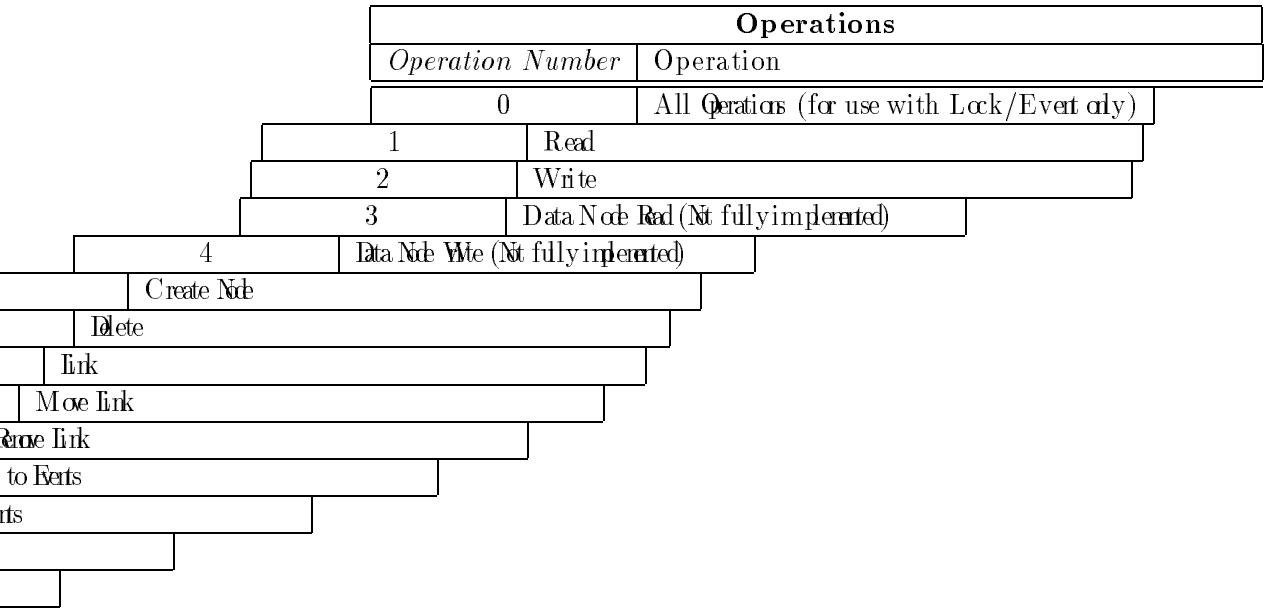
Figure 44 list all possible Operations

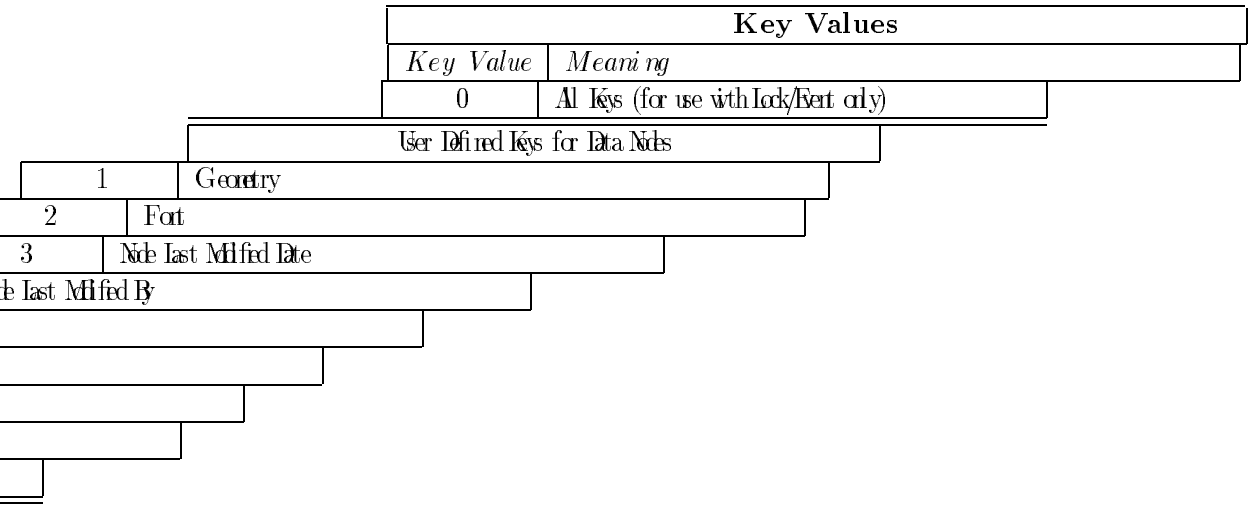
7 Key Values

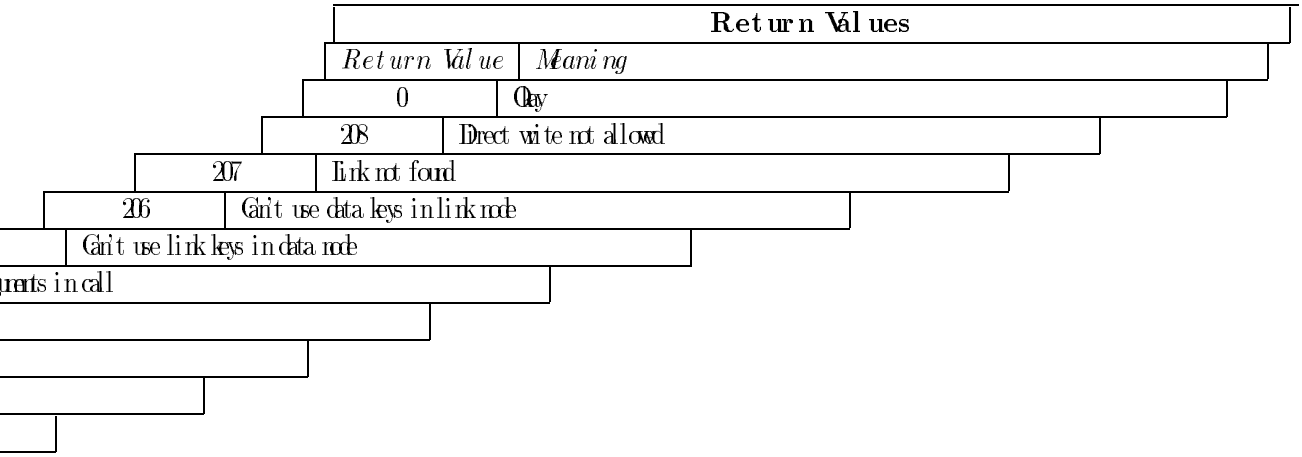
Figure 45 list all possible key values

8 Return Values

Figure 46 lists all possible return values


Operations
Operation Number
Operation
0
All Operations (for use with Lock/Event only)
1
Read
2
Write
3
Data Node Read (Not fully implemented)
4
Data Node Write (Not fully implemented)
Create Node
Delete
Link
Move Link
to Events


Key Values
Key Value
Meaning
0
All Keys (for use with Lock/Exert only)
User Defined Keys for Data Nodes
1
Geometry
2
Font
3
Node Last Modified Date
de Last Modified By


Return Values
Return Value
Meaning
0
Okay
208
Direct write not allowed
207
Link not found
206
Can't use data keys in link node
Can't use link keys in data node
uments in call

9 Proposed Changes

- Create an operation to enable/disable reuse of entity numbers
  - Allow storage of binary information

References

[1] Uffe Kock Wiil et al. Design and implementations of a hyperbase. Technical Report 90-03, Department of Mathematics and Computer Science, University of Aalborg, Denmark., 1990.