amigaguide..hyper

COLLABORATORS

	TITLE : amigaguide..hyper		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		July 18, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

amigaguide..hyper

1.1 amigaguide.adoc

```
AddAmigaGuideHost()
AmigaGuideSignal()
CloseAmigaGuide()
ExpungeDataBases()
ExpungeXRef()
GetAmigaGuideMsg()
LoadXRef()
LockAmigaGuideBase()
LockE()
OpenAmigaGuide()
OpenAmigaGuideAsync()
OpenE()
RemoveAmigaGuideHost()
ReplyAmigaGuideMsg()
SendAmigaGuideCmd()
SendAmigaGuideContext()
SetAmigaGuideContext()
UnlockAmigaGuideBase()
SetHyperHook()
```

1.2 amigaguide.library/AddAmigaGuideHost

```
   NAME
AddAmigaGuideHost - Add a dynamic node host.

   SYNOPSIS
key = AddAmigaGuideHost ( hook, name, attrs);
d0          a0    d0     a1

AMIGAGUIDEHOST key;
struct Hook *hook;
STRPTR name;
struct TagItem *attrs;

   FUNCTION
```

This function adds a callback hook to the dynamic node list.

     INPUTS
hook - The callback hook.
name - Name of the AmigaGuideHost database that you are adding.
     The name must be unique.
attrs - Additional attributes. None are defined at this time.

     RETURNS
key - Key to be passed to RemoveAmigaGuideHost() to remove a
     dynamic node host from the system.

     EXAMPLE


```
/* Hook dispatcher */
ULONG __asm hookEntry(
  register __a0 struct Hook *h,
  register __a2 VOID *obj,
  register __a1 VOID *msg)
{
    /* Pass the parameters on the stack */
    return ((h->h_SubEntry)(h, obj, msg));
}


ULONG __saveds
dispatchAmigaGuideHost(struct Hook *h, STRPTR db, Msg msg)
{
    struct opNodeIO *onm = (struct opNodeIO *) msg;
    ULONG retval = 0;

    switch (msg->MethodID)
    {
  /* Does this node belong to you? */
  case HM_FindNode:
      {
    struct opFindHost *ofh = (struct opFindHost *) msg;

    kprintf("Find [%s] in %s\n", ofh->ofh_Node, db);

    /* Return TRUE to indicate that it's your node, otherwise
     * return FALSE. */
    retval = TRUE;
      }
      break;

  /* Open a node. */
  case HM_OpenNode:
      kprintf("Open [%s] in %s\n", onm->onm_Node, db);

      /* Provide the contents of the node */
      onm->onm_DocBuffer = TEMP_NODE;
      onm->onm_BuffLen = strlen(TEMP_NODE);

      /* Indicate that we were able to open the node */
      retval = TRUE;
      break;
```

```
   /* Close a node, that has no users. */
   case HM_CloseNode:
       kprintf("Close [%s] in %s\n", onm->onm_Node, db);

       /* Indicate that we were able to close the node */
       retval = TRUE;
       break;

   /* Free any extra memory */
   case HM_Expunge:
       kprintf("Expunge [%s]\n", db);
       break;

   default:
       kprintf("Unknown method %ld\n", msg->MethodID);
       break;
     }

     return (retval);
 }

 main(int argc, char **argv)
 {
     struct Hook hook;
     AMIGAGUIDEHOST hh;

     /* Open the library */
     if (AmigaGuideBase = OpenLibrary("amigaguide.library", 33))
     {
   /* Initialize the hook */
   hook.h_Entry = hookEntry;
   hook.h_SubEntry = dispatchAmigaGuideHost;

   /* Add the AmigaGuideHost to the system */
   if (hh = AddAmigaGuideHost(&hook, "ExampleHost", NULL))
   {
       /* Wait until we're told to quit */
       Wait(SIGBREAKF_CTRL_C);

       /* Try removing the host */
       while (RemoveAmigaGuideHost(hh, NULL) > 0)
       {
     /* Wait a while */
     Delay(250);
       }
   }

   /* close the library */
   CloseLibrary(AmigaGuideBase);
     }
 }


   SEE ALSO
 RemoveAmigaGuideHost()
```

