xref.doc

COLLABORATORS

	TITLE : xref.doc	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

xref.doc

1.1 xref.doc

```
--background--    AddXRefDynamicNode()
CloseXRefFile()   CreateXRefFileA()
FindXRefFile()    GetXRefBaseAttrsA()
GetXRefConfigDir()    GetXRefFileAttrsA()
LoadXRefPrefs()   LockXRefBase()
ParseXRef()   RemoveXRefDynamicNode()
SetXRefBaseAttrsA()   SetXRefFileAttrsA()
UnlockXRefBase()    WriteXRefFileEntryA()
XR_ExpungeXRef()    XR_LoadXRef()
```

1.2 xref.library/--background--

```
    PURPOSE
        the xref.library implements a pattern matching for xrefentries off
        all kinds.This is managed by categories and files.A category is a
        group of xreffiles, which have a relationship of a/some aspect(s).
        You can access a entry via a category or just an file. See the
        XREFA_#? defines for this. For example the following four files are
        grouped together in four main categories :
            1)  sys_autodoc.xref  -> CATEGORY SysAutoDoc
            2)  xref_autodoc.xref -> CATEGORY XRefAutoDoc
            3)  sys_include.xref  -> CATEGORY SysInclude
            4)  xref_include.xref -> CATEGORY XRefInclude

        The following categories matches the listed xreffiles :
                #?AutoDoc  -> 1,2
                #?Include  -> 3,4
                Sys#?      -> 1,3
                XRef#?     -> 2,4

    ENVIRONMENT
        The xref.library uses two environment variables. First the
        XREFCONFIGDIR for the configuration of the XRef-System. You can
        specify with this env-variable, where you have all configuration
```

```
    files for the XRef-System.
    The second XREFDIR is set from the library to have an access from
    the shell to the global xref directory ! This variable is changed,
    if you specify a new global xref directory via SetXRefBaseAttrs() !

PREFS
    the xref.library can be configured by a prefs file. If the library
    is opened the first time it looks for a file in xref.prefs. First it
    tries to get the directory from the XREFCONFIGDIR env-variable.
    If this doesn't exists it uses the "sys:config/xref" directory to
    open the xref.prefs file. If this file exists, it reads it via the
    LoadXRefPrefs() function.

SEE ALSO
    GetXRefConfigDir(), LoadXRefPrefs()
```

1.3 xref.library/AddXRefDynamicNode()

```
NAME
    AddXRefDynamicNode - adds the xref.library dynamic node host to the
                         amigaguide system

SYNOPSIS
    success = AddXRefDynamicNode();
      D0

    BOOL AddXRefDynamicNode(void);

FUNCTION
    adds the xref.library dynamic node host to the amigaguide system.
    This enables the xref.library to generate a main page with all
    referenced files actually in the system or for a specified
    category.

INPUTS
    none

DYNAMIC NODES
    The following names represent the dynamic nodes of the xref.library :

    xref.library_xreffile@main - This node displays all xreffiles,
        which are in memory.

    xref.library_xreffile@<filename> - This node displays all files in
        the xreffile specified by the <filename>.

    xref.library_xreftype@Generics - This node displays all entries of
        the XREFT_GENERIC type.

    xref.library_xreftype@Functions - This node displays all entries of
        the XREFT_FUNCTION type

    xref.library_xreftype@Commands - This node displays all entries of
        the XREFT_COMMAND type
```

```
    xref.library_xreftype@Includes - This node displays all entries of
        the XREFT_INCLUDE type

    xref.library_xreftype@Macros - This node displays all entries of
        the XREFT_MACRO type

    xref.library_xreftype@Structures - This node displays all entries of
        the XREFT_STRUCT type

    xref.library_xreftype@Typedefs - This node displays all entries of
        the XREFT_TYPEDEF type

    xref.library_xreftype@Defines - This node displays all entries of
        the XREFT_DEFINE type

RESULTS
    result TRUE for success otherwise FALSE, perhaps there is no
    amigaguide.library or something else

SEE ALSO
    RemoveXRefDynamicNode(), ParseXRef()
```

