

BBSRead.adoc

COLLABORATORS

	<i>TITLE :</i> BBSRead.adoc		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		October 9, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	BBSRead.adoc	1
1.1	bbsread.library	1
1.2	bbsread.library/--background--	3
1.3	bbsread.library/AppendPassiveConfList()	4
1.4	bbsread.library/Archive()	5
1.5	bbsread.library/BBSEventArchiver()	6
1.6	bbsread.library/BBSUserData()	6
1.7	bbsread.library/BufBRClose()	7
1.8	bbsread.library/BufBROpen()	8
1.9	bbsread.library/BufBRRead()	8
1.10	bbsread.library/BufBRSeek()	9
1.11	bbsread.library/BufBRWrite()	10
1.12	bbsread.library/CharsetConvert()	10
1.13	bbsread.library/ConfCharset()	11
1.14	bbsread.library/ConfigBBS()	12
1.15	bbsread.library/ConfigConf()	16
1.16	bbsread.library/ConfigFArea()	19
1.17	bbsread.library/ConfigGlobal()	21
1.18	bbsread.library/ConfigType()	24
1.19	bbsread.library/ConfLineLength()	27
1.20	bbsread.library/EndOfAdding()	27
1.21	bbsread.library/ExternalBBSConfig()	28
1.22	bbsread.library/FindDupBRMsg()	29
1.23	bbsread.library/FindOrginalNr()	30
1.24	bbsread.library/FreeBRObject()	31
1.25	bbsread.library/GetBBSList()	31
1.26	bbsread.library/GetConfigValue()	32
1.27	bbsread.library/GetConfList()	33
1.28	bbsread.library/GetFAreaList()	34
1.29	bbsread.library/GetGlobalConfig()	35

1.30	bbsread.library/GetMarkedMsg()	36
1.31	bbsread.library/GetSignature()	36
1.32	bbsread.library/GetTagFile()	37
1.33	bbsread.library/GetTypeList()	38
1.34	bbsread.library/MakeEventPackage()	39
1.35	bbsread.library/MarkMessage()	40
1.36	bbsread.library/PackDataFile()	41
1.37	bbsread.library/ParseGrab()	43
1.38	bbsread.library/PGPBREvents()	44
1.39	bbsread.library/ReadBREvent()	45
1.40	bbsread.library/ReadBRFile()	46
1.41	bbsread.library/ReadBRKill()	47
1.42	bbsread.library/ReadBRMessage()	49
1.43	bbsread.library/ReadBRUser()	50
1.44	bbsread.library/ReadPassiveConfList()	52
1.45	bbsread.library/ScanForGrabs()	52
1.46	bbsread.library/SearchBRFile()	53
1.47	bbsread.library/SearchBRMessage()	54
1.48	bbsread.library/SearchBRUser()	56
1.49	bbsread.library/SortMessageArray()	58
1.50	bbsread.library/StartOfAdding()	58
1.51	bbsread.library/TypeFromBBS()	59
1.52	bbsread.library/UnArchive()	60
1.53	bbsread.library/UniqueMsgFile()	61
1.54	bbsread.library/UpdateBREvent()	62
1.55	bbsread.library/UpdateBRMessage()	64
1.56	bbsread.library/UpdateDataStruct()	68
1.57	bbsread.library/WriteBREvent()	69
1.58	bbsread.library/WriteBRFile()	72
1.59	bbsread.library/WriteBRIEFMsg()	73
1.60	bbsread.library/WriteBRKill()	73
1.61	bbsread.library/WriteBRMessage()	75
1.62	bbsread.library/WriteBRUser()	80
1.63	bbsread.library/WritePassiveConfList()	81

Chapter 1

BBSRead.adoc

1.1 bbsread.library

```
--background--  
AppendPassiveConfList ()  
Archive ()  
BBSEventArchiver ()  
BBSUserData ()  
BufBRClose ()  
BufBROpen ()  
BufBRRead ()  
BufBRSeek ()  
BufBRWrite ()  
CharsetConvert ()  
ConfCharset ()  
ConfigBBS ()  
ConfigConf ()  
ConfigFArea ()  
ConfigGlobal ()  
ConfigType ()  
ConfLineLength ()  
EndOfAdding ()
```

ExternalBBSConfig()
FindDupBRMsg()
FindOriginalNr()
FreeBRObject()
GetBBSList()
GetConfigValue()
GetConfList()
GetFAreaList()
GetGlobalConfig()
GetMarkedMsg()
GetSignature()
GetTagFile()
GetTypeList()
MakeEventPackage()
MarkMessage()
PackDataFile()
ParseGrab()
PGPBREvents()
ReadBREvent()
ReadBRFile()
ReadBRKill()
ReadBRMessage()
ReadBRUser()
ReadPassiveConfList()
ScanForGrabs()
SearchBRFile()
SearchBRMessage()
SearchBRUser()

```
SortMessageArray()  
StartOfAdding()  
TypeFromBBS()  
UnArchive()  
UniqueMsgFile()  
UpdateBREvent()  
UpdateBRMessage()  
UpdateDataStruct()  
WriteBREvent()  
WriteBRFile()  
WriteBRIEFMsg()  
WriteBRKill()  
WriteBRMessage()  
WriteBRUser()  
WritePassiveConfList()
```

1.2 bbsread.library/--background--

NOTES

Since this library uses functions in the dos.library, only processes should open and use this library.

All strings passed to functions in this library should use the standard Amiga character set (ISO). Use

```
CharsetConvert()  
to convert
```

the charset of strings which is not in ISO.

The library uses two environment variables:

- o THOR/BBSDataPath - Path to where to store the database and configuration files.
- o THOR/THORPath - Path to where the Thor system is installed. This path *must* end with a ':' or a '/'.

Progress Hooks:

Several functions support call back progress hooks. The hook is called with a BRProgress structure as the object parameter. The BRProgress structure is described in <libraries/bbsread.h>. The tags for progress hooks are global:

BR_ProgressHook - Callback hook for progress report is pointed by

(struct Hook *) ti_Data. Returning non-zero from the callback hook will stop the operation and in some cases make the function return failure.

BR_ProgressUpdates - The maximum number of calls to the callback hook for one pass is in (ULONG) ti_Data. The default is 100.

BR_ProgressReturn - Pointer to a ULONG to store the return value from the progress hook is in (ULONG *) ti_Data.

Common configuration tags:

(Not supported by all configuration functions yet.)

BRCFG_Use - Don't store the changes done to the configuration if (BOOL) ti_Data is TRUE.

BRCFG_LastSaved - Will retrieve the configuration last saved before applying any changes.

1.3 bbsread.library/AppendPassiveConfList()

NAME

AppendPassiveConfList - Append to the passive conference list.

SYNOPSIS

```
error = AppendPassiveConfList( bbs, passConfList )
D0                A0        A1
```

```
BOOL AppendPassiveConfList( struct BBSListItem *, struct List * );
```

FUNCTION

Appends a list of struct PasssConfListItem to a passive conference list. No checking for duplicates will be done. If you don't have the internal number for the conference, pl_BBSCnfNr must be initialized to -1.

INPUTS

bbs - Pointer to the bbs which the list should be added to.
passConfList - Pointer to a list of struct PasssConfListItem.

RESULT

error - Boolean.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.4 bbsread.library/Archive()

NAME

Archive -- Add files to an archive.

SYNOPSIS

```
error = Archive( arcitem, archive, tagitems )
D0                A0          A1          A2
```

```
LONG Archive( struct ArcConfigItem *, STRPTR, struct TagItem * );
```

```
error = ArchiveTags( arcitem, archive, Tag1, ...)
```

```
LONG ArchiveTags(struct ArcConfigItem *, STRPTR, ULONG, ...);
```

FUNCTION

This function add files to an archive. The type of archiver to use is determined with the arcitem.

The ability to return failure when the archiver fails depends on how the archivers behave.

The tags not understood are forwarded to dos.library/SystemTagList(). Look at dos.library/SystemTagList() for futher information on tags.

INPUTS

arcitem - Pointer to struct ArcConfigItem to tell which archiver to use.

archive - Pointer to path and name of archive to add to.

tagitems - Pointer to TagItem array. See <dos/dostags.h>. Both dos.library/SystemTagList() tags and dos.library/CreateNewProc() tags may be passed.

Here are the TagItem.ti_Tag values that are defined for Archive().

AR_AddFile - File to add to archive is in (STRPTR) ti_Data. At least one of this tag must be passed.

AR_SourceDir - Path to source directory is in (STRPTR) it_Data. Default is to use current directory as source directory.

RESULT

error - 0 for success, result from archiver, or -1. Note that on error, the caller is responsible for any filehandles or other things passed in via tags.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.5 bbsread.library/BBSEventArchiver()

NAME

BBSEventArchiver - Returns the EventArchiver for a bbs.

SYNOPSIS

```
arcitem = BBSEventArchiver( bbs )
D0                      A0
```

```
struct ArcConfigItem * BBSEventArchiver( struct BBSListItem * );
```

FUNCTION

Returns the ArcConfigItem structure correspondig to the EventArchiver for this bbs. If the bbs has no defined event archiver, the event archiver for the bbs type will be returned.

Use this function instead of searching the ArcItem list by name for the correspondig ArcConfigItem.

The ArcConfigItem structure returned is not a part of a list, so don't use the ac_Node. The structure must be deallocated with

```
FreeBRObject()
```

.

INPUTS

bbs - Pointer to the bbs to get the archiver to.

RESULT

arcitem - Pointer to a ArcConfigItem for the bbs. Returns a NULL pointer on failure (IoErr() will be set) or if the bbs and type has no defined EventArchiver.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.6 bbsread.library/BBSUserData()

NAME

BBSUserData -- Return the prefered userdata for a BBS.

SYNOPSIS

```
userdata = BBSUserData( globals, bbs )
D0                      A0      A1
```

```
struct UserData * BBSUserData( struct GlobalConfig *,
    struct BBSListItem * );
```

FUNCTION

Get the preferred userdata for a bbs. The data is returned in a structure. The structure **must** be freed with `FreeBRObject()`.

INPUTS

`globals` - Pointer to your copy of the global configuration.
`bbs` - Pointer to `BBSListItem` for BBS to get `UserData` for. Can be a `NULL`-pointer.

RESULT

`userdata` - Filled out `UserData` structure. `NULL` on failure.

EXAMPLE

NOTES

The pointers in the `UserData` structure is a copy of the pointers in either `GlobalConfig` or `BBSData`. Therefore, the pointers are only valid as long as you don't free or update any of these structures.

BUGS

SEE ALSO

1.7 bbsread.library/BufBRClose()

NAME

`BufBRClose` -- Close a file used with `BBSRead` buffering.

SYNOPSIS

```
success = BufBRClose( fileid )  
D0                      A0
```

```
BOOL BufBRClose( APTR );
```

FUNCTION

Close a file opened by `BufBROpen()`.

INPUTS

`fileid` - Fileid for file to close.

RESULT

`success` - Boolean.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.8 bbsread.library/BufBROpen()

NAME

BufBROpen -- Open a BBSRead buffered file for input or output.

SYNOPSIS

```
fileid = BufBROpen( name, accessMode )
D0                A0                D0
```

```
APTR BufBROpen( STRPTR, LONG );
```

FUNCTION

The named file is prepared to be used for input or output through the BBSRead buffering system. All files are opened in shared mode. Uses the same mode constants as dos.library/Open().

BBSRead buffered files should **not** be used when you want to read and write large blocks.

BBSRead buffered files should **not** be written to by processes not using

```
BufBRWrite()
```

.

INPUTS

name - Name of file to open.
accessMode - Mode to open file in.

RESULT

fileid - The fileid for the opened file. Returns a NULL pointer if the open failed, and a secondary error code will be available in IoErr().

EXAMPLE

NOTES

The fileid returned is **not** compatible with the file handler returned by dos.library/Open().

BUGS

SEE ALSO

1.9 bbsread.library/BufBRRead()

NAME

BufBRRead -- Read from a BBSRead buffered file.

SYNOPSIS

```
actualLength = BufBRRead( fileid, buffer, length )
D0                A0      A1      D0
```

```
LONG BufBRRead( APTR, APTR, LONG );
```

FUNCTION

Read from a file buffered with the bbsread buffer system.

INPUTS

fileid - APTR to BBSRead file handler.
buffer - Pointer to buffer.
length - Length to read.

RESULT

actualLength - Actual length read. A value of 0 means EOF, and errors are indicated with -1. See IoErr() for error specification.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.10 bbsread.library/BufBRSeek()

NAME

BufBRSeek -- Set the current position for a BBSRead buffered file.

SYNOPSIS

```
oldPosition = BufBRSeek( fileid, position, mode )
D0                A0      D0      D1
```

```
LONG BufBRSeek( APTR, LONG, LONG );
```

FUNCTION

Sets the read/write cursor for a BBSRead buffered file. Works like dos.library/Seek().

INPUTS

fileid - APTR to BBSRead file handler.
position - Position to seek to.
mode - Seek mode. (Same as dos.library/Seek().)

RESULT

oldPosition - Old position in file. Returns -1 on error.

EXAMPLE

NOTES

BUGS

SEE ALSO
dos.library/Seek()

1.11 bbsread.library/BufBRWrite()

NAME
BufBRWrite -- Write to a BBSRead buffered file.

SYNOPSIS
actualLength = BufBRWrite(fileid, buffer, length)
D0 A0 A1 D0

LONG BufBRWrite(APTR, APTR, LONG);

FUNCTION
Write to a file buffered with the bbsread buffer system.

INPUTS
fileid - APTR to BBSRead file handler.
buffer - Pointer to buffer.
length - Length to write.

RESULT
actualLength - Actual length written. Errors are indicated with -1.
See IoErr() for error specification.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.12 bbsread.library/CharsetConvert()

NAME
CharsetConvert -- Converts strings from and to the ISO charset.

SYNOPSIS
success = CharsetConvert(fromchar, tochar, frombuf, tobuf, len)
D0 D1 D2 A0 A1 D3

BOOL CharsetConvert(UBYTE, UBYTE, STRPTR, STRPTR, ULONG);

FUNCTION
Converts strings of charset fromchar in frombuf to strings of charset tochar in tobuf. Available charsets are BRCS_ISO, BRCS_IBN, BRCS_SF7, BRCS_NO7, BRCS_DE7.

INPUTS
fromchar - The charset used in frombuf.

tochars - The charset to convert to.
frombuf - The buffer with the string(s) to convert.
tobuf - The buffer to put the result of the conversion. The size of the buffer must be equal to the size of the frombuf. If this pointer is NULL, the result will be put in frombuf.
len - Number of bytes to convert. If this parameter is 0L, the 0-terminated string in frombuf will be converted.

RESULT
success - Boolean.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.13 bbsread.library/ConfCharset()

NAME
ConfCharset -- Returns which charset the bbs has in this conf.

SYNOPSIS
charset = ConfCharset(bbs, conf)
D0 A0 A1

UBYTE ConfCharset(struct BBSListItem *, struct ConfListItem *);

FUNCTION
Use this function to obtain which charset the messages in a conf from the BBS is expected to be in.

The character set expected for a BBS can be obtained by using a NULL pointer as the conf parameter.

INPUTS
bbs - BBS to obtain charset for.
conf - Conf to obtain charset for.

RESULT
charset - Charset expected for this conf.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.14 bbsread.library/ConfigBBS()

NAME

ConfigBBS -- Set up the configuratin for a BBS.

SYNOPSIS

```
newbbs = ConfigBBS( bbs, tagitems )
```

```
D0                A0        A1
```

```
struct BBSListItem * ConfigBBS( struct BBSListItem *,
    struct TagItem * );
```

```
newbbs = ConfigBBSTags( bbs, Tag1, ... )
```

```
struct BBSListItem * ConfigBBSTags( struct BBSListItem *, ULONG,
    ... );
```

FUNCTION

Changes the setup for a BBS, or adds a new BBS to the database.

INPUTS

bbs - Pointer to the BBSListItem for the bbs to change configuration for. If this pointer is a NULL pointer, a new bbs will be created. The contents of bbs->bl_Data will be updated and string pointers may change.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for ConfigBBS().

BC_DeleteBBS - Markes the BBS pointed by the bbs parameter as deleted if (BOOL) ti_Data is TRUE. Your BBSListItem passed as the bbs parameter will be removed from your bbslist. The pointer returned is the old bbs pointer, but it will not point to a BBSListItem any more.

BC_BBSList - Your BBSList-header is pointed by (struct List *) ti_Data. This is the pointer returned from GetBBSList()

. When

adding a new BBS to the database this tag *must* be supplied.

It *must* also be supplied when using the BC_Top, BC_Bottom, BC_Up, BC_Down, BC_SortBBSList and BC_NewBBSOrder tags.

BC_BBSName - The new name of the BBS is pointed by (STRPTR) ti_Data.