1.3 amigaguide.library/AmigaGuideSignal

```
 NAME
AmigaGuideSignal – Obtain signal bit to Wait for an async AmigaGuide.

 SYNOPSIS
signal = AmigaGuideSignal ( handle );
d0              a0

ULONG signal;
AMIGAGUIDECONTEXT handle;

 FUNCTION
This function returns the signal bit to Wait on for AmigaGuideMsg's
for a particular AmigaGuide database.

 INPUTS
handle  – Handle to a AmigaGuide system.

 EXAMPLE

ULONG sigw, sigh;
AMIGAGUIDECONTEXT handle;

/* get the signal bit to wait on for a AmigaGuide message */
sigh = AmigaGuideSignal(handle);

/* add the signal bit into the total signals to wait on */
sigw |= sigh;

 RETURNS
signal  – Signal bit to Wait on.

 SEE ALSO
OpenAmigaGuideAsync(), GetAmigaGuideMsg(), ReplyAmigaGuideMsg()
```

1.4 amigaguide.library/CloseAmigaGuide

```
  NAME
CloseAmigaGuide – Close a AmigaGuide client.

  SYNOPSIS
CloseAmigaGuide ( handle );
     a0

AMIGAGUIDECONTEXT handle;

  FUNCTION
Closes a synchronous, or asynchronous, AmigaGuide client.

This function will also close all windows that were opened for
the client.

  INPUTS
```

```
handle  - Handle to an AmigaGuide client.

   SEE ALSO
OpenAmigaGuide(), OpenAmigaGuideAsync()
```

1.5 amigaguide.library/ExpungeDataBases

```
   NAME
ExpungeDataBases - Flush all buffers for all cached databases.

   SYNOPSIS
ExpungeDataBases ( flush );
       d0

BOOL flush;

   FUNCTION
Flush the buffers for each cached AmigaGuide database.

   INPUTS
flush - When set to TRUE, then the databases are also closed and
    removed from the cache list.

   SEE ALSO
OpenAmigaGuide(), OpenAmigaGuideAsync(), CloseAmigaGuide()
```

1.6 amigaguide.library/ExpungeXRef

```
   NAME
ExpungeXRef - Clear the cross reference table from memory.

   SYNOPSIS
ExpungeXRef();

   FUNCTION
This function unloads the cross reference table from memory.

   SEE ALSO
LoadXRef()
```

1.7 amigaguide.library/GetAmigaGuideMsg

```
  NAME
GetAmigaGuideMsg - Get messages from a particular AmigaGuide system.

 SYNOPSIS
msg = GetAmigaGuideMsg ( handle );
d0        a0

struct AmigaGuideMsg *msg;
```

```
AMIGAGUIDECONTEXT handle;

 FUNCTION
This function returns a SIPC message from the AmigaGuide system,
if there is a message available.

 INPUTS
handle  - Handle to a AmigaGuide system.

 EXAMPLE

AMIGAGUIDECONTEXT handle;
struct AmigaGuideMsg *agm;

/* get a AmigaGuide message */
while (agm = GetAmigaGuideMsg(handle))
{
    /* process the event */
    switch (agm->agm_Type)
    {
  case ToolCmdReplyID:  /* a command has completed */
      if (agm->agm_Pri_Ret)
      {
    /* An error occurred, the reason is in agm_Sec_Ret.
     * The command string is in agm_Data
     */
      }
      break;

  case ToolStatusID:  /* status message */
      if (agm->agm_Pri_Ret)
      {
    /* an error occurred, the reason is in agm_Sec_Ret */
      }
      break;

  default:
      break;
    }

    /* reply to the AmigaGuide message */
    ReplyAmigaGuideMsg(agm);
}

 RETURNS
msg - Pointer to a SIPC message or NULL if no message was
    available.

 SEE ALSO
OpenAmigaGuideAsync(), AmigaGuideSignal(), ReplyAmigaGuideMsg()
```