1.4 xref.library/CloseXRefFile()

```
NAME
    CloseXRefFile - closes and free's all allocated resources for a
        created xreffile via CreateXRefFileA()

SYNOPSIS
    CloseXRefFile(handle);
                  A0

    void CloseXRefFile(struct XRefFileHandle *);

FUNCTION
    this function closes the created xreffile and free's all resources
    allocated from CreateXRefFileA() and WriteXRefFileEntryA().
    Currently all entries are buffered, so only this function writes the
    data to the disk.

INPUTS
    handle (struct XRefFileHandle *) - handle pointer returned from
        CreateXRefFileA()

RESULTS
    none

SEE ALSO
    CreateXRefFileA(), WriteXRefFileEntryA()
```

1.5 xref.library/CreateXRefFileA()

```
NAME
    CreateXRefFileA - create a xreffile
    CreateXRefFile - varargs stub for CreateXRefFileA

SYNOPSIS
    handle = CreateXRefFileA(file,taglist);
      D0                     A0     A1

    handle = CreateXRefFile(file,tag1,...);

    struct XRefFileHandle *CreateXRefFileA(STRPTR ,struct TagItem *);

    struct XRefFileHandle *CreateXRefFile(STRPTR,Tag1,...);

FUNCTION
    this function creates an xreffile and allocates all needed resources.
    It returns a handle, which is used by the WriteXRefFileEntryA() and
    CloseXRefFile() calls. If you get a handle to the created xreffile ,
    you must call the CloseXRefFile() function. Otherwise the file isn't
    closed and no resources are free'd

TAGS
    XREFA_Priority (BYTE) - specifies a default priority, which is saved
        internally in the xreffile. If no priority is specified via
        XR_LoadXRef() function call, this priority is used.

    XREFA_File (STRPTR) - this tag overrides the file parameter

    XREFA_Category (STRPTR) - specifies a build-in category.This category
        can't be overwritten with a XR_LoadXRef() call.

    XREFA_Name (STRPTR) - defines a name for this xreffile.If this tag
        isn't set. The filename is used as the name.

    XREFA_VersTag (STRPTR) - this tag is used to write an AmigaDOS 2.0
        version-string.

    XREFA_Author (STRPTR) - specifies the author of this xreffile.

    XREFA_Path (STRPTR) - path to use for all entries in this xreffile.

INPUTS
    file (STRPTR) - file to create, if you specify only a filename
        without any path part, this file is created in the global xref-
        file directory.See XREFBA_XRefPath tag.
    taglist (struct TagItem *) - pointer to an array of tagitems.
    tag1 (Tag) - first tag used in varargs stub.

RESULTS
    handle (struct XRefFileHandle *) - handle pointer or NULL for an
        failure

SEE ALSO
    WriteXRefFileEntryA(), CloseXRefFile()
```

1.6 xref.library/FindXRefFile()

```
NAME
    FindXRefFile - find a xreffile with a given name

SYNOPSIS
    xreffile = FindXRefFile(name);
       D0                   A0

    struct XRefFileNode *FindXRefFile(STRPTR);

FUNCTION
    this function searches for xreffile with the given name. If there
    exists a xreffile with this name, it returns a pointer to it. This
    pointer is a handle for GetXRefFileAttrsA()/SetXRefFileAttrsA().

INPUTS
    name (STRPTR) - name to search for

RESULTS
    xreffile (struct XRefFileNode *) - pointer to the xreffile. This is
        only a handle for the appropriate functions !

SEE ALSO
    GetXRefFileAttrsA(), SetXRefFileAttrsA()
```

