SuperPlay_Ref_ENG

COLLABORATORS

	TITLE : SuperPlay_Ref_ENG	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

SuperPlay_Ref_ENG

1.1 SuperPlay_Ref_ENG.doc

```
--background--
SPL_AllocHandle()
SPL_FreeHandle()
SPL_StopReplay()
SPL_FreeResources()
SPL_SuperPlay()
SPL_SuperWrite()
SPL_InitHandleAsDOS()
SPL_InitHandleAsClip()
SPL_SetWriteType()
SPL_GetErrorString()
SPL_SetWriteName()
SPL_FileInfoRequest()
SPL_SetReqIOWindow()
SPL_ReadPlayData()
SPL_ContinueReplay()
SPL_FastForward()
SPL_FastBackward()
SPL_GetSampleList()
SPL_SetSampleList()
SPL_GetFileType()
```

1.2 superplay.library/--background--

```
VERSION
     $VER: SuperPlay_Ref_ENG.doc V5.1 (8.8.96)

COPYRIGHT
     © 1995-96 by Andreas R. Kleinert. All rights reserved.

     - Feel free to translate this Doc-File into other languages. -

GENERAL
           Andreas R. Kleinert,
           Sandstrasse 1,
```

D-57072 Siegen,
Germany.

EMail: Fido Andreas Kleinert 2:2457/350.18
Usenet/InterNet Andreas_Kleinert@superview.ftn.sub.org

If nothing else works, try one of these Fido-InterNet gateways:

Andreas_Kleinert@p10.f435.n2457.z2.fido.sub.org (in Germany)
Andreas_Kleinert@p10.f435.n2457.z2.fidonet.org (USA or other)

HISTORY

V2	bug-fixed ClipBoard-Support with the supplied SPObjects (ak)
V3	created working combination of SPL_SetSampleList() and SuperWrite() calls (ak)
V4	filetype recognition without loading (ak)
V4.6	Autodoc format SetWriteName: Correct type of write_name Change two SVL_ prefix to SPL_ Add version behind NAME (indy)
V5.1	SetWriteName parameter must be UBYTE * (ak)

1.3 superplay.library/SPL_AllocHandle

NAME

SPL_AllocHandle (V1)

SYNOPSIS

```
APTR SPL_AllocHandle(APTR future)
D0   -$1e            A1
```

FUNCTION

Allocates a handle for accessing a Sample/Module via SPObjects.

INPUT(S)

future - always NULL yet

RESULT

A pointer to a new allocated Handle or NULL, if allocation failed.

WARNING

Test, if the result was NULL, or not !

SEE ALSO

SPL_FreeResources, SPL_FreeHandle

1.4 superplay.library/SPL_FreeHandle

```
NAME
     SPL_FreeHandle  (V1)

SYNOPSIS

     VOID SPL_FreeHandle(APTR handle)
     D0   -$24           A1

FUNCTION

    Stops playing, frees all Resources and delocates a Handle, which has
    been allocated with SPL_AllocHandle before.

INPUT(S)

     handle - a valid handle

RESULT

     -

SEE ALSO

     SPL_AllocHandle, SPL_StopReplay, SPL_FreeResources
```

1.5 superplay.library/SPL_StopReplay

```
NAME
     SPL_StopReplay  (V1)

SYNOPSIS

     VOID SPL_StopReplay(APTR handle)
     D0   -$2a           A1

FUNCTION

     Stops playing the Sample/Module, indentified by the handle.
     Some SPObjects support to continue the Sample/Module with
     SPL_ContinueReplay after calling SPL_StopReplay.

INPUT(S)

     handle - a valid handle

RESULT

     -

SEE ALSO

     SPL_ContinueReplay, SPL_FreeResources, SPL_FreeHandle
```