BC_GrabName - The new name of the grabfile (without extension) is pointed by (STRPTR) ti_Data. Wildcards are supported. If wildcards are used, must the name match the full file name.

BC_BBSType - The new type of the BBS is pointed by (STRPTR) ti_Data.

If no BBSType with this name is defined, the funktion will fail.

BC_UserName - The new name the user is registered as is pointed by (STRPTR) ti_Data.

BC_ScriptFlags - Flags for the script is in (LONGBITS) ti_Data. Check

the BBS type for available flags. Non-available flags passed will be ignored.

BC_Signature - Signature to use on this BBS is pointed by (STRPTR) ti_Data. This signature should be used in conferences with no signature defined. A NULL-pointer equals no BBS signature, and the global signature should be used on this BBS. If the signature is in a file, this tag should contain the complete path to the signature file, and the BDF_FILE_SIGNATURE flag must be set.

BC_Top - Move the BBS to the top of the list if (BOOL) ti_Data is TRUE. Both your local and the global list will be updated. Be sure to also pass the BC_BBSList tag.

BC_Bottom - Move the BBS to the bottom of the list if (BOOL) ti_Data is TRUE. Both your local and the global list will be updated. Be sure to also pass the BC_BBSList tag.

BC_Up - Move the BBS one position upwards on the list if (BOOL) ti_Data is TRUE. Both your local and the global list will be updated. Be sure to also pass the BC_BBSList tag.

BC_Down - Move the BBS one position downwards on the list if (BOOL) ti_Data is TRUE. Both your local and the global list will be updated. Be sure to also pass the BC_BBSList tag.

BC_SortBBSList - Sort the BBS-list alphabetically if (BOOL) ti_Data is TRUE. Both your local and the global list will be sorted. Be sure to also pass the BC_BBSList tag.

BC_KeepMessages - Messages to keep in each conference when it's packed is in (ULONG) ti_Data.
BBSData->bd_Flags affecting the use of this value:
- BDF_IGNORE_KEEPMMSG: Messages won't be counted when packing conferences.
- BDF_GLOBAL_KEEPMMSG: The Global KeepMsg value will be used on this BBS. The BC_KeepMessages value will be stored, but it will be ignored.

BC_KeepTime - How old messages to keep in each conference when it's packed is in (ULONG) ti_Data. The time is in seconds.
BBSData->bd_Flags affecting the use of this value:
- BDF_IGNORE_KEEPTIME: Time won't be checked when packing conferences.
- BDF_GLOBAL_KEEPTIME: The Global KeepTime value will be used on this BBS. The BC_KeepTime value will be stored, but it will be ignored.

BC_SetBBSFlags - BBS flags to set is in (LONGBITS) ti_Data. See <libraries/bbsread.h> for documentation on each flag.

BC_ClearBBSFlags - BBS flags to clear is in (LONGBITS) ti_Data.

BC_EmailAddr - The address the user is registered with is pointed by (STRPTR) ti_Data. This is used to check if a message is to the user. This address will be ignored if the BBSType for this has the TDF_NO_ADDR_CHECK flag set.

BC_XpkMethod - BBS Xpk method to use is pointed by (STRPTR) ti_Data.
If BC_XpkMethod is set to NULL, the global Xpk method will be used for this bbs. BBSData->bd_Flags affecting the use of this value:
- BDF_NO_XPK_METHOD: Don't use Xpk on this bbs.

BC_CharSet - The expected charset for the grabs from this bbs is in (UBYTE) ti_Data. Setting BC_CharSet to BRCS_ANY will use the default charset for the BBSType to this bbs. Default charset when adding a new bbs is BRCS_ANY.

BC_LineLength - Max linelength of messages for this bbs is (UWORD) ti_Data. Overrides the setting in the BBSType for this bbs if BC_LineLength is non 0.

BC_UserStreet - User street address is pointed by (STRPTR) ti_Data.
Overrides settings in global configuration if non-NULL.

BC_UserAddress - User address is is pointed by (STRPTR) ti_Data.
Overrides settings in global configuration if non-NULL.

BC_UserCountry - User country is is pointed by (STRPTR) ti_Data.
Overrides settings in global configuration if non-NULL.

BC_UserPhone - User phone number is pointed by (STRPTR) ti_Data.
Overrides settings in global configuration if non-NULL.

BC_Alias - Alias used on this BBS is pointed by (STRPTR) ti_Data.
If the alias is set and a conference on this bbs is defined as CDF_ALIAS, the alias will be used to determine if messages are to/from the user in the particular conference.

BC_DnloadPath - Download path for this BBS is pointed by (STRPTR) ti_Data. Overrides settings in global configuration if non-NULL.

BC_TagFile - Path and name of tagfile to use for this bbs is pointed by (STRPTR) ti_Data. Overrides settings in global configuration if non-NULL.

BC_EventArchiver - Archiver to use when packing the events from this bbs is pointed by (STRPTR) ti_Data. Overrides settings in BBSType if non-NULL.

BC_ReplyPacket - Filename of relyphacket is pointed by (STRPTR) ti_Data. The filename is expected to be relative to the defined upload directory for this bbs.

BC_UploadPath - Upload path for this BBS is pointed by (STRPTR) ti_Data. Overrides settings in global configuration if non-NULL.

BC_NewBBSOrder - Rearrange the order of the bbses according to the list given in the BC_BBSList tag if (BOOL) ti_Data is TRUE.

BC_QuoteType - Preferred quote type for this bbs is in (UBYTE) ti_Data. Overrides settings in bbstype if not QT_USE_SUPER.
See <libraries/BBSRead.h> for definitions of quote types.

BC_QuoteChars - String to use as quote chars in custom quote type is in (STRPTR) ti_Data. Max length of the string is 3. Overrides globally defined quote chars if NULL or 0 length.

BC_ReplyString - Reply string to use when a message is replied _and_ moved is in (STRPTR) ti_Data. Overrides globally defined reply string if non NULL.

BC_BBSEnterScript - Name of Arexx script to be run each time this bbs is entered is pointed by (STRPTR) ti_Data. Overrides globally defined enter script if non NULL.

BC_BBSLeaveScript - Name of Arexx script to be run each time this bbs is left is pointed by (STRPTR) ti_Data. Overrides globally defined leave script if non NULL.

BC_ConfEnterScript - Name of Arexx script to be run each time a conference on this bbs is entered is pointed by (STRPTR) ti_Data. Overrides globally defined enter script if non NULL.

BC_ConfLeaveScript - Name of Arexx script to be run each time a conference on this bbs is left is pointed by (STRPTR) ti_Data. Overrides globally defined leave script if non NULL.

RESULT

newbbs - When changing an existing bbs, this is the same as the bbs parameter. When adding a new bbs, this points to a BBSListItem for the new BBS. On failure, a NULL pointer is returned.

EXAMPLE

NOTES

When adding a BBS, the BBSListItem for this BBS will automatically be inserted in the list supplied with the BC_BBSList tag. Therefore, be sure the list is not attached to a ListView or similar when calling this function. The same applies when deleting BBS'es and when rearranging the bbslist.

You *must* supply BC_BBSName and BC_BBSType when adding a new BBS to the database.

As of V4 is BC_GrabName no longer needed when adding a BBS.

When adding new BBS'es will BC_Top, BC_Bottom, BC_Up, BC_Down, BC_SortBBSList and BC_NewBBSOrder be ignored.

These flags are automatically set when adding a new bbs:
BDF_GLOBAL_KEEPMMSG | BDF_GLOBAL_KEEPTIME | BDF_IGNORE_KEEPMMSG
| BDF_IGNORE_KEEPTIME.

BUGS

SEE ALSO

GetBBSList ()

1.15 bbsread.library/ConfigConf()

NAME

ConfigConf -- Set up the configuration for a conference.

SYNOPSIS

```
newconf = ConfigConf( conf, tagitems )
```

```
DO                                A0      A1
```

```
struct ConfListItem * ConfigConf( struct ConfListItem *,
    struct TagItem * );
```

```
newconf = ConfigConfTags( conf, Tag1, ... )
```

```
struct ConfListItem * ConfigConfTags( struct ConfListItem *, ULONG,
    ... );
```

FUNCTION

Changes the setup for a conference, or adds a new conference to a bbs. Only conferences where messages are expected should be added. Use

```
WritePassiveConfList()
    for complete conference lists.
```

INPUTS

conf - Pointer to the ConfListItem for the conference to change configuration for. If this pointer is a NULL pointer, a new conference will be added to the bbs. The contents of conf->cl_Data will be updated and string pointers may change.
tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for ConfigConf().

CC_DeleteConf - Markes the Conf pointed by the conf parameter as deleted if (BOOL) ti_Data is TRUE. Your ConfListItem passed as the conf parameter will be removed from your conflist. The pointer returned is the old conf pointer, but it will not point to a ConfListItem any more.

CC_ConfList - Your ConfList-header is pointed by (struct List *) ti_Data. This is the pointer returned from GetConfList().
. When adding a new conference to a bbs, this tag **must** be supplied. It **must** also be supplied when using the CC_Top, CC_Bottom, CC_Up, CC_Down, CC_SortConfList and CC_NewConfOrder tags.

CC_AddToBBS - The BBSListItem to add a conference to is pointed by (struct BBSListItem *) ti_Data. **Must** be used when adding a conference, and is ignored when changing the setup on a existing conference.

CC_ConfName - The new name of the conference is pointed by (STRPTR) ti_Data. This tag must be supplied when adding a conference.

CC_SetConfFlags - Conference flags to set is in (LONGBITS) ti_Data.
CC_ClearConfFlags - Conference flags to clear is in
(LONGBITS) ti_Data.

CC_Signature - Signature to use in this conference is pointed by
(STRPTR) ti_Data. A NULL-pointer equals no conference signature.
The BBS signature should be used instead. If the signature is in
a file, this tag should contain the complete path to the
signature file, and the CDF_FILE_SIGNATURE flag must be set.

CC_KeepMessages - Messages to keep in this conference when it's
packed is in (ULONG) ti_Data.
ConfData->cd_Flags affecting the use of this value:
- CDF_IGNORE_KEEPMMSG: Messages won't be counted when packing
this conference.
- CDF_BBS_KEEPMMSG: The BBS KeepMsg value will be used in when
packing this conference. The CC_KeepMessages value will be
stored, but it will be ignored.

CC_KeepTime - How old messages to keep in this conference when it's
packed is in (ULONG) ti_Data. The time is in seconds.
ConfData->cd_Flags affecting the use of this value:
- CDF_IGNORE_KEEPTIME: Time won't be checked when packing
this conference.
- CDF_BBS_KEEPTIME: The BBS KeepTime value will be used when
packing this conference. The CC_KeepTime value will be stored,
but it will be ignored.

CC_Top - Move the conference to the top of the list if (BOOL) ti_Data
is TRUE. Both your local and the global list will be updated. Be
sure to also pass the CC_ConfList tag.

CC_Bottom - Move the conference to the bottom of the list if (BOOL)
ti_Data is TRUE. Both your local and the global list will be
updated. Be sure to also pass the CC_ConfList tag.

CC_Up - Move the conference one position upwards on the list if
(BOOL) ti_Data is TRUE. Both your local and the global list will
be updated. Be sure to also pass the CC_ConfList tag.

CC_Down - Move the conference one position downwards on the list if
(BOOL) ti_Data is TRUE. Both your local and the global list will
be updated. Be sure to also pass the CC_ConfList tag.

CC_Alias - Alias to use in this conference is pointed by (STRPTR)
ti_Data. If the alias is set and the conference is defined as
CDF_ALIAS, it will be used to determine if messages are to/from
the user in this particular conference. Overrides settings in
BBSData if non-NULL.

CC_BBSSconfNr - The internal number this conference has on the BBS is
in (LONG) ti_Data. This value is meant to ease the parsing and
packing when numbers must be used for the conferences.

CC_SortConfList - Sort the Conf-list alphabetically if (BOOL) ti_Data
is TRUE. Both your local and the global list will be sorted. Be

sure to also pass the CC_ConfList tag. You must also pass a valid conf parameter to use this tag.

CC_XpkMethod - Conf Xpk method to use is pointed by (STRPTR) ti_Data. If CC_XpkMethod is set to NULL will the bbs Xpk method be used for this conf. ConfData->cd_Flags affecting the use of this value:
- CDF_NO_XPK_METHOD: Don't use Xpk on this conf.

CC_CharSet - The expected charset for the grabs from this conf is in (UBYTE) ti_Data. Setting CC_CharSet to BRCS_ANY will use the default charset for this bbs. Default charset when adding a newconf is BRCS_ANY.

CC_LineLength - Max linelength of messages for this conf is (UWORD) ti_Data. Overrides the setting for this bbs if CC_LineLength is non 0.

CC_TagFile - Path and name of tagfile to use for this conf is pointed by (STRPTR) ti_Data. Overrides settings on bbs if non-NULL.

CC_EmailAddr - The email address the user has in this conference is pointed by (STRPTR) ti_Data. This is used to check if a message is to the user. Overrides setting in BBSData. This address will be ignored if the BBSType for this has the TDF_NO_ADDR_CHECK flag set. This tag is for use on bbses where the user is member of more than one net.

CC_NewConfOrder - Rearrange the order of the conferences according to the list given in the CC_ConfList tag if (BOOL) ti_Data is TRUE.

CC_QuoteType - Preferred quote type for this conference is in (UBYTE) ti_Data. Overrides settings on the bbs if not QT_USE_SUPER. See <libraries/BBSRead.h> for definitions of quote types.

CC_QuoteChars - String to use as quote chars in custom quote type is in (STRPTR) ti_Data. Max length of the string is 3. Overrides quote chars for bbs if NULL or 0 length.

CC_ReplyString - Reply string to use when a message is replied _and_ moved is in (STRPTR) ti_Data. Overrides reply string defined for the bbs if non NULL.

CC_ConfEnterScript - Name of Arexx script to be run each time this conference is entered is pointed by (STRPTR) ti_Data. Overrides enter script defined for the bbs if non NULL.

CC_ConfLeaveScript - Name of Arexx script to be run each time this conference is left is pointed by (STRPTR) ti_Data. Overrides leave script defined for the bbs if non NULL.

CC_ConfNetType - Which type of network this conference is connected to is in (UBYTE) ti_Data. This tag is to be used on BBS'es of BBS types which supports several network types. The default value is CDNT_NONET. See <libraries/BBSRead.h> for definitions. New definitions are added on request. No need to use this tag if the bbs type only supports one network.

CC_HighMark - The high mark for this conference on the remote server/.. is in (ULONG) ti_Data. This can be used by the parser to keep uptodate what the last article collected was.

RESULT

newconf - When changing an existing conference, this is the same as conf parameter. When adding a new conference this points to a ConfListItem for this conference. On failure a NULL pointer is returned.

EXAMPLE

NOTES

When adding a conference, the ConfListItem for this conference will automatically be inserted in the list supplied with the CC_ConfList tag. Therefore, be sure the list is not attached to a ListView or similar when calling this function. The same applies when deleting conferences and when rearranging the conference list.

When adding new conferences will CC_Top, CC_Bottom, CC_Up, CC_Down, CC_SortConfList and CC_NewConfOrder be ignored.

These flags are automatically set when adding a new conference:

```
CDF_BBS_KEEPMMSG | CDF_BBS_KEEPTIME | CDF_IGNORE_KEEPMMSG
| CDF_IGNORE_KEEPTIME.
```

BUGS

SEE ALSO

GetConfList()

1.16 bbsread.library/ConfigFArea()

NAME

ConfigFArea -- Set up the configuration for a file area.

SYNOPSIS

```
newfarea = ConfigFArea( farea, tagitems )
```

```
D0                A0        A1
```

```
struct FAreaListItem * ConfigFArea( struct FAreaListItem *,
    struct TagItem * );
```

```
newfarea = ConfigFAreaTags(farea, Tag1, ... )
```

```
struct FAreaListItem * ConfigFAreaTags( struct FAreaListItem *,
    ULONG, ... );
```

FUNCTION

Changes the setup for a file area, or adds a new file area to the database.

INPUTS

farea - Pointer to FAreaListItem for the file area to change configuration for. If this pointer is a NULL pointer, a new file area will be created.
tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for ConfigFArea().

CFA_DeleteFArea - Markes the file area pointed by the farea parameter as deleted if (BOOL) ti_Data is TRUE. Your FAreaListItem passed as the farea parameter will be removed from your farealist. The pointer returned is the old fareapointer, but it will not point to a FAreaListItem any more.

CFA_FAreaList - Your FAreaList-header is pointed by (struct List *) ti_Data. This is the pointer returned from GetFAreaList().
. When adding a new file area to the database, this tag *must* be supplied. It *must* also be supplied when using the CFA_Top, FCA_Bottom, CFA_Up, CFA_Down and CFA_SortFAreaList tags.

CFA_AddToBBS - The BBSListItem to add a file area to is pointed by (struct BBSListItem *) ti_Data. *Must* be used when adding a file area, and is ignored when changing the setup on a existing file area.

