hotlinks.hyper

COLLABORATORS

	TITLE : hotlinks.hyper	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 6, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

hotlinks.hyper

1.1 hotlinks.doc

HLSysInfo()

HLRegister()

UnRegister()

AllocPBlock()

FreePBlock()

SetUser()

ChgPassword()

FirstPub()

NextPub()

RemovePub()

Notify()

PubStatus()

GetInfo()

SetInfo()

LockPub()

OpenPub()

ReadPub()

WritePub()

SeekPub()

```
                ClosePub()

                GetPub()

                PutPub()

                PubInfo()

                NewPassword()
```

1.2 hotlinks.library/HLSysinfo

```
  NAME
HLSysInfo - obtain information regarding the hotlinks system.

  SYNOPSIS
error = HLSysInfo(handle, array)
 d0               d0      d1

int HLSysInfo(ULONG, int *);

  FUNCTION
      Will fill array with useful information like memory used, number of
      editions, etc. The exact information to be returned has not been
      decided yet.

      This routine is not yet implemented so it will return UNIMPLEMENTED.

  INPUTS
      handle - the hotlinks 'handle' as returned by HLRegister.

      array - an array of long words that will be filled in by the routine.
              The exact format of this array has not been decided yet, other
              than to say that it will be an array of long words. But exactly
              how many entries are needed and what they each stand for has
              yet to be determined.

  RESULTS
      Will fill in the array with information about the hotlinks system.

      Possible information includes:
              user name
              hotlinks resident code version number
              number of local editions
              number of remote editions
              number of programs registered with hotlinks
              memory used by hotlinks

  BUGS
      None.
```

1.3 hotlinks.library/HLRegister

```
                NAME
   HLRegister - register a program with the hotlinks system.

SYNOPSIS
    handle = HLRegister(id, msgport, screen)
      d0                d0    d1       d2

    ULONG HLRegister(int, struct MsgPort *, struct Screen *);

FUNCTION
    Program must make this call to register themselves with HotLinks
    prior to making any calls to hotlinks.

    Handle must be freed before the registered program exits or memory
    will be lost.

INPUTS
    id - a four byte id to be used for as the creator id of any editions
       that this application may create.

    msgport - a pointer to a standard exec message port. This port is used
       to send Notify messages to the application when an edition changes.
       This may be NULL, if the application never calls
            Notify()
             to set up
       a notify on an edition.

    screen - a pointer to a valid screen where to application wants
       hotlinks to open up it's requester on. If this argument is NULL
       the default public screen is chosen (which is normally Workbench).

RESULTS
    handle - a handle to be maintained throughout the duration of the
       registered application.  This must be freed by calling
            UnRegister()
                       before the application exits or memory will be lost.

    INVPARAM - ?
    NOMEMORY - there was not enough memory to perform this function.

SEE ALSO

            UnRegister()
                BUGS
    None.
```

1.4 hotlinks.library/UnRegister

```
                NAME
    UnRegister - used to unregister an application with hotlinks.
```

```
SYNOPSIS
    error = UnRegister(handle)
     d0                d0

    int UnRegister(ULONG);

FUNCTION
    This call is used to free the handle allocated by
            HLRegister()
            . This
    routine must be called before the registered application exits or
    memory will be lost.

INPUTS
    handle - must be a valid handle returned from
            HLRegister()
            .

RESULTS
    NOERROR - if everything is OK.
    ?

SEE ALSO

            HLRegister()
                BUGS
    None.
```

1.5 hotlinks.library/AllocPBlock

```
                NAME
    AllocPBlock - allocate a publication block.

SYNOPSIS
    pblock = AllocPBlock(handle)
      d0                 d0

    struct PubBlock *AllocPBlock(ULONG);

FUNCTION
    This function will allocate a struct PubBlock and initialize it.
    This is the preferred method of allocating a PubBlock structure because
    if the size of the PubBlock structure changes your program will not
    have to be changed to work properly.

INPUTS
    handle - must be a valid handle as returned by
            HLRegister()
            .

RESULTS
    pblock - points to a valid initialized PubBlock struct.

    NOMEMORY - not enough memory to allocate a PubBlock struct.
```

```
    INVPARAM - the handle was invalid.

SEE ALSO

            FreePBlock()
            ,
            HLRegister()
                BUGS
    None.
```

