FAME-DoorCommands

SieGeL/tRSi

COLLABORATORS			
	TITLE : FAME-DoorCommands		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	SieGeL/tRSi	February 12, 2023	

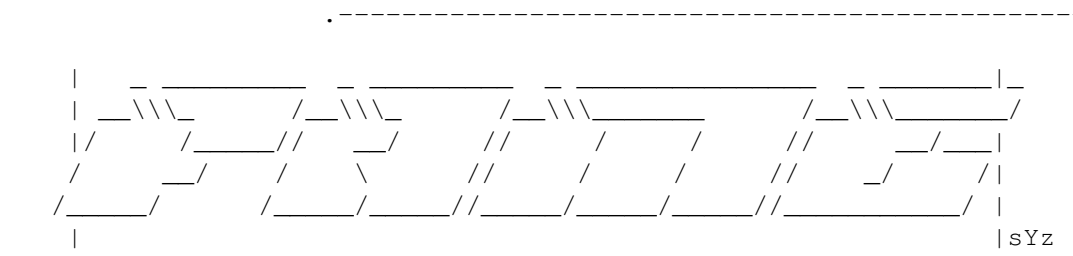
REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FAME-DoorCommands

1.1 FAME DOOR-COMMAND OVERVIEW



F_inal A_miga M_ailbox E_ngine

FAME DOORCOMMAND OVERVIEW

Contents

Back to Main Index

All Amigaguide documentations by:

David 'Strider' Wettig
Sascha 'SieGeL' Pfalz
Oliver 'Bloodrock' Lange
Kai 'big-rat' Hoepner

1.2 Pre-Release Informations

```
                              Pre-Release Informations

FAME is a brand new MailBox program which seems to be becomming very popular
and FAMEous as we can see from the very positive reactions we receive daily.

We already have many bug-reports and suggestions and we believe we will
get tonnes more from the very first release onwards.

We work hard on fixing all reported bugs and realize suggestions as best we
can, but it's impossible to do everything at the same time.

The same with the documents.
The documents are not complete at all... we have to enhance and enlarge
them as we do with the FAME executables.

You have bought FAME at an early stage and because of this not everything
written here will be 100% correct, but we are working on it.


Known not described parts in FAMECommands.guide

 - None.
```

1.3 FAME Door Commands Contents

```
                                  FAME DOORCOMMAND OVERVIEW
                            ~~~~~~~~~~~~~~~~~~~~~~~~~~~~


                        List of all available Commands

                       General Infos about DoorCommands

                           DoorPort-Message Struct

                       Additional Informations for Coders

                           How to contact the Authors
                        This Document describes all freely supported Door-Commands for  ↩
                             FAME,
              currently there exists 396 Commands which are described here.

            This Database and it's functions are related to the include files
                     and developerfiles for FAME-Programming V1.1+.

          Also an enhanced Version of the Door-Commands is available for Developers.
```

1.4 General Informations about this Document

General use of commands:

- A door normally has to give values (numbers like confnumbers/usernumbers, flags etc..) to Data1 in the FAMEDoorMsg-Struct before transfering this to the DoorPort. If more values are needed the next Data will be used: Data2 and then Data3. Values requested by the Door will normally be found in the Data2 Field, only if a value needs a ULONG Data3 must be used.

- The door has to check the ReturnCode of the FAMEDoorMsg-Struct !! See 'FAMEGlobalBBSStructs.h' for definitions of the Returncodes.

- Note that all structure elements of the struct FAMEDoorMsg are beginning with fdom_ (i.e. MyFAMEDoorMsg->fdom_Data1) !

Keyword-Descriptions of FAME-DoorCommands:

All Commands are sorted by some Keywords, which describes the specification of the Command. This Keywords have the following meaning :

KEYWORD	SPECIFICATION	DESCRIPTION
MC <->	Must Commands	Every Door must call these Commands once!
NR <->	Normal Commands	Most String operations (Retrieve/Send datas)
NC <->	Normal Commands	Most String operations (Change datas)
CF <->	Call functions	BBS Functions like Editor,Comp,Modem or Line-length-selectors etc.
SR <->	System Commands	(Retrieve/Send datas)
SC <->	System Commands	(Change datas)
AR <->	Additional Commands	(Retrieve/Send datas)
AC <->	Additional Commands	(Change datas)
RD <->	Reserved!	Do not use it !!!!!!!!!

Define ranges:

MC	1-	9	NR	10-	199
NC	200-	399	CF	400-	599
SR	600-	699	SC	700-	799
AR	800-	899	AC	900-	999

RD - 1000- 9999 - means not secret but reserved.
RD +10000- 99999 + means secret and can't be used in the normal way. (Developers only !!!)