1.8 amigaguide.library/LoadXRef

```
   NAME
LoadXRef - Load a cross reference table.
```

```
   SYNOPSIS
success = LoadXRef(lock,name);
d0         a0   a1

LONG success;
BPTR lock;
STRPTR name;

   FUNCTION
This function loads the specified cross reference table from
disk into memory.

   INPUT
lock  - Lock on the directory that the file resides in.  May
    be NULL.
name  - Name of the file to load.

   RESULT
-1  - Indicates that the load was aborted by a ^C
0 - Failure to load.
1 - Successful load.
2 - Already loaded.

   SEE ALSO
ExpungeXRef()
```

1.9   amigaguide.library/LockAmigaGuideBase

```
   NAME
LockAmigaGuideBase - Lock an AmigaGuide client for exclusive access.

   SYNOPSIS
key = LockAmigaGuideBase ( handle );
d0          a0
LONG key
HYPERCONTEXT handle;

   FUNCTION
This functions locks an AmigaGuide client for exclusive access.

So far, there is no reason for developers to make calls to this
function, since all public functions already do the arbitration.

   INPUTS
handle  - Handle to an AmigaGuide client.

   RETURNS
key - Key to be passed to UnlockAmigaGuideBase() to unlock the
    client.

   SEE ALSO
UnlockAmigaGuideBase()
```

1.10   amigaguide.library/LockE

```
   NAME
LockE -- Search the given environment and lock a directory or file

   SYNOPSIS
lock  = LockE( path,  name, accessMode )
D0         A0     D1     D2

BPTR lock;
BPTR path;
STRPTR name;
LONG accessMode;

   FUNCTION
A filing system lock on the file or directory 'name' is returned if
possible.

If the accessMode is ACCESS_READ, the lock is a shared read lock;
if the accessMode is ACCESS_WRITE then it is an exclusive write
lock.  Do not use random values for mode.

If LockE() fails (that is, if it cannot obtain a filing system lock
on the file or directory) it returns a zero.

Tricky assumptions about the internal format of a lock are unwise,
as are any attempts to use the fl_Link or fl_Access fields.

   INPUTS
path    - pointer to a path list.  If this entry is NULL,
       then the path given in the AmigaGuide/Path environment
       variable is searched (this variable is only loaded
       at library open time).

       SetEnv AmigaGuide/Path "WORK:Docs/Libs WORK:Docs/Devs"

name    - pointer to a null-terminated string
accessMode - integer

   RESULTS
lock    - BCPL pointer to a FileLock.

   SEE ALSO
dos.library/Lock(), dos.library/UnLock(), dos.library/DupLock()
dos.library/ChangeMode(), dos.library/NameFromLock()
dos.library/DupLockFromFH()
```

1.11   amigaguide.library/OpenAmigaGuide

```
   NAME
OpenAmigaGuide - Open a synchronous AmigaGuide database.

   SYNOPSIS
handle = OpenAmigaGuide ( nag, control );
```

```
   d0         a0  a1
```

AMIGAGUIDECONTEXT handle;
struct NewAmigaGuide *nag;
struct MsgPort *control;

   FUNCTION
Opens a AmigaGuide database, complete with the first viewing
window, for synchronous activity.

Before you call OpenAmigaGuide(), you must initialize a NewAmigaGuide
structure. NewAmigaGuide is a structure that contains all the
information needed to open a database. The NewAmigaGuide structure
must be retained until the call returns.

The function will not return until the user closes all the
windows.

   INPUTS
nag – Pointer to an instance of a NewAmigaGuide structure. That
     structure is initialized with the following data.

     nag_Lock
     Lock on the directory that the database is located in.
     Not needed if nag_Name contains the complete path name.

     nag_Name
     Name of the AmigaGuide database.

     nag_Screen
     Screen to open the viewing windows on, NULL for the
     Workbench screen.

     nag_PubScreen
     Pointer to the name of the public screen to open on.
     Must already be opened.

     nag_HostPort
     Name of the applications' ARexx port (currently not used).

     nag_ClientPort
     Base name to use for the databases' ARexx port.

     nag_Flags
     Used to specify the requirements of this database. The
     flags are defined in <libraries/amigaguide.h>.