1.6 hotlinks.library/FreePBlock

```
                NAME
    FreePBlock - frees a publication block obtained by
            AllocPBlock()
            .

SYNOPSIS
    error = FreePBLock(pblock)
     d0                 d0

    int FreePBlock(struct PubBlock *);

FUNCTION
    This will free the memory allocated by
            AllocPBlock()
             pointed to by
    pblock.

INPUTS
    pblock - must point to a valid PubBlock struct as returned by

            AllocPBlock()
            .

RESULTS
    INVPARAM - if the pblock was not a valid PubBlock pointer (NULL
        for example).

SEE ALSO

            AllocPBlock()
                BUGS
    None.
```

1.7 hotlinks.library/SetUser

```
                NAME
    SetUser - sets the current user on the hotlinks system.
```

```
SYNOPSIS
    error = SetUser(handle, name, password);
     d0              d0      a0     a1
    int SetUser(ULONG, char *, char *);

FUNCTION
    Will set the current user to Name, if Password is valid for Name's
    account.

INPUTS
    handle - must be a valid handle as returned by
            HLRegister()
            .
    name - a pointer to a NULL terminated string or NULL.
    password - a pointer to a NULL terminated string or NULL.

    If both the name and password are NULL then the current user is
    logged off the system. The next time a call to hotlinks is made the
    login prompt will be presented.

RESULTS
    INVPARAM - if password is not valid for name.

SEE ALSO

            HLRegister()
                BUGS
    None.
```

1.8 hotlinks.library/ChgPassword

```
NAME
    ChgPassword - allows the changing of a users password.

SYNOPSIS
    Error = ChgPassword(handle, name, oldpwd, newpwd);
     d0                   d0     a0       a1            a2

    int ChgPassword(ULONG, char *, char *, char *);

FUNCTION
    Changes the password of <name> from <oldpwd> to <newpwd>.

INPUTS
    name - a pointer to a NULL terminated string.
    oldpwd - a pointer to a NULL terminated string.
    newpwd - a pointer to a NULL terminated string.

    All values must be valid and not NULL, unless the current user is
    the superuser. As superuser, one must only supply the name and
    newpwd to change a regular user's password. If the superuser wants to
    change the superuser's password then all parameters must be valid and
    not NULL.
```

```
    The arguments need not be kept around for the duration of the program.
    The hotlinks resident code makes copies of the strings.

RESULTS
    INVPARAM - the oldpwd was not valid for name, or a parameter was NULL.
    NOPRIV - the current user tried to change another uses password.
    NOMEMORY - not enough memory to allocate the newpwd.

BUGS
    None.
```

1.9 hotlinks.library/FirstPub

```
                NAME
    FirstPub - fills in a PubBlock with the first publication's information

SYNOPSIS
    error = FirstPub(pblock)
     d0                d0

    int FirstPub(struct PubBlock *);

FUNCTION
    Returns the pblock structure filled in for first available edition.
    This works much like the dos.library Examine call in that it sets
    up for a series os
            NextPub()
             calls to look at all the editions the
    current user can gain access to.

INPUTS
    pblock - pointer to a valid PubBlock as returned by
            AllocPBlock()
            .

RESULTS
    NOERROR - pblock will be filled in with the information for the first
        edition file that the user can gain access to. Note that only the
        editions that the current user has access to will be peeked by
        this function all others will be skipped over.
    NOMOREBLOCKS - if there are no editions this user can access or if
        there are no editions at all.
    INVPARAM - if the pblock is invalid or NULL

SEE ALSO

            NextPub()
                BUGS
    None.
```