1.5 Message-Struct to comunicate with FAME

Messagestruct for FIM-Doors (FAME):

To communicate with the DoorPort of FAME, you *MUST* (!) open FAME.library, get memory for the structure via FAMEAllocObject(FOBJ_FAMEDoorMsg) and after usage, free the memory with FAMEFreeObject(FOBJ_FAMEDoorMsg). If you allocate the memory for this structure by yourself, your programs may crash in future versions of FAME during the enhancement of this structure!! This is

TOTALLY different to other BBS-Systems, but it makes it possible to easily upgrade the BBS without rewriting all doors. More information about the FAME.library functions can be found in the manual FAME.GUIDE.

FAMEDoorMessage struct as defined for FAME V0.366ß and higher:

```
struct  FAMEDoorMsg {
struct  Message fdom_Msg;           /* MessagePort Structure (System)    */
char    fdom_IOString[202];         /* In/OutPut String                  */
STRPTR  fdom_StringPtr;             /* String pointer                    */
long    fdom_Command,               /* Command Node<->Door               */
        fdom_Data1,                 /* Data 1 to transfer data           */
        fdom_Data2;                 /* Data 2 to transfer data           */
ULONG   fdom_Data3;                 /* Data 3 to transfer data           */
long    fdom_ReturnCode,            /* Returncode                        */
        fdom_Node;                  /* NodeNumber                        */
ULONG   fdom_InternalBits;          /* FAME internal data bit flags      */
APTR    fdom_StructDummy1,          /* APTR 1 for SystemStructs          */
        fdom_StructDummy2,          /* APTR 2 for SystemStructs          */
        fdom_StructDummy3;          /* APTR 3 for SystemStructs          */
STRPTR  fdom_StringPtr2;            /* A second string pointer           */
ULONG   fdom_Data4,                 /* Data 4 to transfer data (future)  */
        fdom_BitFlags;              /* A bit mask to give the door infos */
struct  MsgPort *fdom_ExternalPort; /* External AbortIO port             */
};
```

For examples how to use this struct and communicate with FAME, take a look at the Door-Programmer documentation and the included examples from the developers disk or check out the examples shipped with the Userdevelopment archive.

Returncodes for the DoorMsgStruct:

```
ReturnCode =  5     Command aborted
ReturnCode =  4     Command not implemented yet
ReturnCode =  3     Command denied
ReturnCode =  2     Command doesn't exists
ReturnCode =  1     Command not successfully executed
ReturnCode =  0     Command successfully executed
ReturnCode = -1     Abort of the Door requested   -> Door must end now!
ReturnCode = -2     Carrier lost                  -> Door must end now!
ReturnCode = -3     Unknown error                 -> Door must end now!
```

On negative Returncodes the Door *MUST* close *IMMEDIATELY* with the last command being:

```
MC_ShutDown
 !!!
```

If you are using the doorstartup supplied with the FAMEDevelopment archive, those errors are automatically checked by the Startup and also the appropiate actions will be taken if any error occures, so it's strongly recommended to use only our supplied Startup header!

1.6 Additional Informations for DoorProgrammers

```
                    Informations for DoorProgrammers

We have changed the complete layout of the DoorStartup found in FAME
Developer archive V1.0. Please don't use any of the older startups, as
we have now defined a so-called standard for doors, and the DoorStartup
will be moved in the near future to FAME.library, so please make your own
startups as compatible as possible to our supplied version.

If you wish to use our startup, you should call it this way:

1. Initialize your application (get memory, check for nodenumber via the
   command line) and make any other allocations you require. Also you have
   to supply a function called 'ShutDown(long error)' to allow the startup
   code to call it when any error occures. You should free all your
   resources inside this function and maybe also exit your door, but this
   is not a must. But be warned:

   IF OUR STARTUP CALLS YOUR SHUTDOWN() FUNCTION, THE DOORPORT AND THE
   WHOLE COMMUNICATION TO FAME IS NO LONGER VALID!!! SO DON´T USE ANY
   OF THE SUPPORT FUNCTIONS OUT OF THE SHUTDOWN() FUNCTION OR YOUR SYSTEM
   MAY CRASH!

2. Fill out the global Stringpointer 'FAMEDoorPort' with the supplied node
   number, i.e. SPrintf(FAMEDoorPort,"FAMEDoorPort%ld",node); and call
   the FIMStart() function. This will try to lock the message port,
   allocates global memory pointers and register your door to FAME.
   Of course you have to check out the returncode of FIMStart() to indicate
   any error occured during initialization (See DoorStartup.c for further
   informations)

3. Now the communication to FAME is established, you may now use the supplied
   support functions to communicate with FAME.

4. When you have finished your door, call FIMEnd() to end the communication
   to FAME. This function will call your supplied ShutDown() function after
   removing all allocated doorport stuff, so DON´T USE any of the doorport
   commands inside the ShutDown() function!


That's a short overview how to handle our new DoorStartup, please check also
the supplied example code for detailed informations about writing doors for
FAME!
```