       HTF_LOAD_INDEX
       This flag only applies to an ansynchronous open.
       Force the index of the database to always be
       loaded. The AmigaGuide system maintains two date
       stamps, one for the last time that the database was
       opened and the other for the last time that the
       database was accessed by the user. The hyper system
       makes several calculations based on the current
       date stamp and the other two date stamps to
       determine what portions of the database need to be

```
    pre-cached.

    HTF_LOAD_ALL
    Load the entire database, and all its nodes into
    memory.

    HTF_CACHE_NODE
    Don't flush a node from memory after the user is
    finished viewing it.

    HTF_CACHE_DB
    Don't remove buffers when closed.  This will cause
    the buffers to remain until the library is
    expunged or the ExpungeDataBases() function is
    performed.

nag_Context
NULL terminated array of context nodes, in the form of:

/* context array */
STRPTR context[] =
{
    "MAIN",
    "INTRO",
    "GADGETS",
    NULL
};

The context array is not copied, but referenced,
therefore must remain static throughout the useage of
the AmigaGuide system.  This array is only referenced
when using the SetAmigaGuideContext() function.

nag_Node
Node to start at (does not work with OpenAmigaGuideAsync()).

nag_Line
Line to start at (does not work with OpenAmigaGuideAsync()).

nag_Extens
Used by V36 and beyond to pass additional arguments.

nag_Client
This is a private pointer, MUST be initialized to NULL.
```

control - SIPC control port. This field MUST always be set to
NULL.

   EXAMPLE

```
/* Short example showing synchronous AmigaGuide access */
LONG ShowAmigaGuideFile (STRPTR name, STRPTR node, LONG line)
{
    struct NewAmigaGuide nag = {NULL};
    AMIGAGUIDECONTEXT handle;
    LONG retval = 0L;
```

```
    /* Fill in the NewAmigaGuide structure */
    nag.nag_Name = name;
    nag.nag_Node = node;
    nag.nag_Line = line;

    /* Open the AmigaGuide client */
    if ( handle = OpenAmigaGuide(&nag, NULL))
    {
  /* Close the AmigaGuide client */
  CloseAmigaGuide(handle);
    }
    else
    {
  /* Get the reason for failure */
  retval = IoErr();
    }

    return (retval);
}
```

```
  RETURNS
handle  - Handle to a AmigaGuide system.

  SEE ALSO
OpenAmigaGuideAsync(), CloseAmigaGuide(), ExpungeDataBases()
```

1.12 amigaguide.library/OpenAmigaGuideAsync

```
 NAME
OpenAmigaGuideAsync - Open an asynchronous AmigaGuide database.

 SYNOPSIS
handle = OpenAmigaGuideAsync ( nag, attrs );
d0          a0   d0

AMIGAGUIDECONTEXT handle;
struct NewAmigaGuide *nag;
struct TagItem *attrs;

 FUNCTION
Opens an AmigaGuide database for ansynchronous use.

The NewAmigaGuide structure, and its pointers, must stay valid until an
ActiveToolID or ToolStatusID message is received by the calling
process.

This function actually spawns OpenAmigaGuide() as another process, so,
for further documentation, refer to the OpenAmigaGuide() function.

 INPUTS
nag - Pointer to a valid NewAmigaGuide structure.
    (see OpenAmigaGuide() for documentation on its useage).

future  - Future expansion, must be set to NULL for now.
```

```
  RETURNS
handle  - Handle to an AmigaGuide system.

  SEE ALSO
OpenAmigaGuide(), CloseAmigaGuide()
```