CFA_Name - The new name of the file area is pointed by (STRPTR) ti_Data.

CFA_Top - Move the file area to the top of the list if (BOOL) ti_Data is TRUE. Both your local and the global list will be updated. Be sure to also pass the CFA_FAreaList tag.

CFA_Bottom - Move the file area to the bottom of the list if (BOOL) ti_Data is TRUE. Both your local and the global list will be updated. Be sure to also pass the CFA_FAreaList tag.

CFA_Up - Move the file area one position upwards on the list if (BOOL) ti_Data is TRUE. Both your local and the global list will be updated. Be sure to also pass the CFA_FAreaList tag.

CFA_Down - Move the file area one position downwards on the list if (BOOL) ti_Data is TRUE. Both your local and the global list will be updated. Be sure to also pass the CFA_FAreaList tag.

CFA_SortFAreaList - Sort the file area list alphabetically if (BOOL) ti_Data is TRUE. Both your local and the global list will be sorted. Be sure to also pass the CFA_FAreaList tag.

RESULT

newfarea - When changing an existing file area, this is the same as the farea parameter. When adding a new file area, this points to a FAreaListItem for this new file area. On failure, a NULL pointer is returned.

EXAMPLE

NOTES

When adding a file area, the FAreaListItem for this file area will automatically be inserted in the list supplied with the CFA_FAreaList tag. Therefore be sure the list is not attached to a ListView or similar when calling this function. The same applies when deleting file areas and when rearranging the bbslist.

You *must* supply CFA_Name when adding a new file area.

When adding new areas will CFA_Top, FCA_Bottom, CFA_Up, CFA_Down and CFA_SortFAreaList be ignored.

BUGS

SEE ALSO

1.17 bbsread.library/ConfigGlobal()

NAME

ConfigGlobal -- Global configuration for the library.

SYNOPSIS

```
success = ConfigGlobal( globals, tagitems )
D0                A0                A1
```

```
BOOL ConfigGlobal( struct GlobalConfig *, struct TagItem * );
```

```
success = ConfigGlobalTags( globals, Tag1, ...)
```

```
BOOL ConfigGlobalTags( struct GlobalConfig *, ULONG, ... );
```

FUNCTION

This function sets up the global configuration for the library. Your copy of the global configuration will also be updated. Supports BRCFG_Use and BRCFG_LastSaved tags.

INPUTS

globals - Pointer to global data got from
GetGlobalConfig()

. The contents of globals will be updated and string pointers may change.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for ConfigGlobal().

CG_DnloadPath - Path to where downloaded files reside is pointed by (STRPTR) ti_Data.

CG_UploadPath - Path to where files to be uploaded reside is pointed by (STRPTR) ti_Data.

CG_Buffers - Number of buffers to use in buffer system in
(ULONG) ti_Data. 13 buffers is the default in the current version..

CG_BufferSize - Size of each buffer the buffer system should use in
(ULONG) ti_Data. 5096 bytes is the default in the current version.

CG_ConfigArchiver - Configure archiver. Name of archiver is in
(STRPTR) ti_Data.

CG_ArcPattern - MatchPattern to recognize archives of
CG_ConfigArchiver type is in (STRPTR) ti_Data. If the ArcPattern
begins with '\$', the rest of the string will be matched with the
contents of the archive instead of the filename.
i.e. \$????2d6c68 finds LhA-archives.

CG_UnArcCmd - Command to depack an archive of CG_ConfigArchiver type
is in (STRPTR) ti_Data. If the unpack program isn't in c:, the
full path must be given. The file to be unpacked will be appended
to the command given here..

CG_ArcCmd - Command to pack an archive of CG_ConfigArchiver type is in
(STRPTR) ti_Data. The command is expected to use the standard
archiver organization of the arguments. If the archive program
isn't in c:, the full path must be given.

CG_DeleteArchiver - Delete archiver which name is in (STRPTR) ti_Data.
The deletion will fail if one bbstype uses this archiver as an
eventarchiver.

CG_Signature - Global signature is in (STRPTR) ti_Data. This signature
should be used on BBS'es where no signatures is defined. Can be a
NULL-pointer. If the signature is in a file, this tag should
contain the complete path to the signature file, and the
GCF_FILE_SIGNATURE flag must be set.

CG_KeepMessages - Messages to keep in each conference when they are
packed is in (ULONG) ti_Data.
GlobalConfig->gc_Flags affecting the use of this value:
- GCF_IGNORE_KEEPMMSG: Messages won't be counted when packing
conferences.

CG_KeepTime - How old messages to keep in each conference when they
are packed is in (ULONG) ti_Data. The time is in seconds.
GlobalConfig->gc_Flags affecting the use of this value:
- GCF_IGNORE_KEEPTIME: Time won't be checked when packing
conferences.

CG_SetGlobalFlags - Global flags to set is in (LONGBITS) ti_Data.
CG_ClearGlobalFlags - Global flags to clear is in (LONGBITS) ti_Data.

CG_XpkMethod - Global Xpk method to use is pointed by (STRPTR)
ti_Data. GlobalConfig->gc_Flags affecting the use of this value:
- GCF_NO_XPK_METHOD: Don't use Xpk globally.

CG_UserPhone - Phone number for user is pointed by (STRPTR) ti_Data.
CG_TmpDir - Name of temporary directory to use when packing data is
pointed by (STRPTR) ti_Data.

CG_TagFile - Path and name of tagfile to use globally is pointed by (STRPTR) ti_Data.

CG_BufCopyBack - Sets file buffer system to copyback mode if (BOOL) ti_Data is TRUE. In copyback mode, changed buffers will only be written back to file when they are flushed from memory. Each call to ConfigGlobal() with the CG_BufCopyBack tag *must* be coupled with a call to ConfigGlobal() with the CG_BufEndCopyBack tag. The copyback mode should be used while adding large amount of data to the library. (Eg. parsing a grabfile and adding the messages to the database) The use of the CG_BufCopyBack/G_BufEndCopyBack tags can be nested.

CG_BufEndCopyBack - Turns off copyback mode for file buffer system if (BOOL) ti_Data is TRUE.

CG_HitRate - Buffer hitrate is returned in *((ULONG *) ti_Data). The hitrate is in percent.

CG_ReadHitRate - Buffer read hitrate is returned in *((ULONG *) ti_Data). The hitrate is in percent.

CG_WriteHitRate - Buffer write hitrate is returned in *((ULONG *) ti_Data). The hitrate is in percent.

CG_ClearHitRate - Clear internal hitrate statistics if (BOOL) ti_Data is TRUE

CG_HazeLevel1 - Keep messages marked with haze level 1 at least (ULONG) ti_Data seconds.

CG_HazeLevel2 - Keep messages marked with haze level 2 at least (ULONG) ti_Data seconds.

CG_HazeLevel3 - Keep messages marked with haze level 3 at least (ULONG) ti_Data seconds.

CG_PGPCCommand - Command for pgp (with path) is pointed by (STRPTR) ti_Data.

CG_PGPSignID - Id to use when PGP signing messages is pointed by (STRPTR) ti_Data. If NULL should '*' be used as sign id.

CG_QuoteChars - String to use as quote chars in custom quote type is in (STRPTR) ti_Data. Max length of the string is 3.

CG_ReplyString - Reply string to use when a message is replied _and_ moved is in (STRPTR) ti_Data.

CG_StartupScript - Name of Arexx script to be run each time Thor is started is pointed by (STRPTR) ti_Data.

CG_QuitScript - Name of Arexx script to be run each time Thor is quited is pointed by (STRPTR) ti_Data.

CG_BBSEnterScript - Name of Arexx script to be run each time a bbs

is entered is pointed by (STRPTR) ti_Data.

CG_BBSLeaveScript - Name of Arexx script to be run each time a bbs is left is pointed by (STRPTR) ti_Data.

CG_ConfEnterScript - Name of Arexx script to be run each time a conference is entered is pointed by (STRPTR) ti_Data.

CG_ConfLeaveScript - Name of Arexx script to be run each time a conference is left is pointed by (STRPTR) ti_Data.

CG_BufFreeAllocated - Free most memory allocated for buffers if (BOOL) ti_Data is TRUE. The memory will be reallocated when needed.

RESULT
success - Boolean.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.18 bbsread.library/ConfigType()

NAME

ConfigType -- Set up the configuratin for a BBS type.

SYNOPSIS

```
newtype = ConfigType( type, tagitems )
D0          A0      A1
```

```
struct TypeListItem * ConfigType( struct TypeListItem *,
    struct TagItem * );
```

```
newtype = ConfigTypeTags( type, Tag1, ... )
```

```
struct TypeListItem * ConfigTypeTags( struct TypeListItem *,
    ULONG, ... );
```

FUNCTION

Changes the definitions for a BBS type, or adds a new BBS type to the database.

INPUTS

type - Pointer to the TypeListItem for the BBS type to change configuration for. If this pointer is a NULL pointer, a new type will be created. The contents of type->tl_Data will be updated and string pointers may change.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for ConfigType().

CT_DeleteType - Markes the BBS type pointed by the type parameter as deleted if (BOOL) ti_Data is TRUE. The deletion of the type will fail if it is used by any BBS'es.

CT_TypeList - Your TypeList-header is pointed by (struct List *) ti_Data. This is the pointer returned from GetTypeList().
. When adding a new BBS type to the database, this tag *must* be supplied. When changing the setup on existng BBS types this tag is ignored.

CT_TypeName - The new name of the BBS type is pointed by (STRPTR) ti_Data.

CT_LineLength - Max length of lines in messages in (UWORD) ti_Data.
CT_SubjectLength - Max length of subjects in (UWORD) ti_Data.
CT_FileDescrLen - Max length of short filedescription in (UWORD) ti_Data.

CT_ConfigEvent - Set up an event this bbs type support. The event identifier is in (ULONG) ti_Data. More than one event can be set up in each call.

CT_EventNeedTags - Set up what tags the event in the last CT_ConfigEvent needs. The tags needed is in a TAG_END-terminated array pointed by (ULONG *) ti_Data.

CT_EventOptTags - Set up what tags that are optional for the event in the last CT_ConfigEvent. The optional tags is in a TAG_END-terminated array pointed bt (ULONG *) ti_Data.

CT_DeleteEvent - Delete event with (ULONG) ti_Data identifier from list of supported events.

CT_CharSet - The default charset the grab from this BBS type uses is in (UBYTE) ti_Data. Default charset when adding a new BBS type is BRCS_ISO. See <libraries/BBSRead.h> for available charsets.

CT_MsgParser - Command to parse the grabs from a BBS of this type is in (STRPTR) ti_Data. The command must take the following parameters: BBSNAME - name of bbs. GRAB - name of grab, including path. ARCHIVE - Switch, the grab is an archive. DELETE - Switch, delete the grab afterwards if the adding is successful. PUBSCREEN - Name of public screen to open any progress windows on. These parameters must be there even if it is not used by the parser. See <ParseMsg/ParseMsg.c> for more info.

CT_AvailScrFlags - Mask for available scriptflags in (LONGBITS) ti_Data.

CT_FileNameLen - Max length of filenames allowed on this BBS type is in (UWORD) ti_Data. All characters in the filename are counted.

CT_EventPacker - Command to pack events in a package to send to the BBS is in (STRPTR) ti_Data. A NULL-pointer equals no packing of events on this BBS type. The execution of this command should be done on user request by the program which makes the events. The command should take the following parameters: BBSNAME - name of bbs. No more parameters defined so far. PUBSCREEN - Name of public screen to open evt. progress windows on. This parameter must be there even if it is not used by the packer.

CT_EventArchiver - Preferred archiver to use on event package is in (STRPTR) ti_Data. A NULL-pointer equals no archiving of the eventpackage. The archiver *must* have been configured in global configuration.

CT_SetTypeFlags - BBSType flags to set is in (LONGBITS) ti_Data.
CT_ClearTypeFlags - BBSType flags to clear is in (LONGBITS) ti_Data.
CT_AcceptPattern - Pattern to use when accepting grabs for this bbstype is pointed by (STRPTR) ti_Data. A NULL pointer equals to a pattern of #?.

CT_InitMsgFile - Command to initialize a message files is in (STRPTR) ti_Data. A NULL-pointer equals to no initializing other than just creating the file. This command is used by

```
UniqueMsgFile()
```

. The

command must except the following parameters:

BBSNAME/A - Name of bbs the message is for.

FILENAME/A - Name of file to initialized. (The file is already created.)

EVENT/N/A - What event type it should be used in.

USETAG/N/A - What message tag it should be used for.

CT_ExtConfig - Command to do external configuration for a bbs is in (STRPTR) ti_Data. A NULL-pointer equals to no external configuration. The command must except the following parameters:

BBSNAME/A - Name of bbs to configure.

CONFNAME - Name of current conference.

PUBSCREEN - Name of public screen to use for windows.

CT_QuoteType - Preferred quote type for this bbs type is in (UBYTE) ti_Data. See <libraries/BBSRead.h> for definitions of quote types.

RESULT

newtype - When changing an existing BBS type, this is the same as the type parameter. When adding a new BBS type this points to a TypeListItem for this new BBS type. On failure a NULL pointer is returned.

EXAMPLE

NOTES

When adding a BBS type, the TypeListItem for this BBS type will automatically be inserted in the list supplied with the CT_TypeList tag. Therefore, be sure the list is not attached to a ListView or similar when calling this function. The same applies when deleting BBS types.

You must supply the following tags when adding a new BBS type to the database: CT_TypeList, CT_TypeName, CT_LineLength, CT_SubjectLength, CT_MsgParser.

As of V4, CT_MsgParser is no longer needed when adding a new BBS type.

The commands given in the CT_MsgParser, CT_EventPacker, CT_InitMsgFile and CT_ExtConfig is expected to be given with path relative to the Thor home directory.

BUGS

SEE ALSO

GetTypeList()

1.19 bbsread.library/ConfLineLength()

NAME

ConfLineLength -- Returns max line length the bbs has in this conf.

SYNOPSIS

```
linelength = ConfLineLength( conf )
```

```
D0                A0
```

```
UWORD ConfLineLength( struct ConfListItem * );
```

FUNCTION

Use this function to obtain max line length the messages on the BBS is expected to have in this conference.

INPUTS

conf - Conf to get max linelength for.

RESULT

linelength - Linelength expected in this conf.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.20 bbsread.library/EndOfAdding()

NAME

EndOfAdding -- Call after adding a grab. (Used by MsgParser)

SYNOPSIS

```
void EndOfAdding( bbs )
                A0
```

```
void EndOfAdding( struct BBSListItem * );
```

FUNCTION

Function for MsgParser to call after finishing the parsing of a grab. UnLocks your access to add a grab to this BBS. Makes sure 2 or more processes do not add the same grab simultaneously.

Will also free the memory used to hold data during message adding.

Each call to

```
    StartOfAdding()
    must be coupled with a call to this
```

function.

INPUTS

bbs - Pointer to BBSListItem for BBS adding is finished.

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

```
    StartOfAdding()
```

1.21 bbsread.library/ExternalBBSConfig()

NAME

ExternalBBSConfig -- External configuration for bbs'es.

SYNOPSIS

```
error = ExternalBBSConfig( bbs, tagitems )
D0                A0        A1
```

```
LONG ExternalBBSConfig( struct BBSListItem *, struct TagItem * );
```

```
error = ExternalBBSConfigTags( bbs, Tag1, ... )
```

```
LONG ExternalBBSConfigTags( struct BBSListItem *, ULONG, ... );
```

FUNCTION

This function calls the command for external configuration defined in the bbstype of this bbs. If no external configuration command is defined, this function will return success.

INPUTS

bbs - Pointer to BBSListItem for bbs.
tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for ExternalBBSConfig():

EBC_Conference - Pointer to active conference is in (struct ConfListItem *) ti_Data. This tag may be omitted when no conferences are active.

EBC_PublicScreen - Public screen for external configuration to open windows on is in (STRPTR) ti_Data. When this tag is omitted the windows will be opened on the default public screen.

RESULT
error - 0 for success, result from command, or -1.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.22 bbsread.library/FindDupBRMsg()

NAME

FindDupBRMsg -- Find duplicate messages in database.

SYNOPSIS

```
error = FindDupBRMsg( bbs, tagitems )
D0                A0        A1
```

```
BOOL FindDupBRMsg( struct BBSListItem *, struct TagItem * );
```

```
error = FindDupBRMsgTags( bbs, Tag1, ... )
```

```
BOOL FindDupBRMsgTags( struct BBSListItem *, ULONG, ... );
```

FUNCTION

Searches the database for duplicate messages.

Supports callback progress hooks tags. BRProgress->brp_Actions will contain the number of duplicate messages found.

INPUTS

bbs - BBSListItem for the bbs to search for duplicate messages.
tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for FindDupBRMsg().

FDBRM_DeleteDupInConf - Mark duplicate messages in same conference as deleted if (BOOL) ti_Data is TRUE. Default is FALSE. The newest

message will be marked as deleted if duplicates are found.