1.7 List of all Commands described here

```
                        Please choose which Type of Command-Overview you wish :


               MC - MUST COMMANDS
                - Commands you must start once

               NR - NORMAL COMMANDS
```

```
 - Normal commands (Retrieve)

NC - NORMAL COMMANDS
 - Normal commands (Change)

CF - CALL FUNCTION
 - Call commands (Editors, etc.)

SR - SYSTEM COMMANDS
 - System commands (Retrieve)

SC - SYSTEM COMMANDS
 - System commands (Change)

AR - ADD. COMMANDS
 - Additional commands (Retrieve)

AC - ADD. COMMANDS
 - Additional commands (Change)

RD - RESERVED COMMS
 - Reserved Commands
```

1.8 Must Commands - EVERY DOOR HAS TO USE THEM ONCE !

```
- These commands are used to register your Doors, you have to call ↩
    them
at least once when starting/exiting your Door!


MC_DoorStart         1

MC_ShutDown          2

MC_ShutDownLastWords 3
                              -> 3 commands <-
```

1.9 Normal Commands - Retrieve/Send datas

```
- These commands are used for most String-Operations, but are Read ↩
   -only!


NR_SendStr             10

NR_SendStrCRLF        11

NR_SendStrCon          12

NR_SendStrSer         13

NR_PromptChars         14
```

```
NR_HotKey             15
NR_BBSName             16
NR_SysOp              17
NR_SetFlagFile         18
NR_GetFlagFile        19
NR_ActiveNode          20
NR_ActiveNodes        21
NR_TimeOut             22
NR_MainLine           23
NR_NodeID              24
NR_MinUpCPS           25
NR_GetConfNum          26
NR_SearchAccount      27
NR_StampTime           28
NR_CurrTime           29
NR_ThisConfAccess      30
NR_Name               31
NR_Password            32
NR_Location           33
NR_From                34
NR_PhoneNumber        35
NR_SlotNumber          36
NR_AccessLevel        37
NR_RatioType           38
NR_Ratio              39
NR_CompType            40
NR_ModemType          41
NR_MessagePosted       42
```

```
NR_MessageRead        43
NR_NoCalls             44
NR_TimeLastOn         45
NR_TimeUsed            46
NR_TimeLimit          47
NR_TimeRemain          48
NR_StampLastOn        49
NR_ConfAccess          50
NR_Uploads            51
NR_Downloads           52
NR_BytesUpload        53
NR_BytesDownload       54
NR_DailyByteLimit     55
NR_DailyFileLimit      56
NR_DailyByteDld       57
NR_DailyFileDld        58
NR_DailyByteBonus     59
NR_DailyFileBonus      60
NR_Expert             61
NR_NumLines            62
NR_Birthday           63
NR_MenuPrompt          64
NR_EditorType         65
NR_XferProt            66
NR_LostCarrier        67
NR_Zoom                68
NR_SysLanguage        69
NR_Language            70
NR_LineCount          71
```

NR_AnsiColor 72

NR_SentBy 73

NR_AutoFileID 74

NR_NewMessage 75

NR_Goodbye 76

NR_ViewFlag 77

NR_ZippyFlag 78

NR_ReplyMSGFlag 79

NR_NukedFiles 80

NR_NukedBytes 81

NR_MinDownCPS 82

NR_GetEditString 83

NR_ResetFlagFile 84

NR_DeleteFlagFile 85

NR_DelFlagFileNum 86

NR_GetFullArg 87

NR_GetArgument1 88

NR_GetArgument2 89

NR_GetArgument3 90

NR_GetArgument4 91

NR_WaitChar 92

NR_GetConFontSize 93

NR_ResetANSI 94

NR_ResetANSIOnExit 95

NR_CONNumLines 96

NR_NumberOfChats 97

NR_NumberOfPages 98

NR_NumberOfDayPages 99

```
NR_NumOfPagesAllowed  100

NR_NumberOfDayRelogs 101

NR_NumOfRelogsAllowed 102

NR_DoorHelp          103

NR_SetDoorReturnCode  104

NR_GetDoorReturnCode 105

NR_ClrFileFlgLst      106

NR_ClrFileFlgLsts    107

NR_GetRelativeStatus  108

NR_AbsToRel          109

NR_RelToAbs           110

NR_GetAbortIOPort    111

NR_GetDoorDatas       112

NR_GetDoorCallName   113

NR_GetUserLevelFlags  114
                                  -> 105 commands <-
```