1.13   amigaguide.library/OpenE

```
   NAME
OpenE -- Search the given environemnt and open a file for IO.

   SYNOPSIS
file = OpenE ( path, name, accessMode )
D0           A0    D1    D2

BPTR file;
BPTR path;
STRPTR name;
LONG accessMode;

   FUNCTION
The named file is opened and a file handle returned.  If the
accessMode is MODE_OLDFILE, an existing file is opened for reading
or writing.  This call should not be used to create a file.

The 'name' can be a filename (optionally prefaced by a device
name), a simple device such as NIL:, a window specification such as
CON: or RAW: followed by window parameters, or "*", representing the
current window.  Note that as of V36, "*" is obsolete, and CONSOLE:
should be used instead.

If the file cannot be opened for any reason, the value returned
will be zero, and a secondary error code will be available by
calling the routine IoErr().

   INPUTS
path    - pointer to a path list.  If this entry is NULL,
       then the path given in the AmigaGuide/Path environment
       variable is searched (this variable is only loaded
       at library open time).

       SetEnv AmigaGuide/Path "WORK:Docs/Libs WORK:Docs/Devs"

name    - pointer to a null-terminated string
accessMode - integer

   RESULTS
file    - BCPL pointer to a FileHandle

   SEE ALSO
dos.library/Close(), dos.library/ChangeMode(),
dos.library/NameFromFH(), dos.library/ParentOfFH(),
dos.library/ExamineFH(), dos.library/Open()
```

1.14 amigaguide.library/RemoveAmigaGuideHostmigaguide.library/RemoveAmigaGuid

```
   NAME
RemoveAmigaGuideHost - Remove a dynamic node host.

   SYNOPSIS
use = RemoveAmigaGuideHost ( key, attrs)
d0       a0    a1

LONG use;
AMIGAGUIDEHOST key;
struct TagItem *attrs;

   FUNCTION
This function removes a dynamic node host, that was added by
AddAmigaGuideHost(), from the system.

   INPUTS
key - Key that was returned by AddAmigaGuideHost().
attrs - Additional attributes.  None are defined at this time.

   RETURNS
use - Number of outstanding clients of this database.  You
    can not exit until use==0.

   SEE ALSO
AddAmigaGuideHost()
```

1.15 amigaguide.library/ReplyAmigaGuideMsg

```
  NAME
ReplyAmigaGuideMsg - Reply to an AmigaGuide message.

  SYNOPSIS
ReplyAmigaGuideMsg ( msg );
        a0

struct AmigaGuideMsg *msg;

  FUNCTION
This function is used to reply to an AmigaGuide SIPC message.

  INPUTS
msg - Pointer to a SIPC message returned by a previous call to
    GetAmigaGuideMsg().

  SEE ALSO
OpenAmigaGuideAsync(), AmigaGuideSignal(), GetAmigaGuideMsg()
```

1.16 amigaguide.library/SendAmigaGuideCmd

```
 NAME
SendAmigaGuideCmd - Send a command string to a AmigaGuide system.

 SYNOPSIS
success = SendAmigaGuideCmd ( handle, cmd, future );
d0          a0    d0    d1

BOOL success;
AMIGAGUIDECONTEXT handle;
STRPTR cmd;
ULONG future;

 FUNCTION
This function sends a command string to an AmigaGuide system.  The
command can consist of any valid AmigaGuide action command.

The following are the currently valid action commands:

  ALINK <name>  Load the named node into a new window.

  LINK <name> Load the named node.

  RX <macro>  Execute an ARexx macro.

  RXS <command> Execute an ARexx string file.  To display a picture,
    use 'ADDRESS COMMAND DISPLAY <picture name>', to
    display a text file 'ADDRESS COMMAND MORE <doc>'.

  CLOSE   Close the window (should only be used on windows
    that were started with ALINK).

  QUIT    Shutdown the current database.

 INPUTS
handle  - Handle to an AmigaGuide system.
cmd - Command string.
future  - Future expansion, must be set to NULL for now.

 EXAMPLE

/* bring up help on a particular subject */
SendAmigaGuideCmd(handle, "LINK MAIN", NULL);

 RETURNS
success - Returns TRUE if the message was sent, otherwise returns
    FALSE.

 SEE ALSO
SetAmigaGuideContext()
```