1.7 xref.library/GetXRefBaseAttrsA()

```
NAME
    GetXRefBaseAttrsA - get global xref.library attributes
    GetXRefBaseAttrs - varargs stub for GetXRefBaseAttrsA()

SYNOPSIS
    num = GetXRefBaseAttrsA(tagList);
     D0                      A0

    num = GetXRefBaseAttrs(Tag1,...);

    ULONG GetXRefBaseAttrsA(struct TagItem *);

    ULONG GetXRefBaseAttrs(ULONG ,...);

FUNCTION
    This function gets global attributes of the xref.library. You must
    pass a pointer to the appropriate type for each Tag in the ti_Data
    field. See TAGS for the attributes.

TAGS
    XREFBA_List (struct List *) - gets a pointer to the internal xref
```

```
        file list. If you want to step through the list you must lock
        the XRefBase via a LockXRefBase()/UnlockXRefBase() call pair.
        If you do not, the list may be corrupt, because another is
        changing the list via XR_LoadXRef()/XR_ExpungeXRef() calls.

    XREFBA_LineLength (UWORD) - gets the line length, which is used by
        the dynamic node to layout the page.

    XREFBA_Columns (UWORD) - gets the number of columns, which defines
        the layout of a page of a dynamic node

    XREFBA_DefaultLimit - (ULONG) gets default maximal number of matched
        xref entries by a ParseXRef() call.This value will be overwritten
        by the XREFA_Limit tag for the ParseXRef() function.

    XREFBA_XRefDir - (STRPTR) gets the default directory for all
        xreffiles.

INPUTS
    tagList (struct TagItem *) - pointer to a TagItem array
    Tag1 (ULONG) - first tag in the varargs stub

RESULTS
    num (ULONG) - number of tags processed

SEE ALSO
    SetXRefBaseAttrsA(),LoadXRefPrefs(),LockXRefBase(),UnlockXRefBase()
```

1.8 xref.library/GetXRefConfigDir()

```
NAME
    GetXRefConfigDir - returns the xref configuration path

SYNOPSIS
    GetXRefConfigDir(buffer,size);
                      A0    D0

    void GetXRefConfigDir(STRPTR ,ULONG );

FUNCTION
    returns in the given buffer , the current xref configuration
    directory. If the buffer would overflow , it is truncated. Thus
    you should provide a buffer of a appropriate length for a path !

INPUTS
    buffer (STRPTR) - pointer to a memory area to hold the path
    size (ULONG) - number of bytes of the memory area

RESULTS
    none

SEE ALSO
```

1.9 xref.library/GetXRefFileAttrsA()

```
NAME
    GetXRefFileAttrsA - get some attributes of a given xreffile
    GetXRefFileAttrs - varargs stub for GetXRefFileAttrsA()

SYNOPSIS
    num = GetXRefFileAttrsA(xreffile,tagList);
     D0                     A0      A1

    num = GetXRefFileAttrs(xreffile,tag1,...);

    ULONG GetXRefFileAttrsA(struct XRefFileNode *,struct TagItem *);

    ULONG GetXRefFileAttrs(struct XRefFileNode *,Tag ,...);

FUNCTION
    this function gets information about a given xreffile. The handle
    for the xreffile can you get via a call to FindXRefFile() or by
    stepping through the global xreffile list, which you can get with
    the GetXRefBaseAttrsA() function. This is the only way to get infor-
    mation about a xreffile without the name, which can be accessed in
    the ln_Name field of the Node (Perhaps only for displaying in a
    ListView).
    NOTE: You must provide a pointer to the appropriate type for each tag
    in the ti_Data field of the tagitem array.

TAGS
    XREFA_Category (STRPTR) - category of this xreffile

    XREFA_Index (BOOL) - indicates , if there exists an index array
        for all entries of this xreffile or not.

    XREFA_Priority (BYTE) - priority of this xreffile

    XREFA_Lock (BOOL) - returns, if this xreffile is locked or not.

    XREFA_Name (STRPTR) - returns a pointer to the xreffile name.

    XREFA_VersTag (STRPTR) - returns a pointer to the version string of
        this xreffile or NULL.

    XREFA_Path (STRPTR) - returns the global path for all entries in this
        xreffile.

    XREFA_Length (ULONG) - returns the number of bytes used for this
        xreffile.

INPUTS
    xreffile (struct XRefFileNode *) - handle for the xreffile
    tagList (struct TagItem *) - tag array with tags you wish information
        for.
    tag1 (ULONG) first tag in the varargs stub.

RESULTS
    num (ULONG) - number of tags processed
```

```
SEE ALSO
    FindXRefFile(), SetXRefFileAttrsA()
```