1.10 Normal Commands - Change datas

```
               - These commands are used for most string-operations, but are ↩
                  write-only,
meaning that you can only change datas with them!


               NC_TimeOut           200

               NC_Name             201

               NC_Password          202

               NC_Location         203

               NC_From              204

               NC_PhoneNumber      205

               NC_AccessLevel       206

               NC_RatioType        207

               NC_Ratio             208
```

```
NC_CompType         209
NC_ModemType         210
NC_MessagePosted    211
NC_MessageRead       212
NC_NoCalls          213
NC_TimeLastOn        214
NC_TimeUsed         215
NC_TimeLimit         216
NC_TimeRemain       217
NC_Uploads           218
NC_Downloads        219
NC_BytesUpload       220
NC_BytesDownload    221
NC_DailyByteLimit    222
NC_DailyFileLimit   223
NC_DailyByteDld      224
NC_DailyFileDld     225
NC_DailyByteBonus    226
NC_DailyFileBonus   227
NC_Expert            228
NC_NumLines         229
NC_Birthday          230
NC_MenuPrompt       231
NC_EditorType        232
NC_XferProt         233
NC_Zoom              234
NC_SysLanguage      235
NC_Language          236
```

```
NC_LineCount        237

NC_AnsiColor         238

NC_SentBy           239

NC_AutoFileID        240

NC_NewMessage       241

NC_Goodbye           242

NC_ViewFlag         243

NC_ZippyFlag         244

NC_ReplyMSGFlag     245

NC_NukedFiles        246

NC_NukedBytes       247

NC_MinUpCPS          248

NC_MinDownCPS       249

NC_LostCarrier       250

NC_NumberOfChats    251

NC_NumberOfPages     252

NC_NumberOfDayPages 253

NC_NumberOfDayRelogs 254

NC_Nuked            255

NC_NukedAfter        256
                                    -> 57 commands <-
```

1.11 Call Commands - Used for editors, computertypes etc.

```
              - This commands are used to write your own editors, Modem- or  ↩
                 Computer-
selectors etc.


              CF_ShowText        400

              CF_ShowTextSuffix  401

              CF_ShowTextSufLvl  402

              CF_ExecuteCommand  403
```

```
CF_InternalCmd     404

CF_ZModemSend      405

CF_ZModemReceive   406

CF_ZModemSendLst   407

CF_ReturnCommand   408

CF_SetFlagFile     409

CF_GetFlagFile     410

CF_CallersLog      411

CF_UDLog           412

CF_DoorLog         413

CF_CompType        414

CF_ModemType       415

----------------------

CF_Language        417

CF_NumLines        418

CF_ShTxtSufLvlCyc  419

CF_ShowTextSetable 420

CF_InternReturnCmd 421

CF_DoCallersLog    422

CF_SaveWherePhase  423

CF_RestWherePhase  424

CF_FileCopyMove    425

CF_SysOpChat       426

CF_SpecialCmd      427

CF_GetUserConfXS   428
                            -> 29 commands <-
```

1.12 System Commands - Retrieve/Send datas

```
              - This commands are used to read in BBS-specific Information. All ↩
                 commands
listed here are READ-ONLY!

              SR_ConfName        600

              SR_ConfLocation     601

              SR_ConfNum         602

              SR_BBSLocation      603

              SR_Status          604

              SR_ScreenAdress     605

              SR_TaskPri         606

              SR_RawScreenAdress  607

              SR_FAMEVersion     608

              SR_ChatSet          609

              SR_ENVStat         610

              SR_NodeDevice       611

              SR_NodeUnit        612

              SR_NodeBaud         613

              SR_NodeNumber      614

              SR_MCI              615

              SR_GetTask         616

              SR_NodeBaudRate     617

              SR_LogonType       618

              SR_ScrLeft          619

              SR_ScrTop          620

              SR_ScrWidth         621

              SR_ScrHeight       622

              SR_PurgeLine        623

              SR_NonStopText     624

              SR_GoodFile         625
```

```
SR_LastAccountNum  626

SR_PurgeLineStart   627

SR_PurgeLineEnd    628

SR_BBSOrigin        629

SR_DefLineLen      630

SR_NumberOfNodes    631

SR_FileDescUndLine 632

SR_NumberOfConfs    633

SR_ConfNameLoc     634

SR_FAMEDataFileVers 635

SR_ConnectString   636
                                -> 37 commands <-
```

1.13 System Commands - Change datas

```
- This commands are used to change BBS-specific Informations. All  ↩
   commands
listed here are WRITE-ONLY!


----------------------

----------------------

SC_ChatSet         702

SC_ENVStat         703

SC_NonStopText     704

SC_DefLineLen      705

SC_FileDescUndLine 706

SC_GoodFile        707
                                -> 6 commands <-
```