1.10 hotlinks.library/NextPub

```
                NAME
    NextPub - fills in a pubblock struct with the next pub's information.
              Valid only after a call to
            FirstPub()
            .

SYNOPSIS
    Error = NextPub(pblock)
     d0               d0

    int NextPub(struct PubBlock *);

FUNCTION
    Returns the pblock structure filled in for the next available edition.
    pblock must be the same pblock that was used in the call to
            FirstPub()
                    or the previous NextPub() call. This routine functions  ↩
                       much like the
    dos.library ExNext call. Repeated calls to this function will have
    the result of stepping through each of the available editions available
    to the currently logged in user.

INPUTS
    pblock - must be a valid PubBlock as returned by
            AllocPBlock()
             and
        processed by a call to
            FirstPub()
             or a previous call to NextPub().

RESULTS
    NOERROR - pblock will be filled in with the information for the next
        edition file that the user can gain access to. Note that only the
        editions that the current user has access to will be peeked by
        this function all others will be skipped over.
    NOMOREBLOCKS - there are no more editions available.
    INVPARAM - an invalid pblock was passed.

SEE ALSO

            FirstPub()
                BUGS
    None.
```

1.11 hotlinks.library/RemovePub

```
                NAME
    RemovePub - delete an edition

SYNOPSIS
    Error = RemovePub(pblock)
     d0                 d0
```

```
    int RemovePub(struct PubBlock *);

FUNCTION
    Will remove the edition file if the currently logged in user can do so.

    If an edition is deleted while applications still have links to it or
    notifies set up on it they will receive errors when those applications
    try to access any hotlinks function with that pblock. Normally this
    error will be INVPARAM.

INPUTS
    pblock - must be a valid PubBlock as returned by
            AllocPBlock()
            .
        pblock->PRec.ID[0], pblock->PRec.ID[1], and pblock->PRec.Version
        must be valid. The version number must be the latest version
        number or the call will fail.

RESULTS
    INVPARAM - the pblock was NULL or otherwise invalid (the version number
        was not the most recent).
    IOERROR - the dos.library DeleteFile() routine failed.
    CHANGED - ?

BUGS
    None.
```

1.12 hotlinks.library/Notify

```
                NAME
    Notify - set up a notify node for this application on this edition

SYNOPSIS
    Error = Notify(pblock, flag, class, userdata)
     d0              d0     d1     d2      a0

    int Notify(struct PubBlock *, int, int, void *);

FUNCTION
    This will cause a notify to be set up on the pblock. Anytime the
    edition file is changed by any application a message is sent to
    the message port specified in the
            HLRegister()
             call telling it so.

    The message sent is a struct HLMsg.

INPUTS
    pblock - must point to a valid PubBLock as returned by AllocPBLock().
    flag -
       INFORM - will set up a link to the edition file during which time
          if the document is changed. A message will be sent to the
          program indicating a publication has changed (and it's ID) via
          the message port specified in the
```

```
            HLRegister()
             call.

       EXINFORM - will set up a link to the edition file during which
          time if the document is changed. A message will be sent to the
          program indicating a publication has changed (and it's ID) via
          the message port specified in the
            HLRegister()
             call. Using this
          flag only 1 notify per edition per message port may be set up.

       NOINFORM - will cancel the notify request made on a previous call
          to Notify() with either INFORM or EXINFORM.

RESULTS
    NOERROR - a notify was set up on the edition file.
    INVPARAM - an invalid argument was passed.
    NOMEMORY - not enough memory to set up the notify.

SEE ALSO

            HLRegister()
                BUGS
    None.
```