FDBRM_UnMarkCrossPosts - Search through unread messages and mark crossposts as read if (BOOL) ti_Data is TRUE. Default is FALSE. If a instance of the crossposted message is found read in the database, all instances of it will be marked as read. If not, only the first instance of the crossposted message will be kept unread. (The conferences will be scanned in the same order as your conference list.) This tag is only useful on bbs'es where messages are identified with unique message identifiers.

RESULT

error - Boolean.

EXAMPLE

NOTES

If the message has a messageid this messageid is regarded as unique.

BUGS

SEE ALSO

1.23 bbsread.library/FindOrginalNr()

NAME

FindOrginalNr -- Find message by orginal number

SYNOPSIS

```
msgnr = FindOrginalNr( conf, orginalnr )
      D0                A0        D1
```

```
ULONG FindOrginalNr( struct ConfListItem *, ULONG );
```

FUNCTION

Scans the conference for a message with the passed orginal number.

INPUTS

conf - Conference to search in.
orginalnr - Orginal number to search for.

RESULT

msgnr - The number the message has locally. Returns 0 if the orginalnr can't be found.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.24 bbsread.library/FreeBRObject()

NAME

FreeBRObject -- Frees an object allocated

SYNOPSIS

```
void FreeBRObject( object )
                  A0
```

```
void FreeBRObject( void * );
```

FUNCTION

Frees an object allocated with any of the other functions in this library. Safe to call with a NULL pointer or a (APTR) -1 pointer.

INPUTS

object - Pointer to object to free.

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

1.25 bbsread.library/GetBBSList()

NAME

GetBBSList -- Returns a list of available BBS'es

SYNOPSIS

```
bbslist = GetBBSList()
D0
```

```
struct List * GetBBSList( void );
```

FUNCTION

Returns a Exec list of the available BBS'es. This list is your private READ-ONLY copy of the actual list, and you will not notice if anything is changed, i.e. BBS'es added or deleted.

Each node in the list has bl_Node.ln_Name set to bl_Data->bd_Name. You are free to use bl_Node.ln_Name for your own purposes.

The list must be deallocated with
FreeBRObject()

You are allowed to rearrange the order of the nodes in the list.
All nodes must be in the list when the list is deallocated.

INPUTS

RESULT

bbslist - Exec list of the available BBS'es. The list consist of BBSListNode structures. Returns a NULL pointer on failure.

EXAMPLE

NOTES

BUGS

SEE ALSO

<bbsread.h>

1.26 bbsread.library/GetConfigValue()

NAME

GetConfigValue -- Returns the configuration value to use.

SYNOPSIS

```
error = GetConfigValue( tagitems )
D0          A0
```

```
BOOL GetConfigValue( struct TagItem * );
```

```
error = GetConfigValueTags( Tag1, ... )
```

```
BOOL GetConfigValueTags( ULONG, ... );
```

FUNCTION

Get the correct value for a specified conference of BBS.

INPUTS

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for GetConfigValue().

GCV_GlobalConfig - GlobalConfig is pointed by (struct GlobalConfig *) ti_Data. Default value is NULL.

GCV_TypeListItem - TypeListItem is pointed by (struct TypeListItem *) ti_Data. Default value is NULL.

GCV_BBSListItem - BBSListItem is pointed by (struct BBSListItem *) ti_Data. Default value is NULL.

GCV_ConfListItem - ConfListItem is pointed by (struct ConfListItem *) ti_Data. Default value is NULL.

GCV_EventType - Type of event to use config value in is in (ULONG) ti_Data. This value is used by GCV_ConfReplyString and GCV_BBSReplyString tags. Default value of this tag is

EVE_REPLYMSG.

GCV_ConfQuoteType - Where to put conference quote type is pointed by (UBYTE *) ti_Data.

GCV_ConfQuoteChars - Where to put pointer to conference quote chars is pointed by (STRPTR *) ti_Data.

GCV_ConfQuoteReflow - Where to put conference reflow flag is pointed by (BOOL *) ti_Data.

GCV_BBSQuoteType - Where to put bbs quote type is pointed by (UBYTE *) ti_Data.

GCV_BBSQuoteChars - Where to put pointer to bbs quote chars is pointed by (STRPTR *) ti_Data.

GCV_BBSQuoteReflow - Where to put bbs reflow flag is pointed by (BOOL *) ti_Data.

GCV_ConfReplyString - Where to put pointer to conference reply string is pointed by (STRPTR *) ti_Data.

GCV_BBSReplyString - Where to put pointer to bbs reply string is pointed by (STRPTR *) ti_Data.

GCV_ConfEnterScript - Where to put pointer to name of conference enter script is pointed by (STRPTR *) ti_Data.

GCV_ConfLeaveScript - Where to put pointer to name of conference leave script is pointed by (STRPTR *) ti_Data.

GCV_BBSEnterScript - Where to put pointer to name of bbs enter script is pointed by (STRPTR *) ti_Data.

GCV_BBSLeaveScript - Where to put pointer to name of bbs leave script is pointed by (STRPTR *) ti_Data.

RESULT
error - Boolean

EXAMPLE

NOTES
The string pointers returned are only valid until GlobalConfig, TypeData, BBSData or ConfData are updated or freed.

BUGS

SEE ALSO

1.27 bbsread.library/GetConfList()

NAME

GetConfList -- Returns a list of available conferences in a BBS.

SYNOPSIS

```
conflist = GetConflist( bbs )
                A0
```

```
struct List * GetConflist( struct BBSListItem * );
```

FUNCTION

Returns an Exec list of the available conferences on a specified BBS. This list is your private READ-ONLY copy of the actual list, and you will not notice if anything is changed, i.e. conferences added or deleted.

Each node in the list has `cl_Node.ln_Name` set to `cd_Name`. You are free to use `cl_Node.ln_Name` for your own purposes. `cl_UserData` is set to `NULL`.

The list must be deallocated with

```
FreeBRObject()
```

You are allowed to rearrange the order of the nodes in the list. *All* nodes must be in the list when the list is deallocated.

INPUTS

`bbs` - Pointer to the `BBSListItem` to get conference list for.

RESULT

`conflist` - Exec list of the available conferences at the BBS. The list consists of `ConflistItem` structures. Returns a `NULL` pointer on failure.

EXAMPLE

NOTES

Will update the `bi_SumMarked` and `bi_SumM2User` fields of `bbs->bl_Internal`.

BUGS

SEE ALSO

1.28 bbsread.library/GetFAreaList()

NAME

`GetFAreaList` -- Returns a list of available file areas on a BBS.

SYNOPSIS

```
farealist = GetFAreaList( bbs )
                A0
```

```
struct List * GetFAreaList( struct BBSListItem * );
```

FUNCTION

Returns a Exec list of the available file areas on a specified BBS. This list is your private READ-ONLY copy of the actual list, and you will not notice if anything is changed, i.e. conferences added or deleted.

Each node in the list has `al_Node.ln_Name` set to `ad_Name`. You are free to use `al_Node.ln_Name` for your own purposes.

The list must be deallocated with
`FreeBRObject()`

.

INPUTS

`bbs` - Pointer to the `BBSListItem` to get file area list for.

RESULT

`farealist` - Exec list of the available file areas at the BBS. The list consist of `FAreaListItem` structures. Returns a NULL pointer on failure.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.29 bbsread.library/GetGlobalConfig()

NAME

`GetGlobalConfig` -- Returns a copy of the global configuration.

SYNOPSIS

```
globalcfg = GetGlobalConfig()
D0
```

```
struct GlobalConfig *GetGlobalConfig();
```

FUNCTION

Returns a copy of the global configuration. The structure is your private READ-ONLY copy of the actual structure. Beware that some STRPTR's in the returned structure could be NULL-pointers.

The structure must be deallocated with
`FreeBRObject()`

.

INPUTS

RESULT

`globalcfg` - Structure with global configuration. Returns a NULL

pointer on failure.

EXAMPLE

NOTES

Some of the STRPTR in the structure may be NULL-pointers.

BUGS

SEE ALSO

ConfigGlobal()

1.30 bbsread.library/GetMarkedMsg()

NAME

GetMarkedMsg -- Get messagenumbers of marked messages

SYNOPSIS

```
msgnrbuf = GetMarkedMsg( conf, usebuf, offset, numof )
D0                A0      A1      D1      D2
```

```
ULONG * GetMarkedMsg( struct ConfListItem *, ULONG *, ULONG, ULONG );
```

FUNCTION

Returns the mesagenumbers of the marked (unread) messages. If there is not enough messages to fill the supplied buffer, the rest of the entries will be NULL'ed out.

INPUTS

conf - Pointer to conference to get marked messages in.
usebuf - Pointer to the buffer to hold the messagenumbers in, must at least of (numof * sizeof(ULONG)) length.
offset - Where in the list over marked messages to start reading.
numof - Number of messagenumbers to get.

RESULT

msgnrbuf - Pointer to the buffer containing the messagenumbers.
On failure, a NULL pointer is returned.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.31 bbsread.library/GetSignature()

NAME

GetSignature -- Return the preferred signature.

SYNOPSIS

```
signature = GetSignature( globals, bbs, conf)
D0                A0      A1   A2
```

```
STRPTR GetSignature( struct GlobalConfig *, struct BBSListItem *,
    struct ConfListItem * );
```

FUNCTION

Returns the signature to use in a message in the conf conference in the bbs BBS. The pointer returned is only valid until the next time you call this function. If the preferred signature is a file signature, the signature will be loaded and the returned pointer will point to a buffer holding the signature.

INPUTS

globals - Pointer to your copy of the global configuration.
 bbs - Pointer to BBSListItem for bbs to get signature for. Can be a NULL-pointer.
 conf - Pointer to ConfListItem for the conference to get signature for. Can be a NULL-pointer.

RESULT

signature - Returns a string pointer for the signature to use. Returns a NULL-pointer if there is no defined signature or if the function failed. On failure, IoErr() will be non-zero. Will also return a NULL pointer if a NO_SIGNATURE flag is set.

EXAMPLE

NOTES

The pointer returned may be a copy of the Signature string pointer in either GlobalConfig, BBSData or ConfData. Therefore the pointer is only valid as long as you don't free or update any of these structures.

BUGS

SEE ALSO

1.32 bbsread.library/GetTagFile()

NAME

GetTagFile -- Return the preferred tag file.

SYNOPSIS

```
tagfile = GetTagFile( globals, bbs, conf)
D0                A0      A1   A2
```

```
STRPTR GetTagFile( struct GlobalConfig *, struct BBSListItem *,
    struct ConfListItem * );
```

FUNCTION

Returns the tag file to use in a message in the conf conference in the bbs BBS.

INPUTS

globals - Pointer to your copy of the global configuration.
 bbs - Pointer to BBSListItem for bbs to get tag file for. Can be a NULL-pointer.
 conf - Pointer to ConfListItem for the conference to get tag file for. Can be a NULL-pointer.

RESULT

tagfile - Returns a string pointer for the tag file to use. Returns a NULL-pointer if there is no defined tag file. Will also return a NULL pointer if a NO_TAG flag is set.

EXAMPLE**NOTES**

The pointer returned is a copy of the tag file string pointer in either GlobalConfig, BBSData or ConfData. Therefore the pointer is only valid as long as you don't free or update any of these structures.

BUGS**SEE ALSO**

1.33 bbsread.library/GetTypeList()

NAME

GetTypeList -- Returns a list of available BBS types

SYNOPSIS

```
typelist = GetTypeList()
D0
```

```
struct List * GetTypeList( void );
```

FUNCTION

Returns an Exec list with the available BBS types. This list is your private READ-ONLY copy of the actual list, and you will not notice if anything is changed, i.e. BBS types added or deleted.

Each node in the list has tl_Node.ln_Name set to tl_Data->td_TypeName. You are free to use tl_Node.ln_Name for your own purposes.

The list must be deallocated with
 FreeBRObject()

INPUTS**RESULT**

typelist - Exec list of the available BBS types. The list consist of

TypeListNode structures. Returns a NULL pointer on failure.

EXAMPLE

NOTES

BUGS

SEE ALSO

<bbsread.h>

1.34 bbsread.library/MakeEventPackage()

NAME

MakeEventPackage -- Make event package for a bbs

SYNOPSIS

```
error = MakeEventPackage( bbs, tagitems )
D0                A0        A1
```

```
LONG MakeEventPackage( struct BBSListItem *, struct TagItem * );
```

```
error = MakeEventPackageTags( bbs, Tag1, ... )
```

```
LONG MakeEventPackageTags( struct BBSListItem *, ULONG, ... );
```

FUNCTION

This function calls up the command to pack and archive the events for this BBS. The command used is the one set up in the typedata for the BBS. This function returns success if no eventpacker is configured for the bbstype of this bbs.

The tags that are not understood are forwarded to dos.library/SystemTagList(). Look at dos.library/SystemTagList() for further information on tags.

The BDF_EVENTS_CHANGED flag will be cleared for this bbs if this function returns success.

INPUTS

bbs - Pointer to BBSListItem for bbs.

tagitems - Pointer to TagItem array. Tags not defined for MakeEventPackage() are passed to dos.library/SystemTagList(). See <dos/dostags.h>. Both dos.library/SystemTagList() tags and dos.library/CreateNewProc() tags may be passed.

Here are the TagItem.ti_Tag values that are defined for MakeEventPackage().

EP_PublicScreen - Public screen for eventpacker to open possible progress windows on is in (STRPTR) ti_Data. The public screen name is forwarded to the packer as a parameter.

RESULT

error - 0 for success, result from command, or -1. Note that on error, the caller is responsible for any filehandles or other things passed in via tags.

EXAMPLE

NOTES

bbs->bl_Data->bd_Flags will be updated by MakeEventPackage().

BUGS

SEE ALSO

dos.library/SystemTagList(), dos.library/CreateNewProc(),
<dos/dostags.h>

1.35 bbsread.library/MarkMessage()

NAME

MarkMessage -- Mark or unmark messages

SYNOPSIS

```
error = MarkMessage( conf, tagitems )
```

```
D0                A0        A1
```

```
BOOL MarkMessage( struct ConfListItem *, struct TagItem * );
```

```
error = MarkMessageTags( conf, Tag1, ... )
```

```
BOOL MarkMessageTags( struct ConfListItem *, ULONG, ... );
```

FUNCTION

Use this function to mark or unmark messages.

INPUTS

conf - Pointer to conference to mark or unmark messages in.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for MarkMessage().

MM_MarkMessage - The messagenumber to mark is in (ULONG) ti_Data.

MM_MarkArray - An array of ULONG containing messages to mark is pointed to by (ULONG *) ti_Data. A NULL pointer terminates the array.

MM_UnMarkMessage - The messagenumber to unmark is in (ULONG) ti_Data.

MM_UnMarkArray - An array of ULONG containing messages to unmark is pointed to by (ULONG *) ti_Data. A NULL pointer terminates the array.

MM_SuperMarking - The mark tags will set the MDF_SUPERMARKED flag and the unmark tags will clear the MDF_SUPERMARKED flag if (BOOL) ti_Data is TRUE.

MM_Reset - Unmark all marked messages if (BOOL) ti_Data is TRUE.

MM_MineFirst - Move marked messages to user first if (BOOL) ti_Data is TRUE.

MM_Reference - Move marked messages in reference order if (BOOL) ti_Data is TRUE.

MM_ToAllFirst - Move marked messages to all first if (BOOL) ti_Data is TRUE.

MM_SortByMsgNumbers - Sort the marked messages by its messagenumbers if (BOOL) ti_Data is TRUE.

MM_GroupSubject - Group marked messages by subject if (BOOL) ti_Data is TRUE.

MM_Reverse - Reverse the order of the marked messages if (BOOL) ti_Data is TRUE.

MM_SortAlphabetical - Sort alphabetically on subject or author (depending on what other tag is given in) if (BOOL) ti_Data is TRUE.

MM_SortByAuthor - Sort by author names if (BOOL) ti_Data is TRUE.

RESULT
error - Returns TRUE on failure.

EXAMPLE

NOTES
Super marked messages can only be unmarked when the MM_SuperMarking tag is set to TRUE.

BUGS

SEE ALSO

1.36 bbsread.library/PackDataFile()

NAME

PackDataFile -- Removes all deleted entries from a datafile

SYNOPSIS

```
fail = PackDataFile( tagitems )
D0          A0
```

```
struct TagItem * PackDataFile( struct TagItem * );
```

```
fail = PackDataFileTags( Tag1, ... )
```

```
struct TagItem * PackDataFileTags( ULONG, ... );
```

FUNCTION

A function for 'packing' datafiles. This means removing all deleted information from the datafile. This function supports callback progress hooks.

INPUTS

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for PackDataFile().

PD_EventData - Clean up event data base for a bbs. The bbs to clean up is pointed by (struct BBSListItem *) ti_Data. No progress callback for PD_EventData.