1.14 Additional Commands - Retrieve/Send datas

```
- All misc. Commands listed here are READ ONLY !
```

```
AR_GetKey           800
AR_WaitRAWChar      801
AR_EmulateFKeyHelp  802
AR_GetCharHex       803
AR_EditFile         804
AR_Dump             805
AR_UserStatus       806
AR_NewScan          807
AR_Hacks            808
AR_LastConf         809
AR_ConfReJoin       810
AR_HighUpCPS        811
AR_HighDownCPS      812
AR_Baud             813
AR_MsgCLS           814
AR_FileCLS          815
AR_UGlobal          816
AR_DGlobal          817
AR_FileBase         818
AR_Hide             819
AR_MsgRooming       820
AR_StringEdit       821
AR_CryptPW          822
AR_CryptFlag        823
AR_UserFileBPW      824
AR_DropDTR          825
AR_Userlimit        826
AR_MaxNameFailure   827
AR_MaxUserPWFail    828
```

```
AR_MaxSysPWFailure 829

AR_MaxNUPFailure   830

AR_MaxUserHacks    831

AR_ScreensPath     832

AR_SysPWPrompt     833

AR_NewUserPWPrompt 834

AR_UsernamePrompt  835

AR_UserPWPrompt    836

AR_PausePrompt     837

AR_SysOpChatColor  838

AR_UserChatColor   839

AR_UploadPathI     840

AR_DownloadPathI   841

AR_AdditioUlPaths  842

AR_AdditioDlPaths  843

AR_NumberofDirs    844

AR_GiveUlBytes     845

AR_TakeDlBytes     846

AR_FlagFilePath    847

AR_StringToNode    848

AR_GetNodeEnv      849

AR_FullEdStruct    850

AR_SendStr         851

AR_SendStrCon      852

AR_SendStrSer      853

AR_ResetNumFlags   854

AR_SetNumFlag      855

AR_GetNumFlag      856
```

```
AR_NodeVersion     857

AR_NodeStartTime   858

AR_ServerVersion   859

AR_ServerStartTime 860

AR_HotKey          861

AR_DropDtrOnNode   862
                              -> 63 commands <-
```

1.15 Additional Commands - Change datas

```
- All misc. Commands listed here are WRITE ONLY !


AC_UserStatus      900

AC_NewScan         901

AC_HackCount       902

AC_LastConf        903

AC_ConfReJoin      904

AC_HighUpCPS       905

AC_HighDownCPS     906

AC_Baud            907

AC_MsgCLS          908

AC_FileCLS         909

AC_UGlobal         910

AC_DGlobal         911

AC_FileBase        912

AC_Hide            913

AC_MsgRooming      914

AC_StringEdit      915

AC_CryptPW         916

AC_CryptFlag       917

AC_UserFileBPW     918
```

```
AC_Userlimit       919

AC_MaxNameFailure  920

AC_MaxUserPWFail   921

AC_MaxSysPWFailure 922

AC_MaxNUPFailure   923

AC_MaxUserHacks    924

AC_ScreensPath     925

AC_SysPWPrompt     926

AC_NewUserPWPrompt 927

AC_UsernamePrompt  928

AC_UserPWPrompt    929

AC_PausePrompt     930

AC_SysOpChatColor  931

AC_UserChatColor   932

AC_NumberofDirs    933

AC_GiveUlBytes     934

AC_TakeDlBytes     935

AC_ServerAction    936

AC_ServerActionCPS 937
                                -> 38 commands <-
```

1.16 Reserved Commands - DON'T USE THEM !

```
                - This commands are special commands, so only use them if you know ↩
                    what
you are doing !!!


                RD_ShowFlags       1000

                RD_ShowAllFlags    1001

                RD_ConfigLoad      1002

                RD_IconifyIs       1003
```

```
RD_Iconify          1004

RD_Spy              1005

RD_NewUser          1006

RD_LoadAccount      1007

RD_SaveAccount      1008

RD_LoadConfDat      1009

RD_SaveConfDat      1010

RD_AppendAccount    1011

RD_DoOnMaxUserHack 1012

RD_ServerCommand    1013

RD_GetMciFlag       1014

RD_MciFlag          1015

RD_LoadAccountMisc 1016

RD_SaveAccountMisc 1017

RD_ASLLocalUlPath   1018

RD_ASLTextViewPath 1019

RD_ASLSendFilePath 1020

RD_StartNodeCmd     1021

RD_BeADoorOnNode    1022

RD_OpenDoorPort     1023

RD_CloseDoorPort    1024

RD_GetServerList    1025

RD_StringToNode     1026

RD_StringToNodes    1027

RD_CONStatus        1028

RD_SERStatus        1029

RD_SaveMsgFile      1030

RD_SaveMsgList      1031

RD_FreeMsgListFEd   1032
```

```
RD_ActASLULPath    1033
RD_ActASLTextViewP 1034
RD_ActASLDlPath    1035
RD_NodeScrToFront  1036
RD_SetServerActCol 1037
RD_InitNumFlag     1038
RD_RemoveNumFlag   1039
RD_ResetNumFlags   1040
RD_SetNumFlag      1041
RD_GetNumFlag      1042
RD_GetUserDataLoc  1043
RD_ConfName        1044
RD_ConfLocation    1045
RD_FGetNextConf    1046
RD_FGetPrevConf    1047
RD_FGetConfNum     1048
RD_FGetRealConfNum 1049
RD_FGetConfBase    1050
RD_FGetAktConf     1051
RD_GiveNumFromConf 1052
RD_SaveUFlagList   1053
RD_ClearUFlagList  1054
RD_GetUFlagList    1055
RD_GetMailHeader   1056
RD_ShowMailHeader  1057
                              -> 58 commands <-
```

1.17 MC_DoorStart 1

```
COMMAND: MC_DoorStart         1

Register a new (your) door, Door-Counter increases

YOU HAVE TO USE THIS COMMAND BEFORE USING ANY OTHER COMMAND !

INPUT:

None.

RETURN:

None.
```