1.13 hotlinks.library/PubStatus

```
                NAME
    PubStatus - checks to see if the edition has changed.

SYNOPSIS
    Error = PubStatus(pblock)
     d0               d0

    int PubStatus(struct PubBlock *);

FUNCTION
    This will check to see if the edition file has changed from the data
    contained in the pblock passed to it.


INPUTS
    pblock - must be a valid PubBlock as returned by
            AllocPBlock()
            .
       pblock->PRec.ID[0], pblock->PRec.ID[1], and pblock->PRec.Version
       must be valid.

RESULTS
    NOERROR - if the edition has not changed.
    CHANGED - if the edition has changed.
    INVPARAM - if the pblock is invalid or NULL.
    NOPRIV - if the currently logged in user cannot access the edition
        file specified by the pblock.
```

```
    No values in the pblock are modified by this call.

SEE ALSO

            AllocPBlock()
                BUGS
    None.
```

1.14 hotlinks.library/GetInfo

```
NAME
    GetInfo - fills in a pubblock struct with the information for the
              given id.

SYNOPSIS
    Error = GetInfo(pblock)
     d0               d0

    int GetInfo(struct PubBlock *);

FUNCTION
    This function will fill in a PubBlock with all the information for
    the requested edition file.

INPUTS
    pblock - must be a valid PubBlock with the pblock->PRec.ID fields
       filled in for the edition file you want the information for.

RESULTS
    NOERROR - the pblock is filled in with the edition's information.
    INVPARAM - an  invalid or NULL pblock was passed.
    NOPRIV - the currently logged in user does not have access to the
       requested edition file.

BUGS
    None.
```

1.15 hotlinks.library/SetInfo

```
                NAME
    SetInfo - will change the information for the publication to the new
              information as specified in the setinfo call.

SYNOPSIS
    Error = SetInfo(pblock)
     d0               d0

    int SetInfo(struct PubBlock *);

FUNCTION
```

```
    This function will reset all fields in the hotlinks internal database
    record for the edition file with the information from the pblock. All
    the fields will be changed to the data specified in the pblock. To
    change only a few fields, first make a call to the
            GetInfo()
             routine
    to fill the pblock with all the current information. Then make your
    changes and call SetInfo().

INPUTS
    pblock - must be a valid pblock filled in with the new information.

RESULTS
    INVPARAM - an invalid pblock was passed.

SEE ALSO

            GetInfo()
                BUGS
    None.
```

1.16 hotlinks.library/LockPub

```
NAME
    LockPub - locks an edition file for read or write access.

SYNOPSIS
    Error = LockPub(pblock, flags)
     d0               d0     d1

    int LockPub(struct PubBlock *, int);

FUNCTION
    This function will allow you to lock other applications out from being
    able to modify the edition file until you unlock it.

    This is a 'soft' lock in that it is only in effect while the hotlinks
    resident code is active. If the computer is turned off, all locks
    are lost.

INPUTS
    pblock - must be a valid PubBLock.
    flags -
        LOCK_RELEASE - will release the previously held lock.
        LOCK_READ - locks the edition file for reading. This is not an
            exclusive lock. So other applications can also gain read
            access to this file.
        LOCK_WRITE - locks the edition file for writing. This is an
            exclusive lock. No other application can get a lock on the
            edition file until the lock is released.

RESULTS
    NOERROR - got the lock with no problems.
    INVPARAM - a invalid parameter was passed to the function.
```

```
    INUSE - the edition file is locked by some one else.
    NOPRIV - the currently logged in user does not have access to the
       edition file asked for.
    CHANGED - ?

BUGS
    None.
```

1.17 hotlinks.library/OpenPub

```
NAME
    OpenPub - opens a publication file for reading/writing.

SYNOPSIS
    Error = OpenPub(pblock, flags)
     d0               d0     d1

    int OpenPub(struct PubBlock *, int);

FUNCTION
    Opens the edition file for read or write. No one else may read/write
    while the edition file is opened for writing. If opened for write,
    the version number is incremented and modified date and time are set.

INPUTS
    pblock - must point to a valid PubBlock.
        If pblock->PRec.ID[0] and [1] are 0, this means that a new edition
        file should be created. Call with the following parameters in
        PubBlock filled in:
        Type, Access, Name, Desc, Creator.
    flags -
        OPEN_READ - opens the edition file for reading.
        OPEN_WRITE - opens the edition file for writing.

RESULTS
    NOERROR - the pblock is filled in with the latest information
    NOMEMORY - not enough memory to carry out the open.
    INVPARAM - either the pblock was NULL or had invalid information in it,
       or the flags were incorrect.
    IOERROR - the dos.library Open() called failed.
    INUSE - the edition file is currently in use by another application.

BUGS
    None.
```