1.17   amigaguide.library/SendAmigaGuideContextgaguide.library/SendAmigaGuideCor

```
 NAME
SendAmigaGuideContext - Align an AmigaGuide system on the context ID.
```

```
 SYNOPSIS
success = SendAmigaGuideContext ( handle, future );
d0            a0     d0

BOOL success;
AMIGAGUIDECONTEXT handle;
ULONG future;

 FUNCTION
This function is used to send a message to an AmigaGuide system to
align it on the current context ID.

This function effectively does a:

    SendAmigaGuideCmd(handle 'LINK ContextArray[contextID]', NULL);

 INPUTS
handle  - Handle to an AmigaGuide system.
future  - Future expansion, must be set to NULL for now.

 EXAMPLE

struct IntuiMessage *imsg;

...

case RAWKEY:
    switch (imsg->Code)
    {
  case 95:
      /* bring up help on a particular subject */
      SendAmigaGuideContext(handle, NULL);
      break;
  ...
    }
    break;

...

 RETURNS
success - Returns TRUE if the message was sent, otherwise returns
    FALSE.

 SEE ALSO
SetAmigaGuideContext(), SendAmigaGuideCmd()
```

1.18   amigaguide.library/SetAmigaGuideContextmigaguide.library/SetAmigaGuideConte

```
 NAME
SetAmigaGuideContext - Set the context ID for an AmigaGuide system.

 SYNOPSIS
success = SetAmigaGuideContext ( handle, context, future );
d0            a0     d0        d1
```

```
    BOOL success;
    AMIGAGUIDECONTEXT handle;
    ULONG context;
    ULONG future;

     FUNCTION
    This function, and the SendAmigaGuideContext() function, are used to
    provide a simple way to display a node based on a numeric value,
    instead of having to build up a slightly more complex command
    string.

     INPUTS
    handle  - Handle to an AmigaGuide system.
    context - Index value of the desired node to display.
    future  - Future expansion, must be set to NULL for now.

     EXAMPLE

    /* sample context table */
    STRPTR ContextArray[] =
    {
        "MAIN",
        "FILEREQ",
        "PRINT",
        "ABOUT",
        NULL
    };

    /* quickie defines */
    #define HELP_MAIN 0
    #define HELP_FILEREQ  1
    #define HELP_PRINT  2
    #define HELP_ABOUT  3

    ...

    struct NewAmigaGuide nag = {NULL};

    /* initialize the context table */
    nag.nag_Context = ContextArray;

    ...

    /* bring up help on a particular subject */
    SetAmigaGuideContext(handle, HELP_ABOUT, NULL);

     RETURNS
    success - Returns TRUE if a valid context ID was passed,
        otherwise returns FALSE.

     SEE ALSO
    SendAmigaGuideContext(), SendAmigaGuideCmd()
```

1.19 amigaguide.library/UnlockAmigaGuideBasemigaguide.library/UnlockAmigaGuideE

```
   NAME
UnlockAmigaGuideBase - Unlock a locked AmigaGuide client.

   SYNOPSIS
UnlockAmigaGuideBase ( key );
      d0
LONG key;

   FUNCTION
This functions unlocks a locked AmigaGuide client.

So far, there is no reason for developers to make calls to this
function, since all public functions already do the arbitration.

   INPUTS
key - Key returned by LockAmigaGuideBase().

   SEE ALSO
LockAmigaGuideBase()
```

1.20 hyper.library/SetHyperHook

```
 NAME
SetHyperHook - Initialize a callback hook.  *** OBSOLETE ***

 SYNOPSIS
success = SetHyperHook ( handle, kind, hook, data);
d0        a0  d0   a1    a2

LONG success;
HYPERCONTEXT handle;
LONG kind;
VOID (*hook)();
VOID **data;

 FUNCTION
This function is used to establish a client callback hook for
the HyperText library.

 INPUTS
handle  - Handle to a hypertext system.
kind  - Kind of hook to install.

  HTH_OPEN
  Function to be called to get access to a new node.

  HTH_CLOSE
  Function to be called to terminate access to a node.

hook  - A pointer to the function to call.  A NULL hook can
    be specified to turn off a hook.
```

data – A pointer to the data to pass to the function. Should
    be the same for each hook, but can be different.

 RETURNS
success – TRUE if the hook was installed, FALSE if there was
    a failure.

 BUGS
Don't use these functions. They are obsolete. Instead use the
new calls, AddHyperHost() and RemoveHyperHost().

 SEE ALSO
AddHyperHost(), RemoveHyperHost()