PD_Conference - Pack the datafiles of the conference given in (struct ConfListItem *) ti_Data. Messages will be deleted according to ConfData.cd_KeepMsg and ConfData.cd_KeepTime. A conference is packed in two passes. First pass deletes messages, and the second pass actually packs the message data files. BRProgress.brp_Actions is only used for the first pass.

PD_AttachmentList - This is a (struct List **) list consisting of (struct AttachmentItem *) nodes that will contain the file attachments included on messages that are purged when PD_Conference is used. Free it using
FreeBRObject()
when
you are done with it. This list will be valid but empty if no attachments are found (but still needs to be freed).

PD_UserData - Pack the datafiles for the user database on the bbs pointed by (struct BBSListItem *) ti_Data.

PD_KillData - Pack the datafiles for the kill database on the bbs pointed by (struct BBSListItem *) ti_Data.

PD_FileData - Pack the datafiles for the file database on the bbs pointed by (struct BBSListItem *) ti_Data.

PD_SavePackedBRIEF - Name of file to save packed messages in BRIEF format is pointed by (STRPTR) ti_Data. All messages which are deleted because of KeepTime and KeepMsg will be saved to this file. If the file exists, the messages will be appended to the file. Default is not to save any messages. This tag is ignored if the PD_Conference tag is not given.

RESULT

fail - NULL on success. On failure, it points to the tag which caused the failure.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.37 bbsread.library/ParseGrab()

NAME

ParseGrab -- Parse a grab and add it's messages.

SYNOPSIS

```
error = ParseGrab( grabnode, tagitems )
D0                A0        A1
```

```
LONG ParseGrab( struct Node *, struct TagItem * );
```

```
error = ParseGrabTags( grabnode, Tag1, ... )
```

```
LONG ParseGrabTags( struct Node *, ULONG, ... );
```

FUNCTION

This calls up the command for adding messages in a grab to the message database. The grabnode parameter *must* be a node from a list returned by

```
ScanForGrabs()
```

. The command used is the one set up in the typedata for the BBS.

The tags that are not understood are forwarded to dos.library/SystemTagList(). Look at dos.library/SystemTagList() for further information on tags.

INPUTS

grabnode - Pointer to Node structure.

tagitems - Pointer to TagItem array. See <dos/dostags.h>. Both dos.library/SystemTagList() tags and dos.library/CreateNewProc() tags may be passed.

Here are the TagItem.ti_Tag values that are defined for ParseGrab().

PG_PublicScreen - Public screen for eventpacker to open possible progress windows on is in (STRPTR) ti_Data. The public screen name is forwarded to the parser as a parameter. This screen name will also be used if ParseGrab() opens any reqtools requesters.

PG_RequestWindow - Reference window for requesters opened by the ParseGrab() function is pointed by (struct Window *) ti_Data. If this tag is omitted or is NULL and reqtools is not available will requesters appear on the default public screen.

RESULT

error - 0 for success, result from command, or -1. Note that on error, the caller is responsible for any filehandles or other things passed in via tags.

EXAMPLE

NOTES

This function uses reqtools requesters if reqtools.library is available. The reqtools requesters opens on PG_PublicScreen if this tag is given.

BUGS

SEE ALSO

dos.library/SystemTagList(), dos.library/CreateNewProc(),
<dos/dostags.h>,
ScanForGrabs()

1.38 bbsread.library/PGPBREvents()

NAME

PGPBREvents -- PGP sign and/or encrypt events.

SYNOPSIS

```
error = PGPBREvents( bbs, tagitems )
D0                A0        A1
```

```
ULONG PGPBREvents( struct BBSListItem *, struct TagItem * );
```

```
error = PGPBREventsTags(bbs, Tag1, ...)
```

```
ULONG PGPBREventsTags(struct BBSListItem *, ULONG, ... );
```

FUNCTION

This function will PGP sign and/or encrypt BREV_MsgFile files if the BREV_PGPSignID and/or BREV_PGPDecryptID tag is used in an event. The encrypted and/or signed message (with ascii armour) is stored on the disk under the same main name as the original text file, but with an added .asc extension.

This function is meant to be used by event packers before packing any events.

Will not PGP sign and/or encrypt if the file containing the encrypted/signed message is older than the BREV_MsgFile or the event hasn't been changed since the encryption/signing.

INPUTS

bbs - BBS to sign and/or encrypt events on.

tagitems - Pointer to TagItem array. Tags not defined for PGPBREvents() are passed to dos.library/SystemTagList(). See <dos/dostags.h>. Both dos.library/SystemTagList() tags and dos.library/CreateNewProc() tags may be passed.

Here are the TagItem.ti_Tag values that are defined for PGPBREvents().

PGP_PublicScreen - Public screen name for pass phrase requester (needed when signing messages) is pointed (STRPTR) ti_Data.

Default is default public screen.

RESULT

error - 0 on success. On failure will the number of the event which failed be returned.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.39 bbsread.library/ReadBREvent()

NAME

ReadBREvent -- Read a event from the database.

SYNOPSIS

```
eveobj = ReadBREvent( bbs, eventnr, tagitems )
D0                A0    D1    A1
```

```
APTR ReadBREvent( struct BBSListItem *, ULONG, struct Tagitem * );
```

```
eveobj = ReadBREventTags( bbs, eventnr, Tag1, ... )
```

```
APTR ReadBREventTags( struct BBSListItem *, ULONG, ULONG, ... );
```

FUNCTION

Reads the data about an event from the database.

INPUTS

bbs - Pointer to BBSListItem structure returned in
GetBBSList()

eventnr - Number of the event to get data for.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for
ReadBREvent().

RBREV_EventTagsPtr - Where to put a pointer to the loaded event
tagarray is pointed by (struct TagItem **) ti_Data. This tagarray
contains all the BREV_#? tags for this event. These are the same
as supplied with

```
WriteBREvent()
```

when this event was added to the
database. If the event is marked as deleted or if ReadBREvent()
fails, *(struct TagItem **) ti_Data) will be set to NULL.

RBREV_EventType - Where to put the type of this event is pointed by
(ULONG *) ti_Data.

RBREV_EventDate - Where to put the date when the event was added to

the database is pointed by (ULONG *) ti_Data. The date is in seconds from 01-Jan-1978.

RBREV_Flags - Where to put the flags for this event is pointed by (LONGBITS *) ti_Data. See <libraries/bbsread.h> for definitions.

RESULT

eveobj - Returns NULL on failure. On success, eveobj *must* be passed to

```
FreeBRObject()
    if it is anything else than (APTR) -1.
```

EXAMPLE

NOTES

No need to

```
FreeBRObject()
    eveobj if eveobj is (APTR) -1.
```

You are allowed to change the returned strings. (They must not be made any longer)

BUGS

SEE ALSO

1.40 bbsread.library/ReadBRFile()

NAME

ReadBRFile -- Read the data for a file from the database.

SYNOPSIS

```
fileobj = ReadBRFile( farea, filenr, tagitems )
D0                A0      D1      A1
```

```
APTR ReadBRFile( struct FAreaListItem *, ULONG, struct Tagitem * );
```

```
fileobj = ReadBRFileTags( farea, filenr, Tag1, ... )
```

```
APTR ReadBRFileTags( struct FAreaListItem *, ULONG, ULONG, ... );
```

FUNCTION

Reads the data for a file from the database.

INPUTS

farea - Pointer to FAreaListItem structure returned in
GetFAreaList()

filenr - Number of file to get data for.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for ReadBRFile().

RBRF_FileTagsPtr - Where to put a pointer to the loaded file tagarray is pointed by (struct TagItem **) ti_Data. This tagarray contains all the BRFILE_#? tags for this file. These are the same as supplied with

```
WriteBRFile()
```

when this file was written to the database. If the file is marked as deleted or if ReadBRFile() fails, *(struct TagItem **) ti_Data) will be set to NULL.

RBRF_FileDate - Where to put the date when the file was added to the database is pointed by (ULONG *) ti_Data. The date is in seconds from 01-Jan-1978.

RBRF_Flags - Where to put the flags for this file is pointed by (LONGBITS *) ti_Data. See <libraries/bbsread.h> for definitions.

RBRF_NextFile - Where to put the number of the next file in the same file area as this file is pointed by (ULONG *) ti_Data. This tag *must* be used when traversing through all files in a file area. The number of the first file in a file area is in the file areas FAreaData structure. A 0 means there are no more files in this file area.

RESULT

userobj - Returns NULL on failure. On success, fileobj *must* be passed to

```
FreeBRObject()
```

if it is anything else than (APTR) -1.

EXAMPLE

NOTES

No need to

```
FreeBRObject()
```

fileobj if fileobj is (APTR) -1.

You are allowed to change the returned strings. (They must not be made any longer)

BUGS

SEE ALSO

1.41 bbsread.library/ReadBRKill()

NAME

ReadBRKill -- Read the data for a kill from the database.

SYNOPSIS

```
killobj = ReadBRKill( bbs, killnr, tagitems )
```

```
D0                A0    D1    A1
```

```
ULONG ReadBRKill( struct BBSListItem *, ULONG, struct Tagitem * );
```

```
killobj = ReadBRKillTags( bbs, killnr, Tag1, ... )
```

```
ULONG ReadBRKill( struct BBSListItem *, ULONG, ULONG, ... );
```

FUNCTION

Read the data for a kill from the database.

INPUTS

bbs - Pointer to BBSListItem structure returned in
GetBBSList()

killnr - Number of kill to get data for.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for
ReadBRKill().

RBRK_KillTagsPtr - Where to put a pointer to the loaded kill tagarray
is pointed by (struct TagItem **) ti_Data. This tagarray contains
all the BRMSG_#? tags for this kill. These are the same as
supplied with

```
WriteBRKill()
```

when this kill was written to the
database. If the kill is marked as deleted or if ReadBRKill()
fails, *((struct TagItem **) ti_Data) will be set to NULL.

RBRK_KillDate - Where to put the date when the kill was added to the
database is pointed by (ULONG *) ti_Data. The date is in seconds
from 01-Jan-1978.

RBRK_LastKill - Where to put the date when this kill last killed is
pointed by (ULONG *) ti_Data. The date is in seconds from
01-Jan-1978.

RBRK_Flags - Where to put the flags for this kill is pointed by
(LONGBITS *) ti_Data. See <libraries/bbsread.h> for definitions.

RESULT

killobj - Returns NULL on failure. On success, killobj *must* be passed
to

```
FreeBRObject()
```

if it is anything else than (APTR) -1.

EXAMPLE

NOTES

No need to

```
FreeBRObject()
```

userobj if userobj is (APTR) -1.

You are allowed to change the returned strings. (They must not be made
any longer)

BUGS

SEE ALSO

1.42 bbsread.library/ReadBRMessage()

NAME

ReadBRMessage -- Read a message from the database.

SYNOPSIS

```
msgobj = ReadBRMessage( conf, msgnr, tagitems )
D0          A0      D1      A1
```

```
APTR ReadBRMessage( struct ConflistItem *, ULONG, struct TagItem * );
```

```
msgobj = ReadBRMessageTags( conf, msgnr, Tag1, ... )
D0          A0      D1      A1
```

```
APTR ReadBRMessageTags( struct ConflistItem *, ULONG, ULONG, ... );
```

FUNCTION

Read data about a message from the database.

The multiple parts in multipart messages should be presented in the order they are found in the taglist. BRMSG_Text should always be presented first.

INPUTS

conf - Pointer to ConflistItem structure returned in
GetConflist

msgnr - Number of the message to get data for.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for
ReadBRMessage().

RBRMSG_MsgTagsPtr - Where to put a pointer to the loaded message
tagarray is pointed by (struct TagItem **) ti_Data. This tagarray
contains all the BRMSG_#? tags for this message. These are the
same as supplied with
WriteBRMessage()
when this message was added
to the database. If the message is marked as deleted or if
ReadBRMessage() fails, *(struct TagItem **) ti_Data) will be set to
NULL.

RBRMSG_MsgDate - Where to put the date when the message was added to
the database is pointed by (ULONG *) ti_Data. The date is in
seconds from 01-Jan-1978.

RBRMSG_Reference - Where to put the number of the message this message
refers to is pointed by (ULONG *) ti_Data. If the reference is 0,
this message has no references.

RBRMSG_FirstRef - Where to put the number of the first message which

refers to this message is pointed by (ULONG *) ti_Data.

RBRMSG_LastRef - Where to put the number of the last message which refers to this message is pointed by (ULONG *) ti_Data.

RBRMSG_PrevRef - Where to put the number of the previous message which refers to the same message as this message is pointed by (ULONG *) ti_Data.

RBRMSG_NextRef - Where to put the number of the next message which refers to the same message as this message is pointed by (ULONG *) ti_Data.

RBRMSG_Flags - Where to put the flags for this message is pointed by (LONGBITS *) ti_Data. See <libraries/bbsread.h> for definitions.

RBRMSG_GetHeader - Tags considered as header fields are returned in the message tagarray if (BOOL) ti_Data is TRUE. Default is TRUE. To be used in combination with RBRMSG_MsgTagsPtr. What tags which are considered as header fields are defined in <libraries/bbsread.h>

RBRMSG_GetText - Tags considered as text fields are returned in the message tagarray if (BOOL) ti_Data is TRUE. Default is TRUE. To be used in combination with RBRMSG_MsgTagsPtr. What tags which are considered as text fields are defined in <libraries/bbsread.h>

RESULT

msgobj - Returns NULL on failure. On success, msgobj *must* be passed to

```
FreeBRObject()
    if it is anything else than (APTR) -1.
```

EXAMPLE

NOTES

No need to

```
FreeBRObject()
    msgobj if msgobj is (APTR) -1.
```

You are allowed to change the returned strings. (They must not be made any longer)

BUGS

SEE ALSO

1.43 bbsread.library/ReadBRUser()

NAME

ReadBRUser -- Read the data for an user from the database.

SYNOPSIS

```
userobj = ReadBRUser( bbs, usernr, tagitems )
```

```

D0                A0      D1      A1

APTR ReadBRUser( struct BBSListItem *, ULONG, struct Tagitem * );

userobj = ReadBRUserTags( bbs, usernr, Tag1, ... )

APTR ReadBRUserTags( struct BBSListItem *, ULONG, ULONG, ... );

```

FUNCTION

Reads the data for an user from the database.

INPUTS

bbs - Pointer to BBSListItem structure returned in
 GetBBSList()
 .

eventnr - Number of user to get data for.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for
 ReadBRUser().

RBRUSR_UserTagsPtr - Where to put a pointer to the loaded user
 tagarray is pointed by (struct TagItem **) ti_Data. This tagarray
 contains all the BRUSR_#? tags for this user. These are the same
 as supplied with
 WriteBRUser()
 when this user was written to the
 database. If the user is marked as deleted or if ReadBRUser()
 fails, *(struct TagItem **) ti_Data) will be set to NULL.

RBRUSR_UserDate - Where to put the date when the user was added to the
 database is pointed by (ULONG *) ti_Data. The date is in seconds
 from 01-Jan-1978.

RBRUSR_Flags - Where to put the flags for this user is pointed by
 (LONGBITS *) ti_Data. See <libraries/bbsread.h> for definitions.

RESULT

userobj - Returns NULL on failure. On success, userobj *must* be passed
 to
 FreeBRObject()
 if it is anything else than (APTR) -1.

EXAMPLE

NOTES

No need to
 FreeBRObject()
 userobj if userobj is (APTR) -1.

You are allowed to change the returned strings. (They must not be
 made any longer)

BUGS

SEE ALSO

1.44 bbsread.library/ReadPassiveConfList()

NAME

ReadPassiveConfList -- Read the passive conference list.

SYNOPSIS

```
passConfList = ReadPassiveConfList( bbs )
D0                                     A0
```

```
struct List * ReadPassiveConfList( struct BBSListItem * );
```

FUNCTION

Gives you a list of all conferences in the passive conference list datafiles. The passive conference list is a list of **all** available conferences at the bbs. It should be used when the user want to send a join event. Returns a list of struct PassConfListItem. The pl_Node.ln_Name pointer equals the pl_Name pointer.

INPUTS

bbs - Pointer to the bbs which the list should be read from.

RESULT

passConfList - Pointer to the list header of the passive conference list. The list **must** be freed with a call to FreeBRObject()

Returns NULL on failure or if no passive conference list is available. IoErr() will be set on failure.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.45 bbsread.library/ScanForGrabs()

NAME

ScanForGrabs -- Scans download directory for grabs.

SYNOPSIS

```
grablist = ScanForGrabs( void )
D0
```

```
struct MinList * ScanForGrabs( void );
```

FUNCTION

Scans the download directory for grabs. The BBS'es with waiting grabs is returned in a list of struct Node. Node.ln_Name points to the name of the BBS.

If there are more than one grab from a BBS, there will be a node in the list for each grabfile. The nodes in the list are sorted on the name of the grabfile. By this the grabs are sorted correct if they are numbered. The grabs should be parsed in the order they are found in the list.