1.18 hotlinks.library/ReadPub

```
                NAME
    ReadPub - reads data from an edition file into a buffer.

SYNOPSIS
    numbytes = ReadPub(pblock, buffer, len)
```

```
      d0               d0      d1     d2

    int ReadPub(struct PubBlock *, char *, int);

FUNCTION
    This will read len bytes into buff from the edition file pointed to
    by the pblock.

INPUTS
    pblock - must be a valid pblock previously opened via
            OpenPub()
            .
    buffer - a pointer to a buffer at least len bytes in size.
    len - the number of bytes to read into buffer.

RESULTS
    numbytes - the actual number of bytes read.
    NOPRIV - the currently logged in user does not have access to the
      specified edition file.
    IOERROR - the dos.library Read() call failed.

SEE ALSO

            OpenPub()
            ,
            WritePub()
            ,
            SeekPub()
            ,
            ClosePub()
                BUGS
    None.
```

1.19 hotlinks.library/WritePub

```
                NAME
    WritePub - writes data to an edition file from a buffer.

SYNOPSIS
    error = WritePub(pblock, buffer, len)
     d0               d0      d1     d2

    int WritePub(struct PubBlock *, char *, int);

FUNCTION
    This will read len bytes into buff from the edition specified by
    pblock.

INPUTS
    pblock - a valid pblock previously opened via
            OpenPub()
            .
    buffer - a pointer to a buffer at least len bytes long from which
        data will be written to the edition file.
```

```
    len - the number of bytes to write to the edition file from buffer.

RESULTS
    NOERROR - no problems occurred while writing.
    NOPRIV - the currently logged in user does not have access to the
       requested edition file.
    INVPARAM - there was a problem with one of the arguments.
    IOERROR - the dos.library Write() failed or fewer bytes than requested
       were written to the edition.
    CHANGED - ?

SEE ALSO

            OpenPub()
            ,
            ReadPub()
            ,
            SeekPub()
            ,
            ClosePub()
                BUGS
    None.
```

1.20 hotlinks.library/SeekPub

```
                NAME
    SeekPub - Sets the current read/write position in the file.

SYNOPSIS
    position = SeekPub(pblock, offset, flags)
       d0                d0      d1     d2

    int SeekPub(struct PubBlock *, int, int);

FUNCTION
    Sets the current read/write position in the file. Will return the
    new position in the file relative to the beginning.

INPUTS
    pblock - must be a valid pblock, previously opened by
            OpenPub()
            .
    offset - the number of bytes to move.
    flags -
       SEEK_BEGINNING - the offset is from the start of the file.
       SEEK_CURRENT - the offset is from the current position.
       SEEK_END - the offset is from the end of the file.

RESULTS
    position - the new position in the file.

    IOERROR - the dos.library Seek() call failed.
    INVPARAM - one of the arguments was invalid.
    NOPRIV - the currently logged in user does not have access to the
```

```
        requested edition file.

SEE ALSO

            OpenPub()
            ,
            ReadPub()
            ,
            WritePub()
            ,
            ClosePub()
                BUGS
    None.
```