1.10 xref.library/LoadXRefPrefs()

```
NAME
    LoadXRefPrefs - load a prefs file for the xref.library

SYNOPSIS
    success = LoadXRefPrefs(file);
       D0                   A0

    ULONG LoadXRefPrefs(STRPTR );

FUNCTION
    loads the specified prefs file and sets up the xref.library
    attributes to the values given in this file.The file must be an ASCII
    text, which is line oriented.Each line is interpreted to one of the
    following templates :
        - AUTOOPEN/S,FILE/K/A,XREFPRI/N/K,LOCK/S,INDEX/S
        - AUTOLOAD/S,FILE/K/A,XREFPRI/N/K,LOCK/S,INDEX/S
        - MAINPAGE/S,LINELENGTH/N/K/A,COLUMNS/N/K/A
        - CLEANUP/S,FORCE/S
        - XREFDIR/S,DIR/K/A

    The AUTOOPEN-Line :

    This type of an entry may exists several times in a prefs file.
    It specifies the xreffile, which is opened during parsing of this
    file. The specified arguments are passed to the XR_LoadXRef()
    function, thus see this doc.

    The AUTOLOAD-Line :

    This type specifies xreffiles, which should be automatically loaded,
    if an entry wasn't found in the xreffiles in memory !

    The MAINPAGE-Line :

    This entry specifies the line length and the column number, which
    will be used for the "XRef-Library Main Page" and all dynamic nodes.
    Thus you can set these values appropriate to your screen/window
    resolution. This works only under V40 and above. In V39 and below all
    dynamic nodes have just one entry per line !.

    The CLEANUP-Line :

    If this line appears in a prefs file, all xreffiles are flushed
    form the memory. And if you specify the FORCE switch, all locked
    xreffiles are also flushed.

    The XREFPATH-Line :
```

```
    If you specify this line the path is used as the default xreffile
    path.

INPUTS
    file (STRPTR) - prefs file to load

RESULTS
    TRUE for success , FALSE to indicate an error (see IoErr() result for
    the reason) !

SEE ALSO
    XR_LoadXRef(), XR_ExpungeXRef()
```

1.11 xref.library/LockXRefBase()

```
NAME
    LockXRefBase - locks the base of the xref.library

SYNOPSIS
    handle = LockXRefBase(key);
      D0                  D0

    ULONG LockXRefBase(ULONG);

FUNCTION
    This function locks the xref.library base. The key specifies, which
    internal resource of the xref.library should be locked. At the moment
    there is only a value of zero valid. This locks the whole library
    base. But in future it may exists more different resource types, thus
    you must pass zero at this time, to be compatible in future !!!

INPUTS
    key (ULONG) - key , which specify the internal resource to lock. Only
        zero is valid now.

RESULTS
    handle (ULONG) - handle to pass to the UnlockXRefBase() call

SEE ALSO
    UnlockXRefBase(), GetXRefBaseAttrsA(), SetXRefBaseAttrsA()
```