1.6 superplay.library/SPL_FreeResources

```
NAME
     SPL_FreeResources (V1)

SYNOPSIS

     VOID SPL_FreeResources(APTR handle)
     D0   -$30              A1

FUNCTION

     Frees all resources belonging to the specific Sample/Module,
     indentified by the handle, which are not needed to just replay it.
     Playing of the Sample/Module is not stopped and/or interrupted.

INPUT(S)

     handle - a valid handle

RESULT

     -

SEE ALSO

     SPL_AllocHandle, SPL_StopReplay, SPL_FreeHandle
```

1.7 superplay.library/SPL_SuperPlay

```
NAME
     SPL_SuperPlay  (V1)

SYNOPSIS

     ULONG SPL_SuperPlay(APTR handle, char *filename)
     D0    -$36          A1           A2

FUNCTION

     Loads and plays the Sample/Module described by FileName.
     The handle is initialized and the fitting SPObject is opened
     and accessed for replaying the Sample/Module.

     Playing can be stopped either via full delocation of the handle
     or temporal interruption with SPL_StopReplay.

INPUT(S)

     handle   - a valid handle
     filename - a valid AmigaDOS FilePath and -Name

RESULT
```

```
     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_AllocHandle, SPL_StopReplay, SPL_ContinueReplay,
     SPL_FastForward, SPL_FastBackward, SPL_FreeHandle
```

1.8 superplay.library/SPL_SuperWrite

```
NAME
     SPL_SuperWrite  (V1)

SYNOPSIS

     ULONG SPL_SuperWrite(APTR handle, APTR source_handle)
     D0    -$3c           A1           A2

FUNCTION

     Usually a Sample/Module is loaded AND played via SPL_SuperPlay :
     No separate reading and playing calls are done.

     For writing - that means : converting - a Sample/Module, you
     have to use the following order :

     source_handle = SPL_AllocHandle(N);
     result        = SPL_ReadPlayData(source_handle, source_name);
     dest_handle   = SPL_AllocHandle(N);
     /* result     = SPL_InitHandleAsDOS(dest_handle, N); */ /* default */
     result        = SPL_SetWriteName(dest_handle, dest_name, N);
     result        = SPL_SetWriteType(dest_handle, dest_type, N);
     result        = SPL_SuperWrite(dest_handle, source_handle);
     SPL_FreeHandle(dest_handle);
     SPL_FreeHandle(source_handle);


     Also : Check the "result" value AFTER EACH function call (see
            Example SourceCodes) !

     All available values for dest_type can be found in the specific
     SPOBject-List in the SuperPlayBase.
     The values WILL change with every re-initialization of
     superplay.library, so update them with every new opening !

     If "source_handle" is NULL,
     it is checked, whether a SampleList has been set via
     "SPL_SetSampleList()", and then this SampleList is completely
     saved.

     It is suggested to use the old style for converting between
     FileFormats and the new style for saving single (self-created)
     SampleLists.

INPUT(S)
```

```
     handle        - a valid handle        (used for Write Access)
     source_handle - an other valid handle (used for Read Access)

RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO


     SPL_AllocHandle, SPL_ReadPlayData, SPL_FreeHandle
```

1.9 superplay.library/SPL_InitHandleAsDOS

```
NAME
     SPL_InitHandleAsDOS (V1)

SYNOPSIS

     ULONG SPL_InitHandleAsDOS(APTR handle, APTR future)
     D0    -$42                A1           A2

FUNCTION

     Initializes a Handle for AmigaDOS access, so that the given
     AmigaDOS FileNames are used.
     Another possibility is sometimes to initialize Handles
     for ClipBoard Access (depending on the specific SPObject,
     e.g. IFF-8SVX).

INPUT(S)

     handle - a valid handle
     future - always NULL yet


RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO


     SPL_InitHandleAsClip
```

1.10 superplay.library/SPL_InitHandleAsClip

```
NAME
     SPL_InitHandleAsClip  (V1)
```

```
SYNOPSIS

     ULONG SPL_InitHandleAsClip(APTR handle, APTR future)
     D0    -$48                 A1           A2

FUNCTION

     Initializes a Handle for ClipBoard access, so that the (possibly)
     given AmigaDOS FileNames are ignored.
     The nearly always used possibility is to initialize Handles
     for AmigaDOS Access (supported by ALL SPObjects).

INPUT(S)

     handle - a valid handle
     future - always NULL yet


RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_InitHandleAsDOS
```