The list must be deallocated with
FreeBRObject()
.

INPUTS

RESULT

grablist - Pointer to MinList structure. List contains Node structures. Returns an empty list if there are no new grabs. Returns NULL-pointer on failure and sets IoErr() if possible.

EXAMPLE

NOTES

Returns NULL-pointer if gc_DnloadPath isn't set with GlobalConfig().

BUGS

SEE ALSO

1.46 bbsread.library/SearchBRFile()

NAME

SearchBRFile -- Search file database.

SYNOPSIS

```
found = SearchBRFile( tagitems )
D0          A0
```

```
struct SFileResult * SearchBRFile( struct TagItem * );
```

```
found = SearchBRFileTags( Tag1, ... )
```

```
struct SFileResult * SearchBRFileTags( ULONG, ... );
```

FUNCTION

This function scans the filedatabase for a file matching the given searchkey. It's possible to search a file area or all file areas on a bbs.

Supports callback progress hooks tags.

INPUTS

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for SearchBRFile().

SBRF_SearchFAreaList - List of file areas to search in is pointed by (struct List *) ti_Data. The list *MUST* have been obtained with

```
GetFAreaList()
```

. The point of this tag is to be able to search all file areas on a bbs in one call.

SBRF_SearchFArea - Search the file area pointed by (struct FAreaListItem *) ti_Data. This tag has higher priority than the SBRF_SearchFAreaList tag.

SBRF_SearchStr - String to search for is pointed by (STRPTR) ti_Data. Wildcards are allowed.

SBRF_SearchName - Search for a file with a name matching SBRF_SearchStr exact if (BOOL) ti_Data is TRUE. Default is FALSE.

SBRF_SearchAll - Search all strings for a match with SBRF_SearchStr if (BOOL) ti_Data is TRUE. Default is FALSE. This tag has higher priority than the SBRF_SearchName tag.

SBRF_NewerThan - Find files newer than (ULONG) ti_Data. Time is in seconds since 1.January 1978

RESULT

found - Pointer to a SFileResult structure containing the results of the search. The SFileResult structure must be deallocated with

```
FreeBRObject()
```

. Returns a NULL pointer if no matches where found or if the function failed. IoErr() will be non null on failure.

EXAMPLE

NOTES

This function will fail if neither SBRF_SearchBBS nor SBRF_SearchFArea is given in the taglist.

If found->fr_NextResult is non NULL, a linked list of SFileResult structures is returned.

BUGS

SEE ALSO

1.47 bbsread.library/SearchBRMessage()

NAME

SearchBRMessage -- Search for messages in a conference.

SYNOPSIS

```
found = SearchBRMessage( conf, tagitems )
D0          A0          A1
```

```
struct SearchResult * SearchBRMessage( struct ConflListItem *,
    struct TagItem * );
```

```
found = SearchBRMessageTags( conf, tag1, ... )
```

```
struct SearchResult * SearchBRMessageTags( struct ConflListItem *,
    ULONG, ... );
```

FUNCTION

Search through the messages in a conference. The search is not case sensitive.

Standard AmigaDOS wildcards are supported.

Supports callback progress hooks tags.

INPUTS

conf - Conference to search in.
tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for SearchBRMessage().

SC_FindString - String to search for is in (STRPTR) ti_Data. There can be searched for more than one string on each call by using this tag more than once. As of V3, this tag is no longer needed when searching.

SC_FromUser - Search for messages from the user in (STRPTR) ti_data.

SC_ToUser - Search for messages to the user in (STRPTR) ti_data.
Overrides the SC_ToAll tag.

SC_FromMessage - Message number to start searching at is in (ULONG) ti_Data. Default is to start at the first message in the conference.

SC_ToMessage - Message number to end searching at is in (ULONG) ti_Data. Default is to end at the last message in the conference.

SC_SearchSubject - Search in the subject if (BOOL) ti_Data is TRUE.
Default value for this tag is TRUE.

SC_SearchMessage - Search in the message if (BOOL) ti_Data is TRUE.
Default value for this tag is TRUE.

SC_SearchComment - Search in the comment if (BOOL) ti_Data is TRUE.
Default value for this tag is TRUE.

SC_SearchMsgID - Search in the message id if (BOOL) ti_Data is TRUE. Default value for this tag is TRUE.

SC_MessageArray - Message numbers to search is in a NULL-terminated

array of ULONG pointed by (ULONG *) ti_Data. This tag overrides the SC_FromMessage and SC_ToMessage tags.

SC_ToAll - Search for messages to ALL if (BOOL) ti_Data is TRUE.

SC_KeptMessages - Search for messages with the MDF_KEEP flag set if (BOOL) ti_Data is TRUE.

SC_NewerThan - Search for messages newer than (ULONG) ti_Data. The time is in seconds since 1.January 1978.

SC_OlderThan - Search for messages older than (ULONG) ti_Data. The time is in seconds since 1.January 1978.

RESULT

found - Pointer to a SearchResult structure containing the results of the search. The SearchResult structure must be deallocated with

```
FreeBRObject()
```

. Returns a NULL pointer if no matches where found or the function failed. IoErr() will be non null on failure.

EXAMPLE

NOTES

If found->sr_NextResult is non NULL, a linked list of SearchResult structures is returned.

The found->sr_Messages[] array is garantied to be NULL terminated.

BUGS

SEE ALSO

1.48 bbsread.library/SearchBRUser()

NAME

SearchBRUser -- Search user database.

SYNOPSIS

```
found = SearchBRUser( bbs, tagitems )
D0                A0        A1
```

```
struct SUserResult * SearchBRUser( struct BBSListItem *,
    struct TagItem * );
```

```
found = SearchBRUserTags( bbs, Tag1, ... )
```

```
struct SUserResult * SearchBRUserTags( struct BBSListItem *, ULONG,
    ... );
```

FUNCTION

This function scans the userdatabase for an user at the given BBS. When suggesting users, SoundEx hashing will be used to suggest user

names which match the search string. If there are more than one name in the search string, the first and the last will be used to narrow the suggestions. The search order is aliases, names, addresses and comment.

Standard AmigaDOS wildcards is supported.

INPUTS

bbs - Pointer to BBS to search in.
namestr - Name to search for.
tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for SearchBRUser().

SBRU_SearchStr - String to search for is pointed by (STRPTR) ti_Data.

SBRU_SearchName - Search for a name matching SBRU_SearchStr exact if (BOOL) ti_Data is TRUE. Default is TRUE.

SBRU_SearchAddr - Search for a address matching SBRU_SearchStr exact if (BOOL) ti_Data is TRUE. Default is TRUE.

SBRU_SearchAlias - Search for a alias matching SBRU_SearchStr exact if (BOOL) ti_Data is TRUE. Default is TRUE.

SBRU_SearchComment - Search in comment for a match to SBRU_SearchStr if (BOOL) ti_Data is TRUE. Default is FALSE. When wildcards are used, the search pattern must match a line in the comment.

SBRU_SuggestUsers - A list of suggestions for user names if SBRU_SearchStr doesn't match any user names in the database will be pointed by *((struct MinList **) ti_Data) when this function returns. The list will consist of UserSuggestion structures. The list pointer may be a NULL-pointer if no users where found or if there was a failure. NB: The list *must* be freed with

FreeBRObject()

. This tag is ignored when the SBRU_SearchName tag is FALSE or if SBRU_SearchStr contains wildcards.

RESULT

found - Pointer to a SUserResult structure containing the results of the search. The SUserResult structure must be deallocated with

FreeBRObject()

. Returns a NULL pointer if no matches where found or if the function failed. IoErr() will be non null on failure.

EXAMPLE

NOTES

If found->ur_NextResult is non NULL, a linked list of SUserResult structures is returned.

BUGS

SEE ALSO

1.49 bbsread.library/SortMessageArray()

NAME

SortMessageArray -- Sorts messagenumbers in an array by the given method (V5
)

SYNOPSIS

```
msgnrbuf = SortMessageArray( conf, usebuf, method )
```

```
D0                A0    A1    D0
```

```
ULONG * SortMessageArray( struct ConflListItem *, ULONG * );
```

FUNCTION

Sorts the NULL-terminated array usebuf according to the given method.

INPUTS

conf - Pointer to conference to use for the array

usebuf - Pointer to the buffer that contains the messagenumbers.

NOTE: The buffer must be NULL-terminated.

method - What method should be used when sorting the messages. See

<libraries/bbsread.h> for definitions

RESULT

msgnrbuf - Pointer to the buffer containing the messagenumbers.

On failure, a NULL pointer is returned.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.50 bbsread.library/StartOfAdding()

NAME

StartOfAdding -- Call before adding a grab. (Used by MsgParser)

SYNOPSIS

```
void StartOfAdding( bbs )
```

```
A0
```

```
void StartOfAdding( struct BBSListItem * );
```

FUNCTION

Function for MsgParser to call before doing anything with UnArchiving or parsing. Locks your access to add a grab to this BBS. Makes sure 2 or more processes do not add the same grab simultaneously.

Each call to this function must be coupled with a call to

```
EndOfAdding()
```

.

INPUTS

bbs - Pointer to BBSListItem for BBS to start adding to.

RESULT

EXAMPLE

NOTES

BUGS

SEE ALSO

```
EndOfAdding()
```

1.51 bbsread.library/TypeFromBBS()

NAME

TypeFromBBS - Returns the TypeListItem structure of a BBS.

SYNOPSIS

```
bbstype = TypeFromBBS( bbs )
```

```
D0                               A0
```

```
struct TypeListItem * TypeFromBBS( struct BBSListItem * );
```

FUNCTION

Returns the TypeListItem structure correspondig to the type of the bbs parameter. Use this function instead of searching the type list by name for the correspondig TypeListItem.

The TypeListItem structure returned is not a part of a list, so don't use the tl_Node. The structure must be deallocated with

```
FreeBRObject()
```

.

INPUTS

bbs - Pointer to the BBSListItem to get the type of.

RESULT

bbstype - Pointer to a TypeListItem for the bbs. Returns a NULL pointer on failure.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.52 bbsread.library/UnArchive()

NAME

UnArchive -- Unarchives an archive.

SYNOPSIS

```
error = UnArchive( archive, tagitems )
D0                A0        A1
```

```
BOOL UnArchive( STRPTR, struct TagItem * );
```

```
error = UnArchiveTags( archive, Tag1, ...)
```

```
LONG UnArchiveTags( STRPTR, ULONG, ...);
```

FUNCTION

This function unarchives an archive. The type of archiver to use is determined with the global configuration for the library. If no archivetypes match this archive, the function will return failure. The ability to return failure when the unarchivers fail depends on how the unarchivers behave.

Files which match no known archiver will be copied to the destination. Files with .txt extension and files without extension are treated as text files and trailing numbers in the file name will be removed. (.txt extension will also be removed)

The tags not understood are forwarded to dos.library/SystemTagList(). Look at dos.library/SystemTagList() for further information on tags.

INPUTS

archive - Pointer to path and name of archive to unarchive.
tagitems - Pointer to TagItem array. See <dos/dostags.h>. Both dos.library/SystemTagList() tags and dos.library/CreateNewProc() tags may be passed.

Here are the TagItem.ti_Tag values that are defined for UnArchive().

UA_RetrieveFile - File to retrieve from archive is in (STRPTR) ti_Data. More than one of this tag can be passed. If this tag is omitted, all files in archive will be unarchived.

UA_DestDir - Path to destination directory is in (STRPTR) it_Data. Default is to use current directory as destination directory.

UA_ArchiverUsed - Where to put a pointer to the ArcConfigItem structure for the archiver used in archive is pointed by (struct ArcConfigItem **) ti_Data. The ArcConfigItem structure returned is not a part of a list, so don't use the ac_Node. The

structure must be deallocated with
 FreeBRObject()
 . You get a
 NULL when no identifiable archiver is used or the function failed
 with -1.

RESULT
 error - 0 for success, result from archiver, or -1. Note that on
 error, the caller is responsible for any filehandles or other
 things passed in via tags.

EXAMPLE

NOTES

BUGS

If a destination directory is specified, the archive `_must_` be given
 with an absolute path.

SEE ALSO

1.53 bbsread.library/UniqueMsgFile()

NAME

UniqueMsgFile -- Create a unique message file (for events)

SYNOPSIS

```
filename = UniqueMsgFile( bbs, filepart, tagitems )
```

```
D0                A0      A1      A2
```

```
STRPTR UniqueMsgFile( struct BBSListItem *, STRPTR *,  
    struct TagItem * );
```

```
filename = UniqueMsgFileTags(bbs, filepart, Tag1, ... )
```

```
STRPTR UniqueMsgFileTags( struct BBSListItem *, STRPTR *,  
    ULONG, ... );
```

FUNCTION

Creates a unique message file to be used with `BREV_MsgFile` and
`BREV_DetailedFileDescr` tags. The file is created in the directory for
the given bbs. If the type of the bbs has `td_InitMsgFile` set, the
command will be used to initialize the msg file.

INPUTS

bbs - Pointer to `BBSListItem` for bbs.

filepart - Pointer to `STRPTR` where to put the pointer to the file
part of the filename and path returned. The pointer returned in
*filepart must be used in the `BREV_MsgFile` and
`BREV_DetailedFileDescr` tags.

tagitems - Pointer to `TagItem` array.

Here are the `TagItem.ti_Tag` values that are defined for
`UniqueMsgFile()`:

UMF_Extension - Extension to use for message file is pointed by (STRPTR) ti_Data. The default extension is "msg".

UMF_UseTag - What message tag the file should be used for is in (ULONG) ti_Data. The default value is BREV_MsgFile.

UMF_Event - What type of event this message should be used for is in (ULONG) ti_Data. The default event type is EVE_ENTERMSG.

RESULT

filename - Complete path to the created file. Returns NULL on failure. The filename *must* be freed with a call to

```
FreeBRObject()
```

.

EXAMPLE

NOTES

If you for some reason don't use the returned filename in any event tag, you should delete the file before you call

```
FreeBRObject()
```

.

BUGS

SEE ALSO

1.54 bbsread.library/UpdateBREvent()

NAME

UpdateBREvent -- Updates the flags of one event.

SYNOPSIS

```
success = UpdateBREvent( bbs, eventnr, tagitems )
D0                A0      D1      A1
```

```
BOOL UpdateBREvent( struct BBSListItem *, ULONG, struct TagItem * );
```

```
success = UpdateBREventTags( bbs, eventnr, Tag1, ... )
```

```
BOOL UpdateBREventTags( struct BBSListItem *, ULONG, ULONG, ... );
```

FUNCTION

Lets you update the flags of an event. This function will set the BDF_EVENTS_CHANGED flag for this BBS. The priority of the flags are: (highest to lowest) EDF_DELETED, EDF_DONE, EDF_ERROR, EDF_FREEZE, EDF_PACKED.

INPUTS

bbs - BBS to update event on.

eventnr - Eventnr to update.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for UpdateBREvent().

UBRE_SetDeleted - Set EDF_DELETED flag if (BOOL) ti_Data is TRUE. Will clear EDF_PACKED and EDF_ERROR flags.

UBRE_ClearDeleted - Clear EDF_DELETED flag if (BOOL) ti_Data is TRUE.

UBRE_SetPacked - Set EDF_PACKED flag if (BOOL) ti_Data is TRUE. This flag is for use against BBS'es where evenets are sent in packages. Set this flag when the event is packed into the package. Then use this flag to delete the events done when the package is successfully sent. This makes it possible to repack a package easily without loosing any events.

UBRE_ClearPacked - Clear EDF_PACKED flag if (BOOL) ti_Data is TRUE.

UBRE_SetError - Set EDF_ERROR flag if (BOOL) ti_Data is TRUE. Set this flag if the event couldn't be executed. While this flag is set, event packers should ignore this event. Will clear the EDF_PACKED flag.

UBRE_ClearError - Clear EDF_ERROR flag if (BOOL) ti_Data is TRUE.

UBRE_SetDone - Set EDF_DONE flag if (BOOL) ti_Data is TRUE. This flag should be set when an event is successfully executed. Events marked with this flag will be removed from the datafile when packing the eventdata.

UBRE_ClearDone - Clear EDF_DONE flag if (BOOL) ti_Data is TRUE.

UBRE_SetFreeze - Set EDF_FREEZE flag if (BOOL) ti_Data is TRUE. When his flag is set, event pakkers should ignore this event. Will clear the EDF_PACKED flag.

UBRE_ClearFreeze - Clear EDF_FREEZE flag if (BOOL) ti_Data is TRUE.

UBRE_Activate - Activate event if (BOOL) ti_Data is TRUE. Will clear EDF_FREEZE, EDF_DONE, EDF_ERROR, EDF_PACKED and EDF_DELETED flags.

RESULT
success - Boolean.

EXAMPLE

NOTES
bbs->bl_Data->bd_Flags and bbs->bl_Data->bd_NumEvents will be updated by UpdateBREvent().