1.12 xref.library/ParseXRef()

```
NAME
    ParseXRef - parses the xref lists for a given pattern or name
    ParseXRefTags - varargs stub for ParseXRef

SYNOPSIS
    success = ParseXRef(string,tagList);
     D0                 A0     A1
```

```
    success = ParseXRefTags(string,tag1,...);

    ULONG ParseXRef(STRPTR ,struct TagItem *);

    ULONG ParseXRefTags(STRPTR ,Tag ,...);

FUNCTION
    this function parses the specified xref lists for a given pattern
    or name, according to the XREFA_Matching tag value. If this function
    finds a xref entry, the callback function provided by XREFA_ParseHook
    is called with PM_XREF message. This callback function is called in
    2.0 standard Hook way :
       func(REGA0 struct Hook *hook,REGA2 struct XRefFileNode *xrnode,
            REGA1 struct pmXRef *msg);

INPUTS
    string (STRPTR) - string or pattern to search for
    tagList (struct TagItem *) - TagItem list with required/optional
        arguments

TAGS
    XREFA_XRefHook (struct Hook *) - pointer to a hook structure with
        the callback function, which is called if a xref entry matches
        or a new xref file or file is parsed.If you return a non zero
        value, the parse is suspended, if you return zero the parse
        goes on. The hook function must support at least the XRM_XREF
        message. Currently the following attributes are used in the
        msg->xref_Attrs field :

            ENTRYA_Type, ENTRYA_Name, ENTRYA_Line, ENTRYA_File,
            ENTRYA_NodeName, XREFA_Path, XREFA_Name.

        THIS TAG IS REQUIRED !

    XREFA_Category (STRPTR) - category pattern string to specify the
        category for the search. If no category is specified all
        categories matches.

    XREFA_CategoryParsed (STRPTR) - a parsed category pattern using the
        ParsePatternNoCase() function. This will speedup parsing, if you
        call this function a lot with the same category pattern.

    XREFA_File (STRPTR) - file to parse. The name must be a real path,
        because the comparision is made by the lock to the file.This
        tag overrides the XREFA_Category tag.

    XREFA_Limit (ULONG) - maximal number of matched xref entries. This
        tag overwrites the DefaultLimit of the library base.

    XREFA_Matching (ULONG) - mode for the matching mechanism one of the
        following modes should specified :
          XREFMATCH_PATTERN_CASE       - uses MatchPattern() (default)
          XREFMATCH_PATTERN_NOCASE     - uses MatchPatternNoCase()
          XREFMATCH_COMPARE_CASE       - uses strcmp()
          XREFMATCH_COMPARE_NOCASE     - uses Stricmp()
          XREFMATCH_COMPARE_NUM_CASE   - uses strncmp(string,xrefentry,
```

```
                                                        strlen(string))
            XREFMATCH_COMPARE_NUM_NOCASE - uses Strnicmp(string,xrefentry,
                                                         strlen(string))

    XREFA_AcceptTypes (ULONG *) - ~0 terminated array, which defines
        explicitly the types, which should be used. For example the
        array ULONG mytypes[] = {XREFT_GENERIC,XREFT_FUNCTION,~0} accept
        only entries of the these two types.

    XREFA_RejectTypes (ULONG *) - ~0 terminated array, which defines
        types which should not be used. For example the array
        ULONG mytypes2[] = {XREFT_TYPEDEF,XREFT_FIELD,~0} excludes these
        types.

RESULTS
    TRUE for success, that the function has parsed all xref nodes with
    the given parameters. FALSE for an error code see IoErr() to get the
    reason.

NOTES
    this function needs at least the tag XREFA_ParseHook, to get an valid
    hook function pointer to call for any messages.

SEE ALSO
    XR_LoadXRef(), dos.library/ParsePattern(), dos.library/MatchPattern()

    dos.library/ParsePatternNoCase(), dos.library/MatchPatternNoCase(),
    utility.library/Stricmp(), utility.library/Strnicmp(), c.lib/strcmp()

    c.lib/strncmp()
```

1.13 xref.library/RemoveXRefDynamicNode()

```
NAME
    RemoveXRefDynamicNode - removes the xref.library dynamic node host

SYNOPSIS
    success = RemoveXRefDynamicNode();

    BOOL RemoveXRefDynamicNode(void);

FUNCTION
    this removes the previously installed xref.library dynamic node host
    from the amigaguide system.

INPUTS
    none

RESULTS
    TRUE , if the dynamic node is removed. FALSE if not.

SEE ALSO
    AddXRefDynamicNode()
```