1.18 MC_ShutDown 2

```
                COMMAND: MC_ShutDown          2

Tells the node that the door is shutting down, Door-Counter decreases.

YOU HAVE TO USE THIS COMMAND TO EXIT A DOOR ! ALSO IF AN ERROR OCCURS !

See also
                MC_ShutDownLastWords
                 as alternate Command.

INPUT:

None.

RETURN:

None.
```

1.19 MC_ShutDownLastWords 3

```
                COMMAND: MC_ShutDownLastWords 3

Tells the node that the door is shutting down, but unlike
                MC_ShutDown
                you can also write some last words before exiting.

INPUT:

IOString <- The String to be send to the user.

RETURN:

None.
```

1.20 NR_SendStr 10

```
COMMAND: NR_SendStr          10

Sends a string to the user.

INPUT:

IOString <- The String to be sent to the user.

RETURN:

None.
```

1.21 NR_SendStrCRLF 11

```
COMMAND: NR_SendStrCRLF      11

Sends a string to the user including CR/LF.

INPUT:

IOString <- The string to be sent to the user.

RETURN:

None.
```

1.22 NR_SendStrCon 12

```
COMMAND: NR_SendStrCon       12

Sends a string to Console only.

INPUT:

IOString <- The string to be sent to console.

RETURN:

None.
```

1.23 NR_SendStrSer 13

```
COMMAND: NR_SendStrSer       13

Sends a string to serial only

INPUT:
```

```
IOString <- The string to be sent to serial.

RETURN:

None.
```

1.24 NR_PromptChars 14

```
COMMAND: NR_PromptChars      14

Prompt the user for a specified number of chars

INPUT:

Data1    <- Max. chars the User can enter.
Data2    <- Flag for different modes (see below).
IOString <- String to Display in prompt.

RETURN:

Data3    -> Returncode from internal procedure (usable for Data2 flag 3).
IOString -> String typed by the user.

Description of Data2-Flag:

Value | Description
------+-------------------------------------------------------------------
  0   | Simple String-Input without Line-editing and other features
------+-------------------------------------------------------------------
  1   | String-Input for Chatmode. Uses Wordwrapping. (Don't use this to
      | get a complete string, because after wordwrap the string changes!).
------+-------------------------------------------------------------------
  2   | String-Input for Msg-Editor (You can't use this Flag, because it
      | could cause damage in fact of using a list! Flag disabled!).
------+-------------------------------------------------------------------
  3   | String-Input for Bulletin-Viewers (It returns CRSR-Keys left/right,
      | Up&down for the Bullview to switch the -<X>-BullHelp.txt's). See docs
      | for more informations about Cycle-Texts. If using this Flag, the
      | field Data3 contains the following values for the CRSR-Keys :
      |           65 -> Up, 66 -> Down, 67 -> Right, 68 -> Left.
------+-------------------------------------------------------------------
  4   | Same as Flag 0, but all typed chars will be displayed as '*'. Useful
      | for Password-Entries etc. NOTE: If the Sysop has set the Flag
      | "Display Passwords to SysOp" in the SystemEditor, the entered Chars
      | are only displayed on serial output as '***', the console gets the
      | real Output!
------+-------------------------------------------------------------------
  5   | Currently unused.
------+-------------------------------------------------------------------
  6   | A real feature! This is the recommended Flag to use everywhere, users
      | can enter anything with the full comfort of a "REAL" Line-Editor,
      | supporting DEL/BACKSPACE, Inserting/Removing of chars, cursor keys
      | etc..please use this Flag as Default-Value.
------+-------------------------------------------------------------------
```

```
  7   | The same as 4, but here will be no single char prompted when the user
      | types something. No single char means of course also *NO* stars (*)
      | will be printed! If the SysOp has set the Flag "Display Passwords to
      | Sysop", the typed chars will be printed as real input string, but
      | only to console.
------+-------------------------------------------------------------------------
  8   | Numeric mode. Only numeric chars can be typed, no alphabetical chars
      | allowed in this mode.
------^-------------------------------------------------------------------------

NOTE:

Higher values as 8 for Data2-Flag are reserved and can't be used!

If IOString contains data when calling this command, the user will get this
Data as default prompt and can edit those datas (flag 6).
```

1.25 NR_HotKey 15

```
COMMAND: NR_HotKey           15

Gets a char without waiting for it.

INPUT:

None.

RETURN:

Data2    -> The char typed by the user (ASCII-Value)
Data3    -> Where the char was typed: 0 = Console, 1 = Serial
```

1.26 NR_BBSName 16

```
COMMAND: NR_BBSName          16

Retrieve the BBS-Name

INPUT:

None.

RETURN:

IOString -> The BBS Name.
```

1.27 NR_SysOp 17

```
COMMAND: NR_SysOp            17

Retrieve the SysOp Name.

INPUT:

None.

RETURN:

IOString -> The SysOp Name.
```