BUGS

SEE ALSO

WriteBREvent()

1.55 bbsread.library/UpdateBRMessage()

NAME

UpdateBRMessage -- Updates the flags of one message.

SYNOPSIS

```
success = UpdateBRMessage( conf, msgnr, tagitems )
D0                A0    D1        A1
```

```
BOOL UpdateBRMessage( struct ConflListItem *, ULONG, struct TagItem * );
```

```
success = UpdateBRMessageTags( conf, msgnr, Tag1, ... )
```

```
BOOL UpdateBRMessageTags( struct ConflListItem *, ULONG, ULONG, ... );
```

FUNCTION

Lets you update the flags of a message. Starting from V6 of the library, this function can also be used to update the headers and text of a message.

INPUTS

conf - Pointer to conference where to update message.
 msgnr - Number of the message to .
 tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for UpdateBRMessage().

UBRM_SetDelete - Set MDF_DELETED flag if (BOOL) ti_Data is TRUE.
 Will also unmark the message if it is marked.

UBRM_ClearDelete - Clear MDF_DELETED flag if (BOOL) ti_Data is TRUE.
 Will fail if MDF_UNRECOVERABLE flag is set.

UBRM_SetKeep - Set MDF_KEEP flag if (BOOL) ti_Data is TRUE.

UBRM_ClearKeep - Clear MDF_KEEP flag if (BOOL) ti_Data is TRUE.

UBRM_SetReplied - Set MDF_REPLIED flag if (BOOL) ti_Data is TRUE.

UBRM_ClearReplied - Clear MDF_REPLIED flag if (BOOL) ti_Data is TRUE.

UBRM_SetUrgent - Set MDF_URGENT flag if (BOOL) ti_Data is TRUE.

UBRM_ClearUrgent - Clear MDF_URGENT flag if (BOOL) ti_Data is TRUE.

UBRM_SetImportant - Set MDF_IMPORTANT flag if (BOOL) ti_Data is TRUE.

UBRM_ClearImportant - Clear MDF_IMPORTANT flag if (BOOL) ti_Data is TRUE.

UBRM_ClearMarked - Clear MDF_MARKED flag (make msg read) if (BOOL) ti_Data is TRUE.

URBM_SetMarked - Set MDF_MARKED flag (make msg unread) if (BOOL) ti_Data is TRUE.

URBM_SetSuperUnread - Set MDF_SUPERMARKED flag if (BOOL) ti_Data is TRUE.

URBM_ClearSuperUnread - Clear MDF_SUPERMARKED flag if (BOOL) ti_Data is TRUE.

URBM_SetHazeLevel - Haze Level to set for message is in (ULONG) ti_Data. Possible haze levels are 0 to 3.

- 0 - No hazing.
- 1 - The message will be kept at least as long as GlobalConfig->gc_HazeLevel1 seconds.
- 2 - The message will be kept at least as long as GlobalConfig->gc_HazeLevel2 seconds.
- 3 - The message will be kept at least as long as GlobalConfig->gc_HazeLevel3 seconds.

URBM_SetConfidential - Set MDF_CONFIDENTIAL flag if (BOOL) ti_Data is TRUE.

URBM_ClearConfidential - Clear MDF_CONFIDENTIAL flag if (BOOL) ti_Data is TRUE.

URBM_SetQueued - Set MDF_QUEUED flag if (BOOL) ti_Data is TRUE. (V6)

URBM_ClearQueued - Clear MDF_QUEUED flag if (BOOL) ti_Data is TRUE. (V6)

URBM_UpdateHeaders - Update header tags if (BOOL) ti_Data is TRUE. Note that all header tags of the message must be replaced, it will not work to replace just a few. (V6)

URBM_UpdateText - Update text tags if (BOOL) ti_Data is TRUE. Note that all text tags of the message must be replaced, it will not work to replace just a few. (V6)

If URBM_UpdateHeaders is TRUE, the following header tags are accepted:

BRMSG_FromName - The name of the user who this message is written by is pointed by (STRPTR) ti_Data. This is only the 'real name' of the author. This tag must be supplied.

BRMSG_FromAddr - The address of the user this message is written by is pointed by (STRPTR) ti_Data. This is the author's net-address. The name needn't redundantly be repeated in this field, if it's already in BRMSG_FromName. Only use this if the message is in a network conference.

BRMSG_ToName - The name of the user who this message is to is pointed by (STRPTR) ti_Data. This is only the 'real name'. When adding messages with no receiver (to all), use a NULL pointer in ti_Data or omit this tag.

BRMSG_ToAddr - The address of the user this message is to is pointed by

(STRPTR) `ti_Data`. This is the addressed person's net-address. The name needn't redundantly be repeated in this field, if it's already in `BRMSG_ToName`. Only use this if the message is in a network conference. This tag can only be used when you use the `BRMSG_ToName` tag.

`BRMSG_MsgID` - The message ID string is pointed by (STRPTR) `ti_Data`. Use this tag when the message has a string as identifier. This id is considered to be unique.

`BRMSG_OriginalNr` - The number this message has on the BBS is in (ULONG) `ti_Data`. Use this tag when the message has a number as identifier.

`BRMSG_RefID` - If this message is an answer (STRPTR) `ti_Data` points to the ID string for the message this message refer to.

`BRMSG_RefNr` - If this message is an answer (ULONG) `ti_Data` contains the messagenumber this message refer to. The messagenumber is the number the refered message has on the BBS.

`BRMSG_MsgSize` - The size of the message on the remote system is in (ULONG) `ti_Data`. Use this tag when the message has a known size in bytes.

`BRMSG_SpecialID` - A special ID for use by the mail/news parser can be kept in this (STRPTR) `ti_Data`. For instance, it can be used to keep POP server information to be able to retrieve a particular message at a later stage.

`BRMSG_ErrorString` - An error description is in (STRPTR) `ti_Data`. This is for use by the parser to notify the host program of errors during retrieval of a message. (V6)

`BRMSG_CreationDate` - The time the message was created is in (ULONG) `ti_Data`. The time is in seconds since 1.January 1978.

`BRMSG_CreationDateTxt` - The time the message was created is in (STRPTR) `ti_Data`. The formatting of the string is free. Use this when it is impossible to use `BRMSG_CreationDate`

`BRMSG_Subject` - The subject of the message is pointed by (STRPTR) `ti_Data`. This tag must be supplied.

If `URBM_UpdateText` is TRUE, the following text tags are accepted:

`BRMSG_ReplyConf` - The name of the conference a reply to this message should be put is pointed by (STRPTR) `ti_Data`. If the message is private, it should only be moved if it is possible to have private messages in this conference.

`BRMSG_ReplyName` - Name of the user a reply of this message should go to is in (STRPTR) `ti_Data`. Same format as `BRMSG_ToName`.

`BRMSG_ReplyAddr` - The address of the user a reply of this message should go to is in (STRPTR) `ti_Data`. Same format as `BRMSG_ToAddr`.

BRMSG_Comment - Comments concerning this message is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed. Header information not used in another tag could be put here.

BRMSG_Text - The text in this message is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed. This tag must be supplied, but can point to a 0 line STRPTR array. Any information that do not belong to the original text must not be put here. The text may be of any size. Lines may be of any length. It is up to the reader program to wrap lines if needed.

BRMSG_BinaryPart - File name for file containing the binary part is pointed by (STRPTR) ti_Data. Paths should be relative to bbs->bl_BBSPath, but absolute paths are allowed. This tag can be used more than once in a message.

BRMSG_BinaryPartDesc - A description of the binary part is pointed by (STRPTR) ti_Data. The N'th use of this tag describes the n'th use of the BRMSG_BinaryPart tag. This tag must be used each time the BRMSG_BinaryPart tag is used.

BRMSG_BinaryPartComment - Comments concerning a binary part is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed. The N'th use of this tag comments the n'th use of the BRMSG_BinaryPart tag. This tag must be used each time the BRMSG_BinaryPart tag is used. It's legal for the comment to contain 0 lines.

BRMSG_TextPart - If the message contains more than one text part, the parts beyond the first part should use this tag. The text is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed. This tag can be used more than once in a message.

BRMSG_TextPartComment - Comments concerning a text part is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed. The N'th use of this tag comments the n'th use of the BRMSG_TextPart tag. This tag must be used each time the BRMSG_TextPart tag is used. It's legal for the comment to contain 0 lines.

BRMSG_TextPartUniqueID - A unique numerical ID for the textpart should be provided with this (ULONG) tag. This ID can be used by the application to find a certain textpart, and must be unique within the same message. Eg. the first textpart might get an ID of 1, the second 2, etc.

BRMSG_TextPartFilename - If the textpart consists of a textfile or somesuch where a filename is expected to be used when saving the part, the suggested filename is provided as (STRPTR) with this tag.

BRMSG_MsgPart - A tag array for a message part is pointed by (TagItem *) ti_Data. The tagarray can contain *all* message tags, and *must*

contain at least one BRMSG_#? tag. This tag can be used more than once in a message.

Other accepted tags when updating headers and/or text:

WBRMSG_Private - Message is flagged as private if (BOOL) ti_Data is TRUE. Default is FALSE.

WBRMSG_Read - Message is flagged as read by reciever if (BOOL) ti_Data is TRUE. Default is FALSE.

WBRMSG_Urgent - Message is flagged as urgent if (BOOL) ti_Data is TRUE. Default is FALSE.

WBRMSG_Important - Message is flagged as important if (BOOL) ti_Data is TRUE. Default is FALSE.

WBRMSG_Confidential - Message is flagged as confidential if (BOOL) ti_Data is TRUE. Default is FALSE.

WBRMSG_ToFromUserStatus - The status of the MDF_TO_USER and MDF_FROM_USER flags this message should have is in (LONGBITS) ti_Data. When this tag is used, the sender or receiver will *not* be matched against the name and/or address of the user. When using this tag, the parser is responsible for checking if the message is to or from the user.

RESULT

success - Boolean.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.56 bbsread.library/UpdateDataStruct()

NAME

UpdateDataStruct -- Updates data structures.

SYNOPSIS

```
fail = UpdateDataStruct( tagitems )
D0                                     A0
```

```
struct TagItem * UpdateDataStruct( struct TagItem * );
```

```
fail = UpdateDataStructTags( Tag1, ... )
```

```
struct TagItem * UpdateDataStructTags( ULONG, ... );
```

FUNCTION

Function for updating data structures. This method involves less

overhead than freeing an getting a new data structure. If you don't use the UD_RemoveDeleted tag, you will be guaranteed that all entries in the lists are still there. The entries in a list may have changed order, and there may have been inserted new nodes.

INPUTS

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for UpdateDataStruct().

UD_ConfList - Conference list to update datastructures in is pointed by (List *) ti_Data.

UD_ConfItem - ConfListItem structure to update is in (struct ConfListItem *) ti_Data.

UD_BBSList - BBS list to update datastructures in is pointed by (List *) ti_Data.

UD_BBSItem - BBSListItem structure to update is in (struct BBSListItem *) ti_Data.

UD_FAreaList - File area list to update datastructures in is pointed by (List *) ti_Data.

UD_FAreaItem - FAreaListItem structure to update is in (struct FAreaListItem *) ti_Data.

UD_RemoveDeleted - Remove deleted entries in lists if (BOOL) ti_Data is TRUE. Default is FALSE.

RESULT

fail - NULL on success. On failure, it points to the tag which caused the failure.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.57 bbsread.library/WriteBREvent()

NAME

WriteBREvent -- Adds a event to a bbs.

SYNOPSIS

```
eventnr = WriteBREvent( bbs, event, tagitems )
```

```
D0                A0    D1    A1
```

```
ULONG WriteBREvent( struct BBSListItem *, ULONG, struct TagItem * );
```

```
eventnr = WriteBREventTags( bbs, event, Tag1, ... )
```

```
ULONG WriteBREventTags( struct BBSListItem *, ULONG, ULONG, ... );
```

FUNCTION

This function adds a event to the list of events to be done on the next call to the BBS.

Available eventtypes and what tags that are allowed to use with the different event types are defined in the BBSType for this BBS. Be sure to pass all the NeedTags with appropriate values. This function will set the BDF_EVENTS_CHANGED flag for this BBS.

INPUTS

bbs - Pointer to the BBSListItem for the bbs to add the event to.
 event - What type of event to add. For a list of types se
 <libraries/bbsread.h>
 tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for WriteBREvent().

BREV_ToName - Name to send to is pointed by (STRPTR) ti_Data.
 BREV_ToAddr - Address to send to is pointed by (STRPTR) ti_Data.
 BREV_Subject - Subject of message is pointed by (STRPTR) ti_Data.
 Must not be longer than TypeData->td_SubjectLength.

BREV_Conference - Name of conference is pointed by (STRPTR) ti_Data.
 BREV_RefNr - The number of the message to reply to is in (ULONG)
 ti_Data. The message number is the number this message has in the
 local database.

BREV_RefOriginalNr - Number of the message on BBS to reply to is in
 (ULONG) ti_Data.

BREV_RefId - The idstring of the message to reply to is pointed by
 (STRPTR) ti_Data.

BREV_MsgFile - Name of file with text is pointed by (STRPTR) ti_Data.
 Filename is relative to bbs->bl_BBSPath. Each line in this file
 should not be longer than TypeData->td_LineLength.

BREV_Private - Message should be flagged as private if (BOOL) ti_Data
 is TRUE. Default value is FALSE.

BREV_LocalFile - Path and name of local file is in (STRPTR) ti_Data.
 BREV_Directory - Directory for file to up/down load is in (STRPTR)
 ti_Data. This is the remote directory.

BREV_FileName - Name of file to upload/download is pointed by (STRPTR)
 ti_Data. This is the remote filename. It must not be longer than
 TypeData->td_FileNameLen.

BREV_DownloadNotify - Notify on download if (BOOL) ti_Data is TRUE.
 Default value is FALSE.

BREV_FileDescr - File description is pointed by (STRPTR) ti_Data. Must

not be longer than `TypeData->td_FileDescrLen`.

`BREV_DetailedFileDescr` - Name of file with detailed file description is pointed by `(STRPTR) ti_Data`. Filename is relative to `bbs->bl_BBSPath`. Each line in this file should not be longer than `TypeData->td_LineLength`.

`BREV_FromMessageNr` - Message number to start at is in `(ULONG) ti_Data`.

`BREV_ToMessageNr` - Message number to end at is in `(ULONG) ti_Data`.

`BREV_CommandString` - Command string is in `(STRPTR) ti_Data`.

`BREV_Boolean` - Boolean value is in `(BOOL) ti_Data`.

`BREV_Date` - Time in seconds since 1.January 1978 is in `(ULONG) ti_Data`.

`BREV_PGPSignID` - Id for key to sign with is pointed by `(STRPTR) ti_Data`. It's preferable that the keyid is used to identify a key. (e.g. `0x5B4231FD`) Using a '*' will sign with the first key in the secret keyring.

`BREV_PGPEncryptID` - Id for key(s) to encrypt with is pointed by `(STRPTR) ti_Data`. It's preferable that the keyid is used to identify a key. (e.g. `0x5B4231FD`) If more than one key are used to encrypt, the id for each key should be separated with a space.

`BREV_RefConference` - Name of the conference of the message to reply to is in `(STRPTR) ti_Data`. This tag is to be used when replying messages to other conferences than the conference of the original message. Should only be used when the bbs type has the `TDF_GLOBAL_REPLIES` flag set.

`BREV_Urgent` - Flag event as urgent if `(BOOL) ti_Data` is TRUE. Default value is FALSE.

`BREV_Important` - Flag event as important if `(BOOL) ti_Data` is TRUE. Default value is FALSE.

`BREV_Confidential` - Flag event as confidential if `(BOOL) ti_Data` is TRUE. Default value is FALSE.

`BREV_ReturnReceipt` - Set return receipt flag for event if `(BOOL) ti_Data` is TRUE. Default value is FALSE.

`BREV_Encode8bit` - Encode outgoing text if it is 8 bit and `(BOOL) ti_Data` is TRUE. Default value is FALSE.

`WBREV_UpdateEventNr` - Update event with event number `(ULONG) ti_Data`. All old tags for this event will be discarded. Will clear the flags set for this event.

RESULT

`eventnr` - The number the event got in the database. Returns 0 on failure.

EXAMPLE

NOTES

All strings is considered to be in ISO charset.

bbs->bl_Data->bd_LastEvent, bbs->bl_Data->bd_NumEvents and bbs->bl_Data->bd_Flags will be updated by WriteBREvent().

BUGS

SEE ALSO

1.58 bbsread.library/WriteBRFile()

NAME

WriteBRFile -- Write an entry to the file database.

SYNOPSIS

```
filenr = WriteBRFile( farea, tagitems )
D0                A0        A1
```

```
ULONG WriteBRFile( struct FAreaListItem *, struct TagItem * );
```

```
filenr = WriteBRFileTags( farea, Tag1, ... )
```

```
ULONG WriteBRFileTags( struct FAreaListItem *, ULONG, ... );
```

FUNCTION

Writes an entry to a file area in the file database for a bbs.