1.14 xref.library/SetXRefBaseAttrsA()

```
NAME
    SetXRefBaseAttrsA - set global attributes of the xref.library
    SetXRefBaseAttrs - varargs stub for SetXRefBaseAttrsA()

SYNOPSIS
    num = SetXRefBaseAttrsA(tagList);
     D0                     A0

    num = SetXRefBaseAttrs(Tag1,...);

    ULONG SetXRefBaseAttrsA(struct TagItem *);

    ULONG SetXRefBaseAttrs(ULONG,...);

FUNCTION
    This function sets some global attributes of the xref.library.
    Which attributes are changeable see TAGS.

TAGS
    XREFBA_LineLength - (UWORD) sets number of characters per line for
        the dynamic node. In conjunction with the XREFBA_Columns
        attribute you can specify your individual page layout.

    XREFBA_Columns - (UWORD) sets number of columns for the dynamic node.
        If you specify for example 3 columns and 90 chars per line, the
        dynamic node uses 3 columns with 30 characters.

    XREFBA_DefaultLimit - (ULONG) sets default maximal number of matched
        xref entries by a ParseXRef() call.This value will be overwritten
        by the XREFA_Limit tag for the ParseXRef() function.

    XREFBA_XRefDir - (STRPTR) sets the default dir for all xreffiles.
        This dir is used as the current directory, during the
        XR_LoadXRef() call.Thus if you use only filenames without any
        dir parts all xreffile can be placed in this directory.
        This tag changes the XREFDIR env variable !

INPUTS
    tagList (struct TagItem *) - pointer to a TagItem array
    Tag1 (ULONG) - first tag in the varargs stub

RESULTS
    num (ULONG) - number of tags processed

NOTE
    You don't need to call LockXRefBase()/UnlockXRefBase() pair. This
    is done by this function internally.

SEE ALSO
    GetXRefBaseAttrsA(), LoadXRefPrefs()
```

1.15 xref.library/SetXRefFileAttrsA()

```
NAME
    SetXRefFileAttrsA - set xreffile attributes
    SetXRefFileAttrs - varargs stub for SetXRefFileAttrsA

SYNOPSIS
    num = SetXRefFileAttrsA(xreffile,taglist);
     D0                     A0      A1

    num = SetXRefFileAttrs(xreffile,tag1,...);

    ULONG SetXRefFileAttrsA(struct XRefFileNode *,struct TagItem *);

    ULONG SetXRefFileAttrs(struct XRefFileNode *,Tag ,...);

FUNCTION
    this function sets some attributes of the given xreffile. The handle
    for the xreffile can you get via a call to FindXRefFile() or by
    stepping through the global xreffile list, which you can get with
    the GetXRefBaseAttrsA() function.
    The only way to change attributes of a xreffile is to call this
    function ! See TAGS for the attributes to be changed.

TAGS
    XREFA_Priority (BYTE) - new priority of this xreffile

    XREFA_Index (BOOL) - if set to TRUE, this function tries to allocate
        an index buffer and build an index for binary search.If this
        fails, this tag is interpreted as not processed.
        If you set this tag to FALSE, a previously allocated index buffer
        is deallocated.

    XREFA_Lock (UWORD) - if set to XREF_LOCK, this xreffile is locked,
        thus it isn't removed during a memory flush. If set to
        XREF_UNLOCK, the lock is removed.

INPUTS
    xreffile (struct XRefFileNode *) - handle for the xreffile
    tagList (struct TagItem *) - tag array with tags you wish to set
    tag1 (ULONG) - first tag in the varargs stub.

RESULTS
    num (ULONG) - number of tags processed

SEE ALSO
    FindXRefFile(), GetXRefFileAttrsA()
```