1.28 NR_SetFlagFile 18

```
COMMAND: NR_SetFlagFile      18

Add files to the list of flagged files.

INPUT:

Data1    <- The number of the conference you want to add flag files to.
            If Data1 lower than 1 or higher than the maximum conferences,
            it will be set to the current conference.
IOString <- The file(s) to be flagged, you can flag more files at the same
            time, but they must have spaces between each other.
            Maximum of 12 chars per filename and overall 201 chars including
            the spaces. Note that the BBS filters double filenames !

RETURN:

Data3    -> Error-Message. See description below.

NOTE:

Data3 Returnvalues:

       0 -> successfully flagged
       1 -> file is already flagged
       2 -> length of IOString is less than 1 Byte, i.e. a Null-String.
      -1 -> list element memory allocation failed.
      -2 -> given conference not found. normally won't happen, because of
            the Data1 check.

When giving more than 1 filename to flag, the Returncodes may not contain
correct values. If you supply only one filename, all returncodes are valid,
else the Returncode will be set for the last filename supplied in the list.
```

1.29 NR_GetFlagFile 19

```
               COMMAND: NR_GetFlagFile      19
```

```
Get File from the Flaglist

INPUT:

Data1    <- The number of the conference you want to get a flagged file from.
            If Data1 lower than 1 or higher than the maximum conferences,
            it will be set to the current conference.

RETURN:

IOString -> The filename of the flagged file.

NOTE:

This command counts the current File number internally. So if you start this
command and get a filename, you get the next filename of the list by calling
this command again until the IOString is empty. To reset the flagging
pointer, use the Command
                NR_ResetFlagFile
                 ! This command resets the flagging
pointer and gives you the *FIRST* flagged File.
After getting it, simply call NR_GetFlagFile repeatly until IOString is
empty.
```

1.30 NR_ActiveNode 20

```
COMMAND: NR_ActiveNode       20

Check if a specific Node exists

INPUT:

Data1    <- The Node you want to check if it exists or not.

RETURN:

Data2    -> Returns 1 if Node exists, 0 if it doesn't.
```

1.31 NR_ActiveNodes 21

```
COMMAND: NR_ActiveNodes      21

Retrieve~a string of active & inactive nodes.

INPUT:

None.

RETURN:

IOString -> The string contains an X at every position if the relative
            Node exists. Byte 0 stands for Node0 !
```

```
            A maximum of 200 Nodes will be checked.
```

1.32 NR_TimeOut 22

```
COMMAND: NR_TimeOut          22

Retrieve the door timeout limit.

INPUT:

None.

RETURN:

Data2    -> Contains the door timeout limit.
```

1.33 NR_MainLine 23

```
                COMMAND :NR_MainLine         23

Retrieve the menu prompt arguments prior to the door being entered.

INPUT:

None.

RETURN:

IOString -> Contains the MainLine, the full argumentstring which was entered
            when starting the door.

NOTE:

With FAME it makes no difference if you insert several spaces between the
arguments typed at the menu prompt because FAME will correctly interpret
them! But with this Function, you get the same string that the user entered
when starting your door, meaning that you have to parse IOString yourself.
If this is too much work for you, try the following Commands:


                NR_GetFullArg
                 or use


                NR_GetArgument1
                ,
                NR_GetArgument2
                ,

                NR_GetArgument3
                ,
                NR_GetArgument4
```

.

NR_GetFullArg gives you the Mainline, but with only 1 space between the
first 4 arguments, all others remain unchanged. The NR_GetArgument(x)
Commands give you the desired Arguments (1-4), but *ALWAYS* without the
Command-Name, only the Arguments are given! For more information take
a look at the chapters for these commands.

1.34 NR_NodeID 24

COMMAND: NR_NodeID 24

Retrieve the Nodenumber of the current node.

INPUT:

None.

RETURN:

Data2 -> The current Nodenumber.

1.35 NR_MinUpCPS 25

COMMAND: NR_MinUpCPS 25

Retreive min. Upload CPS.

INPUT:

None.

RETURN:

Data2 -> Users min. Upload CPS.

1.36 NR_GetConfNum 26

COMMAND: NR_GetConfNum 26

Retrieve the current Confnumber.

INPUT:

None.

RETURN:

Data2 -> Contains the actual Conferencenumber.