INPUTS

farea - Pointer to FAreaListItem for file area to write file to.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for WriteBRFile().

BRFILE_Name - The name the file is identified with on the bbs is pointed by (STRPTR) ti_Data. This tag must be supplied when adding or updating file tags.

BRFILE_Date - The date the file has on the bbs is in (ULONG) ti_Data. The time is in seconds since 1.January 1978.

BRFILE_Size - The size of the file is in (ULONG) ti_Data.

BRFILE_Description - A (short) description of the file is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed.

BRFILE_Downloads - Number of times the file has been downloaded from the bbs is in (ULONG) ti_Data.

WBRF_UpdateFileNr - Update the file with filename (ULONG) ti_Data. All old tags for this file will be discarded.

WBRF_DeleteFile - Mark the file with file number supplied in the WBRF_UpdateFileNr tag as deleted if (BOOL) ti_Data is TRUE.

This tag is ignored when the WBRF_UpdateFileNr tag is not supplied.

RESULT

filenr - The number the file got in the database. Returns 0 on failure.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.59 bbsread.library/WriteBRIEFMsg()

NAME

WriteBRIEFMsg -- Write a message in BRIEF.

SYNOPSIS

```
error = WriteBRIEFMsg( fileid, conf, msgnr )
D0                A0      A1      D0
```

```
BOOL WriteBRIEFMsg( APTR, struct ConfListItem *, ULONG );
```

FUNCTION

Writes a message in BRIEF to a file opened by
BufBROpen()

INPUTS

fileid - APTR to BBSRead file handler.
conf - Pointer to ConfListItem for conference.
msgnr - Message number to write.

RESULT

error - Boolean.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.60 bbsread.library/WriteBRKill()

NAME

WriteBRKill -- Write an entry to the kill database.

SYNOPSIS

```
killnr = WriteBRKill( bbs, tagitems )
D0                A0        A1
```

```
ULONG WriteBRKill( struct BBSListItem *, struct TagItem * );
```

```
killnr = WriteBRKillTags( bbs, Tag1, ... )
```

```
ULONG WriteBRKillTags( struct BBSListItem *, ULONG, ... );
```

FUNCTION

Writes an entry to the kill database for aa bbs. This kill database is used during message adding. Killed messages will be added to the database, but will not be marked as unread unless WBRK_MarkDeleted is used, in which case the message is NOT added to the database. A message will be killed if the message data matches *one* of the kills for a bbs. Kills can also be used to set message flags, see tags below.

INPUTS

bbs - Pointer to BBSListItem for bbs to writekill to.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for WriteBRKill().

BRMSG_#? - All normal message tags can be used in a kill. Standard AmigaDos wildcards can be used in all string tags. In string array, tags all given lines must match one line in the corresponding tag in the message. At least one message tag must be supplied to add a kill.

BRKILL_Conference - Which conference to kill in is pointed by (STRPTR) ti_Data. Standard AmigaDos wildcards can be used. Default value for this tag is '#?'.

BRKILL_Private - The kill will only match private messages if (BOOL) ti_Data is TRUE. If (BOOL) ti_Data is FALSE, the kill will only match non-private messages. The default is not to regard the private flag.

BRKILL_Description - A description the user can use for this kill is pointed by (STRPTR) ti_Data. New for V5 of bbsread.library.

WBRK_UpdateKillNr - Update the kill will kill number (ULONG) ti_Data. All old tags and flags for this kill will be discarded.

WBRK_DeleteKill - Mark the kill with the kill number supplied in the WBRK_UpdateKillNr tag as deleted if (BOOL) ti_Data is TRUE.

WBRK_MarkKeep - Mark messages matching the kill with MDF_KEEP if (BOOL) ti_Data is TRUE. The message will be marked as unread.

WBRK_MarkUrgent - Mark messages matching the kill with MDF_URGENT if (BOOL) ti_Data is TRUE. The message will be marked as unread.

WBRK_MarkImportant - Mark messages matching the kill with MDF_IMPORTANT if (BOOL) ti_Data is TRUE. The message will be marked as unread.

WBRK_MarkDeleted - Mark messages matching the kill with MDF_DELETED if (BOOL) ti_Data is TRUE. The message will not be written do the database in this case. (4.66+)

WBRK_MarkHazeLevel - Mark messages matching the kill with the haze level according to (ULONG) ti_Data. Possible haze levels are 0 to 3. The message will be marked as unread.

- 0 - No hazing.
- 1 - The message will be kept at least as long as GlobalConfig->gc_HazeLevel1 seconds.
- 2 - The message will be kept at least as long as GlobalConfig->gc_HazeLevel2 seconds.
- 3 - The message will be kept at least as long as GlobalConfig->gc_HazeLevel3 seconds.

RESULT

killnr - The number the kill got in the database. Returns 0 on failure.

EXAMPLE

NOTES

WBRK_MarkDeleted is new for 4.66 and higher
BRKILL_Description is new for V5.

BUGS

SEE ALSO

1.61 bbsread.library/WriteBRMessage()

NAME

WriteBRMessage -- Writes a message to a conference.

SYNOPSIS

```
msgnr = WriteBRMessage( conf, tagitems )
```

```
D0                A0        A1
```

```
ULONG WriteBRMessage( struct ConflistItem *, struct TagItem * );
```

```
msgnr = WriteBRMessageTags( conf, Tag1, ... )
```

```
ULONG WriteBRMessageTags( struct ConflistItem *, ULONG, ... );
```

FUNCTION

This funtions Writes a message to a conference.

The multiple parts in multipart messages should be put in the taglist in the order they should be presented. BRMSG_Text should always contain the first text part of the message.

INPUTS

conf - Pointer to the ConfListItem for the conference to Write the message to.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for WriteBRMessage().

BRMSG_FromName - The name of the user who this message is written by is pointed by (STRPTR) ti_Data. This is only the 'real name' of the author. This tag must be supplied.

BRMSG_FromAddr - The address of the user this message is written by is pointed by (STRPTR) ti_Data. This is the author's net-address. The name needn't redundantly be repeated in this field, if it's already in BRMSG_FromName. Only use this if the message is in a network conference.

BRMSG_ToName - The name of the user who this message is to is pointed by (STRPTR) ti_Data. This is only the 'real name'. When adding messages with no receiver (to all), use a NULL pointer in ti_Data or omit this tag.

BRMSG_ToAddr - The address of the user this message is to is pointed by (STRPTR) ti_Data. This is the addressed person's net-address. The name needn't redundantly be repeated in this field, if it's already in BRMSG_ToName. Only use this if the message is in a network conference. This tag can only be used when you use the BRMSG_ToName tag.

BRMSG_MsgID - The message ID string is pointed by (STRPTR) ti_Data. Use this tag when the message has a string as identifier. This id is considered to be unique.

BRMSG_OriginalNr - The number this message has on the BBS is in (ULONG) ti_Data. Use this tag when the message has a number as identifier.

BRMSG_RefID - If this message is an answer (STRPTR) ti_Data points to the ID string for the message this message refer to.

BRMSG_RefNr - If this message is an answer (ULONG) ti_Data contains the messagenumber this message refer to. The messagenumber is the number the referred message has on the BBS.

BRMSG_MsgSize - The size of the message on the remote system is in (ULONG) ti_Data. Use this tag when the message has a known size in bytes. (V6)

BRMSG_SpecialID - A special ID for use by the mail/news parser can be kept in this (STRPTR) ti_Data. For instance, it can be used to keep POP server information to be able to retrieve a particular message at a later stage. (V6)

BRMSG_ErrorString - An error description is in (STRPTR) ti_Data. This is for use by the parser to notify the host program of errors

during retrieval of a message. (V6)

BRMSG_CreationDate - The time the message was created is in (ULONG) ti_Data. The time is in seconds since 1.January 1978.

BRMSG_CreationDateTxt - The time the message was created is in (STRPTR) ti_Data. The formatting of the string is free. Use this when it is impossible to use BRMSG_CreationDate

BRMSG_Subject - The subject of the message is pointed by (STRPTR) ti_Data. This tag must be supplied.

BRMSG_ReplyConf - The name of the conference a reply to this message should be put is pointed by (STRPTR) ti_Data. If the message is private, it should only be moved if it is possible to have private messages in this conference.

BRMSG_ReplyName - Name of the user a reply of this message should go to is in (STRPTR) ti_Data. Same format as BRMSG_ToName.

BRMSG_ReplyAddr - The address of the user a reply of this message should go to is in (STRPTR) ti_Data. Same format as BRMSG_ToAddr.

BRMSG_Comment - Comments concerning this message is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed. Header information not used in another tag could be put here.

BRMSG_Text - The text in this message is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed. This tag must be supplied, but can point to a 0 line STRPTR array, even if adding only header information in V6 of bbsread. Any information that do not belong to the original text must not be put here. The text may be of any size. Lines may be of any length. It is up to the reader program to wrap lines if needed.

BRMSG_BinaryPart - File name for file containing the binary part is pointed by (STRPTR) ti_Data. Paths should be relative to bbs->bl_BBSPath, but absolute paths are allowed. This tag can be used more than once in a message.

BRMSG_BinaryPartDesc - A description of the binary part is pointed by (STRPTR) ti_Data. The N'th use of this tag describes the n'th use of the BRMSG_BinaryPart tag. This tag must be used each time the BRMSG_BinaryPart tag is used.

BRMSG_BinaryPartComment - Comments concerning a binary part is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed. The N'th use of this tag comments the n'th use of the BRMSG_BinaryPart tag. This tag must be used each time the BRMSG_BinaryPart tag is used. It's legal for the comment to contain 0 lines.

BRMSG_TextPart - If the message contains more than one text part, the parts beyond the first part should use this tag. The text is

in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed. This tag can be used more than once in a message.

BRMSG_TextPartComment - Comments concerning a text part is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed. The N'th use of this tag comments the n'th use of the BRMSG_TextPart tag. This tag must be used each time the BRMSG_TextPart tag is used. It's legal for the comment to contain 0 lines.

BRMSG_TextPartUniqueID - A unique numerical ID for the textpart should be provided with this (ULONG) tag. This ID can be used by the application to find a certain textpart, and must be unique within the same message. Eg. the first textpart might get an ID of 1, the second 2, etc.

BRMSG_TextPartFilename - If the textpart consists of a textfile or somesuch where a filename is expected to be used when saving the part, the suggested filename is provided as (STRPTR) with this tag.

BRMSG_MsgPart - A tag array for a message part is pointed by (TagItem *) ti_Data. The tagarray can contain *all* message tags, and *must* contain at least one BRMSG_#? tag. This tag can be used more than once in a message.

WBRMSG_MarkMessage - Message will be marked if (BOOL) ti_Data is TRUE. NB: Default is TRUE. Messages written by bd_UserName/gc_GlobalUserName which not are to bd_UserName and messages which matches a normal kill will not be marked.

WBRMSG_Private - Message is flagged as private if (BOOL) ti_Data is TRUE. Default is FALSE.

WBRMSG_Read - Message is flagged as read by reciever if (BOOL) ti_Data is TRUE. Default is FALSE.

WBRMSG_Urgent - Message is flagged as urgent if (BOOL) ti_Data is TRUE. Default is FALSE.

WBRMSG_Important - Message is flagged as important if (BOOL) ti_Data is TRUE. Default is FALSE.

WBRMSG_Confidential - Message is flagged as confidential if (BOOL) ti_Data is TRUE. Default is FALSE.

WBRMSG_ToFromUserStatus - The status of the MDF_TO_USER and MDF_FROM_USER flags this message should have is in (LONGBITS) ti_Data. When this tag is used, the sender or receiver will *not* be matched against the name and/or address of the user. When using this tag, the parser is responsible for checking if the message is to or from the user.

WBRMSG_Queued - Message is flagged as queued for later download if (BOOL) ti_Data is TRUE. Default is FALSE. (V6)

WBMSG_EmptyBody - Message is flagged as having no message body if (BOOL) ti_Data is TRUE. Default is FALSE. Make sure to use this tag when adding only headers. This flag will be cleared automatically when adding a body using

```
UpdateBRMessage()
. (V6)
```

RESULT

msgnr - The number the message got in the database. Returns 0 on failure and -1 if the message matched a kill that would delete the message, in which case the message will not be added to the database.

EXAMPLE

Some examples for splitting addresses in 'Name' and 'Address':

RFC:

```
"Martin Horneffer <maho@balrog.dfv.rwth-aachen.de>"
-> name: "Martin Horneffer"
    address: "maho@balrog.dfv.rwth-aachen.de"
```

```
"horneff@pool.informatik.rwth-aachen.de (Martin Horneffer)"
-> name: "Martin Horneffer"
    address: "horneff@pool.informatik.rwth-aachen.de"
```

```
"horneff@pool.informatik.rwth-aachen.de"
-> name: "horneff"
    address: "horneff@pool.informatik.rwth-aachen.de"
```

FidoNet:

```
"Martin Horneffer at 2:242/7.9"
-> name: "Martin Horneffer"
    address: "2:242/7.9"
```

```
"Joerg Gutzke at 2:242/7"
-> name: "Joerg Gutzke"
    address: "2:242/7"
```

NOTES

All strings is considered to be in the BRCS_ISO charset. To get the charset defined for this Conf/BBS/BBSType use

```
ConfCharset()
.
```

-1 is a new return code added for 4.65 and higher of bbsread.library

BRMSG_Text must contain atleast an empty array, even if you are adding only header information.

BUGS

SEE ALSO

1.62 bbsread.library/WriteBRUser()

NAME

WriteBRUser -- Writes a user to a bbs.

SYNOPSIS

```
usernr = WriteBRUser( bbs, tagitems )
D0                A0      A1
```

```
ULONG WriteBRUser( struct BBSListItem *, struct TagItem * );
```

```
usernr = WriteBRUserTags( bbs, Tag1, ... )
```

```
ULONG WriteBRUserTags( struct BBSListItem *, ULONG, ... );
```

FUNCTION

Writes a user to the userdatabase for a bbs.

INPUTS

bbs - Pointer to BBSListItem for bbs to write user to.

tagitems - Pointer to TagItem array.

Here are the TagItem.ti_Tag values that are defined for WriteBRUser().

BRUSR_Name - The name the user has on this bbs is pointed by (STRPTR) ti_Data. This tag must be supplied when adding or updating a user.

BRUSR_Address - The address the user has on this bbs is pointed by (STRPTR) ti_Data.

BRUSR_Alias - Alias to use for referring to this user is pointed by (STRPTR) ti_Data. This alias can be used to look up this user in the database.

BRUSR_Comment - A comment that can be attached to this user is in a NULL terminated STRPTR array pointed by (STRPTR *) ti_Data. Each STRPTR represents a line of text. No newline characters are allowed.

BRUSR_Encode8BitMsg - Encode 8 bit private messages to this user if (BOOL) ti_Data is TRUE. Default is FALSE. Only applicable when the TDF_SUPPORTS_ENCODE_8BIT_MAIL flag is set for the bbstype of this bbs.

BRUSR_PGPkeyID - The PGP key ID for this users PGP key is pointed by (STRPTR) ti_Data.

WBRUSR_UpdateUserNr - Update the user with user number (ULONG) ti_Data. All old tags for this user will be discarded.

WBRUSR_DeleteUser - Mark the user with the usernumber supplied in the WBRUSR_UpdateUserNr tag as deleted if (BOOL) ti_Data is TRUE. This tag is ignored when the WBRUSR_UpdateUserNr tag is not supplied.

WBRUSR_OnlyIfNotExist - If (BOOL) ti_Data is TRUE, the user will only be written to the database if the name given in BRUSR_Name don't exist in the database. Default is FALSE. This tag is ignored when updating users with WBRUSR_UpdateUserNr.

RESULT

usernrr - The number the user got in the database. Returns 0 on failure.

EXAMPLE

NOTES

BUGS

SEE ALSO

1.63 bbsread.library/WritePassiveConfList()

NAME

WritePassiveConfList -- Write to the passive conference list.

SYNOPSIS

```
error = WritePassiveConfList( bbs, passConfList )
D0                                A0      A1
```

```
BOOL WritePassiveConfList( struct BBSListItem *, struct List * );
```

FUNCTION

Writes a list of conference names to the passive conference list datafiles. The old list is deleted before writing this new one. The passive conference list is a list of *all* available conferences at the bbs. It should be used when the user want to send a join event.

The list should only be written when the parser has got a list over all conferences at the bbs. If you don't have the internal number for the conference, pl_BBSSconfNr must be initalized to -1.

INPUTS

bbs - Pointer to the bbs which the list should be written to.
passConfList - Pointer to a list of struct PasssConfListItem.

RESULT

error - Boolean.

EXAMPLE

NOTES

BUGS

SEE ALSO