1.11 superplay.library/SPL_SetWriteType

```
NAME
     SPL_SetWriteType  (V1)

SYNOPSIS

     ULONG SPL_SetWriteType(APTR handle, ULONG write_type, APTR future)
     D0    -$4e             A1           A2                A3

FUNCTION

     Sets the write_type for a SPL_SuperWrite() call.
     See description there and example SourceCodes for more and
     detailed information.

INPUT(S)

     handle     - a valid handle
     write_type - a valid temporary write_type code form the SPObject List
     future     - always NULL yet


RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO
```

```
     SPL_SuperWrite
```

1.12 superplay.library/SPL_GetErrorString

```
NAME
     SPL_GetErrorString  (V1)

SYNOPSIS

     char * SPL_GetErrorString(ULONG error_code)
     D0    -$54                A1

FUNCTION

     Returns the Error Message for a specific Error Code, returned
     by any function of the superplay.library.

INPUT(S)

     error_code - any SPERR Error Code

RESULT

     read-only pointer to a SPERR Error String

SEE ALSO

     -
```

1.13 superplay.library/SPL_SetWriteName

```
NAME
     SPL_SetWriteName  (V1)

SYNOPSIS

     ULONG SPL_SetWriteNameClip(APTR handle, UBYTE *write_name, APTR future)
     D0    -$5a                 A1           A2                 A3

FUNCTION

     Sets the write_name for a SPL_SuperWrite() call.
     See description there and example SourceCodes for more and
     detailed information.

INPUT(S)

     handle     - a valid handle
     write_name - a valid AmigaDOS FilePath and -Name description
     future     - always NULL yet
```

```
RESULT

     NULL or an adequate SPERR-Errorcode.


SEE ALSO

     SPL_SuperWrite
```

1.14 superplay.library/SPL_FileInfoRequest

```
NAME
     SPL_FileInfoRequest  (V1)

SYNOPSIS

     ULONG SPL_FileInfoRequest(APTR handle, struct Window *window,
     D0    -$60                A1           A2

                               APTR future)
                               A3

FUNCTION

     Pops up an Info-Requester with more or less detailed information
     on the currently loaded Sample/Module.
     A window pointer may be given to select the place to pop it up.

INPUT(S)

     handle - a valid handle
     window - a valid Window Pointer or NULL
     future - always NULL yet


RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_SetReqIOWindow
```

1.15 superplay.library/SPL_SetReqIOWindow

```
NAME
     SPL_SetReqIOWindow  (V1)

SYNOPSIS

     ULONG SPL_SetReqIOWindow(APTR handle, struct Window *window)
     D0    -$66               A1           A2
```

```
FUNCTION

     Sets a new Default-Window (default : IntuitionBase->FirstWindow)
     for any Requester IO.

INPUT(S)

     handle - a valid handle
     window - a valid Window Pointer or NULL

RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_FileInfoReq
```

1.16 superplay.library/SPL_ReadPlayData

```
NAME
     SPL_ReadPlayData  (V1)

SYNOPSIS

     ULONG SPL_ReadPlayData(APTR handle, char *filename)
     D0    -$6c             A1           A2

FUNCTION

     Loads but NOT plays the data of the Sample/Module described by
     FileName.
     The handle is initialized and the fitting SPObject is opened
     to access the Sample/Module data for use with SPL_SuperWrite.

INPUT(S)

     handle   - a valid handle
     filename - a valid AmigaDOS FilePath and -Name

RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_SetWriteName, SPL_SetWriteType, SPL_SuperWrite
```