1.16 xref.library/UnlockXRefBase()

```
NAME
    UnlockXRefBase - unlocks the xref.library base

SYNOPSIS
    UnlockXRefBase(handle);
                   D0

    void UnlockXRefBase(ULONG);

FUNCTION
    This function unlocks the xref.library base, which was locked via a
    previously LockXRefBase() call. You must pass the handle you get from
    the LockXRefBase() call.This handle specifies the internal resource
    to unlock.

INPUTS
    handle (ULONG) - handle from a call to LockXRefBase()

RESULTS
    none

SEE ALSO
    LockXRefBase(), GetXRefBaseAttrsA(), SetXRefBaseAttrsA()
```

1.17 xref.library/WriteXRefFileEntryA()

```
NAME
    WriteXRefFileEntryA - writes a xref entry in the xreffile
    WriteXRefFileEntry - varargs stub for WriteXRefFileEntryA

SYNOPSIS
    success = WriteXRefFileEntryA(handle,taglist);
      D0                          A0     A1

    success = WriteXRefFileEntry(handle,tag1,...);

    ULONG WriteXRefFileEntryA(struct XRefFileHandle *,struct TagItem *);

    ULONG WriteXRefFileEntry(struct XRefFileHandle *,Tag ,...);

FUNCTION
    this function writes a xref entry in the specified xreffile. The
    entries are fully buffered at the moment, so the entries are only
    written to disk, if you call CloseXRefFile().The entries are sorted
    in case-sensetive mode. The file table are build automatically.
    If in the current xreffile exists a entry with the exact ENTRY_Name
    and ENTRY_File, the function fails with the IoErr() set to
    ERROR_OBJECT_EXISTS !

TAGS
    ENTRYA_Type (ULONG) - specifies the type of the entry. This tag is
        REQUIRED. Valid types for now are :
            XREFT_GENERIC - generic (AmigaGuide node)
```

```
            XREFT_FUNCTION - a library function
            XREFT_COMMAND - a device command
            XREFT_INCLUDE - a include file
            XREFT_MACRO - a define macro
            XREFT_STRUCT - a structure definition
            XREFT_FIELD - a field name of a structure
            XREFT_TYPEDEF - a typedef
            XREFT_DEFINE - a normal define

    ENTRYA_Name (STRPTR) - specifies the name of the entry. This tag is
        REQUIRED !

    ENTRYA_File (STRPTR) - specifies the relative path for the file
        including the filename.

    ENTRYA_Line (ULONG) - specifies the line number in the file.

    ENTRYA_NodeName (STRPTR) - specifies the amigaguide node name. If
        this isn't set the ENTRYA_Name will be used for the nodename.

    ENTRYA_CheckMode (ULONG) - one of the following modes :
            ENTRYCHECK_NONE : include the entry without any checking
            ENTRYCHECK_NAME : include the entry only, if no entry exists
                              with this name
            ENTRYCHECK_FILE : include the entry, if there exists a entry
                              with this name, but in a different file !
        default is : ENTRYCHECK_NAME !

INPUTS
    handle (struct XRefFileHandle *) - handle pointer returned from
        CreateXRefFileA()
    taglist (struct TagItem *) - pointer to th e first element of the
        tagitem array
    tag1 (Tag) - first tag in the varargs stub

RESULTS
    success (ULONG) - TRUE for success , FALSE for a failure reason see
        IoErr().

SEE ALSO
    CreateXRefFileA(), CloseXRefFile()
```