1.21 hotlinks.library/ClosePub

```
                NAME
    ClosePub - closes an edition file.

SYNOPSIS
    error = ClosePub(pblock)
     d0                d0

    int ClosePub(struct PubBlock *);

FUNCTION
    This will close the edition file that was opened via
            OpenPub()
            .

INPUTS
    pblock - must be a valid pblock previously opened via
            OpenPub()
            .

RESULTS
    NOERROR - the edition file closed without any problems.
    NOPRIV - the currently logged in user does not have access to the
       requested edition file.
    IOERROR - the dos.library Close() call failed.

SEE ALSO

            OpenPub()
            ,
            ReadPub()
            ,
            WritePub()
            ,
            SeekPub()
                BUGS
    None.
```

1.22 hotlinks.library/GetPub

```
                NAME
    GetPub - presents an edition requester (much like a file requester).

SYNOPSIS
    error = GetPub(pblock, filterproc)
     d0              d0        d1

    int GetPub(struct PubBlock *, int (*)());

FUNCTION
    Presents the user with an edition requester. This functions much like
    a file requester except it only shows hotlinks edition files.

    The requester will be opened on the screen specified in the

            HLRegister()
             call.

    Only the editions available to the currently logged in user will be
    shown in the requester.

INPUTS
    pblock - must be a valid pblock.
    filterproc - a pointer to a procedure that returns an integer in d0.
        The filterproc is called with a pointer to a PubBlock in a0. This
        allows the calling application to display only the editions in the
        edition requester it wants to. It can decide if the edition should
        be displayed by examining the PubBlock (passed in a0) and returning
        ACCEPT or NOACCEPT in d0. This is useful if the application only
        handles one type of edition file (ILBM, DTXT, etc.). If this
        argument is NULL, then all editions will be shown.

RESULTS
    on return - the pblock is filled in with the information for the
       edition the user selected.

    NOERROR - the data in the pblock is valid for the selected edition.

SEE ALSO

            PutPub()
                BUGS
    None.
```

1.23 hotlinks.library/PutPub

```
                NAME
    PutPub - presents a requester for a new edition file.

SYNOPSIS
    Error = PutPub(pblock, filterproc)
```

```
     d0              d0        d1

    int PutPub(struct PubBlock *, int (*)());

FUNCTION
    This presents the user with a new edition requester and allows
    them to edit the name, description, and access code.

    The requester will be opened on the screen specified in the

            HLRegister()
             call.

    Note that this does not create the edition file, the application must
    still call
            OpenPub()
             with the ID set to 0 for the edition to be
    created.

INPUTS
    pblock - must be a valid pblock returned by
            AllocPBlock()
            . The
       pblock->PRec.Name, pblock->PRec.Description, and
       pblock->PRec.Access will be shown to the user when the requester
       opens. This is used for the application to give some default
       values.
    filterproc - This argument is not utilized at this time and may be
       set to NULL.

RESULTS
    NOERROR - no problem.
    NOMEMORY - not enough memory to open the requester.
    IOERROR - ?

SEE ALSO

            GetPub()
            ,
            OpenPub()
                BUGS
    None.
```

1.24 hotlinks.library/PubInfo

```
NAME
    PubInfo - presents a requester with the information for the edition
              and allows the user to make changes.

SYNOPSIS
    error = PubInfo(pblock)
     d0               d0

    int PubInfo(struct PubBlock *);
```

```
FUNCTION
    Presents the user with the new publication requester with the fields
    filled out and allows the user to make changes and save them to the
    edition file.

INPUTS
    pblock - must be a valid pblock.

RESULTS
    NOERROR - the new information is saved to the edition file and changed
       in the pblock.

BUGS
    None.
```

1.25 hotlinks.library/NewPassword

```
                NAME
    NewPassword - presents the new password requester.

SYNOPSIS
    error = NewPassword(handle);
     d0                 d0

    int NewPassword(ULONG);

FUNCTION
    Presents the user with a requester that allows them to change
    passwords interactively.

    The requester will be opened on the screen specified in the

            HLRegister()
             call.

INPUTS
    handle - must be a valid handle as returned by
            HLRegister()
            .

RESULTS
    NOERROR - the new password has been successfully saved to the
        password file.

SEE ALSO

            ChgPassword()
                BUGS
    None.
```