1.17 superplay.library/SPL_ContinueReplay

```
NAME
     SPL_ContinueReplay  (V1)

SYNOPSIS

     ULONG SPL_ContinueReplay(APTR handle)
     D0    -$72               A1

FUNCTION

     Some SPObjects support to continue a Sample/Module which
     has been stopped by calling SPL_StopReplay.

INPUT(S)

     handle - a valid handle

RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_SuperPlay, SPL_StopReplay
```

1.18 superplay.library/SPL_FastForward

```
NAME
     SPL_FastForward  (V1)

SYNOPSIS

     ULONG SPL_FastForward(APTR handle)
     D0    -$78            A1

FUNCTION

     Some SPObjects might support a "cassette recorder"-like way
     to browse trough a Sample/Module via FastForward and Rewind.

INPUT(S)

     handle - a valid handle

RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_FastBackward
```

1.19 superplay.library/SPL_FastBackward

```
NAME
     SPL_FastBackward  (V1)

SYNOPSIS

     ULONG SPL_FastBackward(APTR handle)
     D0    -$7e            A1

FUNCTION

     Some SPObjects might support a "cassette recorder"-like way
     to browse trough a Sample/Module via FastForward and Rewind.

INPUT(S)

     handle - a valid handle

RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_FastForward
```

1.20 superplay.library/SPL_GetSampleList

```
NAME
     SPL_GetSampleList  (V2)

SYNOPSIS

     ULONG SPL_GetSampleList(APTR handle, struct SPO_SampleList **list)
     D0    -$84              A1           A2

FUNCTION

     While/after loading a Sample-File Version 2 SPObjects might create
     a special list of all Samples, which appear in the File.
     This will usually be one one Sample for SampleFiles, but many more
     for ModuleFiles.

     Not all SPObjects, especially not all ModuleType-SPObjects,
     may support this and will return an error value or an empty list.

INPUT(S)

     handle - a valid handle
     list   - a pointer to a SPO_SampleList pointer

RESULT
```

```
     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_SetSampleList
```

1.21 superplay.library/SPL_SetSampleList

```
NAME
     SPL_SetSampleList  (V2)

SYNOPSIS

     ULONG SPL_SetSampleList(APTR handle, struct SPO_SampleList *list)
     D0    -$8a              A1           A2

FUNCTION

     For saving Sample-Files with Version 2 SPObjects, there may be used
     a special list of all Samples, which should appear in the File.

     This will usually be one one Sample for SampleFiles, but many more
     for ModuleFiles.

     Not all SPObjects, especially not all ModuleType-SPObjects,
     may support this and can return SPERR_ACTION_NOT_SUPPORTED.

INPUT(S)

     handle - a valid handle
     list   - a pointer to a SPO_SampleList

RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_GetSampleList
```

1.22 superplay.library/SPL_GetFileType

```
NAME
     SPL_GetFileType  (V4)

SYNOPSIS

     ULONG SPL_GetFileType(APTR handle, char *filename, ULONG *filetype)
     D0    -$90            A1           A2              A3

FUNCTION
```

```
     Finds out superplay-specific FileType-Code
     (redefined with every re-initialization of the Library) or
     SP_FILE_TYPE_UNKNOWN (== NULL == FALSE).

     Use the following call for a simple check :

     handle   = SPL_AllocHandle(N);
     SPerr    = SPL_GetFileType(handle, filename, &filetype);
                SPL_FreeHandle(handle);

     This handle should NOT be used for any further operations
     on the file (will be opened and checked twice but only
     closed once, etc).
     Initialization operations are allowed nevertheless
     (e.g. SPL_InitHandleAsClip() ).

     Note, that this function fills in FILETYPES, not SUBTYPES.
     For e.g. writing you must specify SUBTYPES.

     FILETYPES are only for short identification and do only
     specify the global file type (8SVX, ST).

INPUT(S)

     handle   - a valid handle
     filename - a valid AmigaDOS FilePath and -Name
     filetype - pointer to ULONG for SP_FILETYPE-Value

RESULT

     NULL or an adequate SPERR-Errorcode.

SEE ALSO

     SPL_AllocHandle(), SPL_FreeHandle()
```