1.18 xref.library/XR_ExpungeXRef()

```
NAME
    XR_ExpungeXRef - expunge a given or all xreffile from memory
    XR_ExpungeXRefTags - varargs stub for XR_ExpungeXRef()

SYNOPSIS
    num = XR_ExpungeXRef(tagList);
                         A0

    num = XR_ExpungeXRefTags(tag1,...);
```

```
    ULONG XR_ExpungeXRef(struct TagItem *);

    ULONG XR_ExpungeXRefTags(ULONG ,...);

FUNCTION
    expunges a given or all xreffiles from memory. This must specified
    via TAGS. A NULL pointer for the tagList means, that all xreffiles
    should be expunged !

INPUTS
    tagList (struct TagItem *) - tags, which specify which xreffile(s)
        should be expunged

TAGS
    XREFA_Category (STRPTR) - expunge all xreffiles, which belongs to the
        specified category (pattern) !

    XREFA_File (STRPTR) - expunge only the given file, if you have setup
        the global XRefPath, this can be only the filename.

    XREFA_Lock (ULONG) - if the data value if XREF_UNLOCK expunge also
        xreffiles, which are locked

    XREFA_XRefHook (struct Hook *) - callback function to confirm the
        expunge of the given xreffile. The function is called in the
        Amiga 2.0 standard way with the object pointer to the
        XRefFileNode structure (struct XRefFileNode *) and the message
        XRM_EXPUNGE : (struct xrmExpunge *) !

RESULTS
    number of expunged files

SEE ALSO
    XR_LoadXRef(), GetXRefFileAttrsA(), SetXRefFileAttrsA()
```

1.19 xref.library/XR_LoadXRef()

```
NAME
    XR_LoadXRef - load a xreffile into the memory
    XR_LoadXRefTags - varargs stub for XR_LoadXRef()

SYNOPSIS
    xrefnode = XR_LoadXRef(file,tagList);
     D0                    A0     A1

    xrefnode = XR_LoadXRefTags(file,tag1,...);

    struct XRefFileNode *XR_LoadXRef(STRPTR ,struct TagItem *);

    struct XRefFileNode *XR_LoadXRefTags(STRPTR,ULONG,...);

FUNCTION
    loads the specified file in the memory and adds it to the global
    xref list in the library.
```

```
INPUTS
    file (STRPTR) - file to load into the memory
    tagList (struct TagItem *) - pointer to a TagItem array with more
        arguments see TAGS.

TAGS
    XREFA_Category (STRPTR) - category for the xreffile, if no category
        is specified, it searches for a category definition in the
        xreffile. If there is also no specified, then this file is a
        global xreffile and is parsed every time ParseXRef() is called.

    XREFA_Priority (BYTE) - priority for this xreffile. This priority is
        used to insert the xreffile in the global xref list using the
        exec.library/Enqueue() function.Thus if a xreffile has a high
        priority it is parsed before any xreffile with lower priority.

    XREFA_Lock (ULONG) - You can pass XREF_LOCK or XREF_UNLOCK here.
        If you pass XREF_LOCK, this xreffile is locked and can be removed
        only by a XR_ExpungeXRef() call with the XREF_UNLOCK value. Or
        by changing the lock mode via a SetXRefFileAttrsA() call.
        If the xreffile is locked it can't removed by a memory flush !!!

    XREFA_CustomHook (struct Hook *) - callback function to handle a
        specified xreffile. This is used if someone has a own xreffile
        format !
        Not implemented yet !

    XREFA_File (STRPTR) - overwrites the filename from the first argument

    XREFA_Name (STRPTR) - name of the xreffile, this name is used instead
        of the filename

    XREFA_Index (BOOL) - if TRUE, this function tries to allocate an
        index buffer for all entries. So the if this is going allright,
        each search function uses the binary search algorithm. If it is
        FALSE it deallocates this index buffer and the search is then
        sequential.

    XREFA_XRefHook (struct Hook *) - callback function to notify, that
        the given xreffile was loaded.With object (struct XRefFileNode *)
        and message XRM_LOAD: (struct xrmLoad *) !

RESULTS
    NULL for a failure (see IoErr() for the reason),otherwise the pointer
    to the XRefFileNode structure !

    Special error codes :

    ERROR_OBJECT_WRONG_TYPE - indicates , that the given file has not the
        right fileformat (wrong fileformat version) !

    ERROR_NO_MORE_ENTRIES - indicates , that the given file has no
        entries. Thus this file is obsolete !

SEE ALSO
    XR_ExpungeXRef(), ParseXRef(), SetXRefFileAttrsA(),
```

```
        GetXRefFileAttrsA(), libraries/xref.h
```