1.37 NR_SearchAccount 27

```
COMMAND: NR_SearchAccount    27

Looks for a specific Account in the User datas.

INPUT:

Data1        <- The Usernumber you are searching for.
StructDummy1 <- Pointer to a user.keys Struct.

RETURN:

StructDummy1 -> The User.keys Structure for the specified user.

NOTE:

You *MUST* check the ReturnCode of this Command!!! If it is non-zero
then the StructDummy1 Pointer is not valid !
```

1.38 NR_StampTime 28

```
COMMAND: NR_StampTime        28

Retrieve the current timestring.

INPUT:

None.

RETURN:

IOString -> Contains the current timestring.
```

1.39 NR_CurrTime 29

```
COMMAND: NR_CurrTime         29

Retrieve the current time in seconds since 00:00:00 Greenwhich Mean Time,
January 1, 1970.

INPUT:

None.

RETURN:

Data2    -> Contains the current timevalue.
```

1.40 NR_ThisConfAccess 30

```
COMMAND: NR_ThisConfAccess   30

Retrieve the users conference access of selected Conference.

INPUT:

Data1    <- The Conferencenumber you want to know if the user has access to.
            If Data1 lower than 1 or higher than the maximum conferences,
            then it will be set to the current conference.

RETURN:

Data2    -> 1 if the user has access to the conference, else 0.
```

1.41 NR_Name 31

```
COMMAND: NR_Name             31

Retrieve user name/handle.

INPUT:

None.

RETURN:

IOString -> Contains the Username.
```

1.42 NR_Password 32

```
COMMAND: NR_Password         32

Retrieve user Password.

INPUT:

None.

RETURN:

IOString -> Contains the Password of the User.
```

1.43 NR_Location 33

```
COMMAND: NR_Location         33

Retrieve Users Location.
```

INPUT:

None.

RETURN:

IOString -> Contains the user's location.

1.44 NR_From 34

COMMAND: NR_From 34

Retrieve user's 'from' (Origin).

INPUT:

None.

RETURN:

IOString -> Contains the user's origin.

1.45 NR_PhoneNumber 35

COMMAND: NR_PhoneNumber 35

Retrieve the user's~phonenumber.

INPUT:

None.

RETURN:

IOString -> Contains the user's phonenumber.

1.46 NR_SlotNumber 36

COMMAND: NR_SlotNumber 36

Retrieve the user's slot number.

INPUT:

None.

RETURN:

Data2 -> Contains the user's slot number.

1.47 NR_AccessLevel 37

```
COMMAND: NR_AccessLevel      37

Retrieve user's access level.

INPUT:

None.

RETURN:

Data2    -> Contains user's level.
```

1.48 NR_RatioType 38

```
COMMAND: NR_RatioType        38

Retrieve user's ratiotype.

INPUT:

None.

RETURN:

Data2    -> Contains user's ratiotype.
```

1.49 NR_Ratio 39

```
COMMAND: NR_Ratio            39

Retrieve user's ratio.

INPUT:

None.

RETURN:

Data2    -> Contains user's ratio.
```

1.50 NR_CompType 40

```
COMMAND: NR_CompType         40

Retrieve user's computertype.

INPUT:
```

```
None.

RETURN:

Data2    -> Contains user's computertype Number
IOString -> Contains user's computertype.
```

1.51 NR_ModemType 41

```
COMMAND: NR_ModemType        41

Retrieve user's modemtype.

INPUT:

None.

RETURN:

Data2    -> Contains User's modemtype number.
IOString -> Contains User's modemtype.
```

1.52 NR_MessagePosted 42

```
COMMAND: NR_MessagePosted    42

Retrieve user's number of messages posted.

INPUT:

None.

RETURN:

Data3    -> Contains User's number of messages posted.
```

1.53 NR_MessageRead 43

```
COMMAND: NR_MessageRead      43

Retrieve the number of messages read.

INPUT:

None.

RETURN:

Data3    -> Contains User's messages read.
```

1.54 NR_NoCalls 44

```
COMMAND: NR_NoCalls          44

Retrieve user's number of calls.

INPUT:

None.

RETURN:

Data3    -> Contains User's number of calls.
```

1.55 NR_TimeLastOn 45

```
COMMAND: NR_TimeLastOn       45

Retrieve the time the user last called.

INPUT:

None.

RETURN:

Data2    -> Contains the User's last time called.

NOTE:

Data2 contains the seconds since 00:00:00 GMT-Time, January 1, 1970!
```

1.56 NR_TimeUsed 46

```
COMMAND: NR_TimeUsed         46

Retrieve time used today (in seconds).

INPUT:

None.

RETURN:

Data2    -> Contains User's time used.
```

1.57 NR_TimeLimit 47

```
COMMAND: NR_TimeLimit        47

Retrieve time allowed for a user (in seconds).

INPUT:

None.

RETURN:

Data2    -> Contains User's time limit.
```

1.58 NR_TimeRemain 48

```
COMMAND: NR_TimeRemain       48

Retrieve total time remaining for today (in seconds).

INPUT:

None.

RETURN:

Data2    -> Contains User's total time remaining for today.
```

1.59 NR_StampLastOn 49

```
COMMAND: NR_StampLastOn      49

Retrieve a date string containing the date when the user last logged on.

INPUT:

None.

RETURN:

IOString -> Contains User's last time called as a 26 Byte ASCII-String.
```

1.60 NR_ConfAccess 50

```
COMMAND: NR_ConfAccess       50

Retrieve user's conference access.

INPUT:

None.
```

```
RETURN:

IOString -> Contains User's conference access alias.
```

1.61 NR_Uploads 51

```
COMMAND: NR_Uploads          51

Retrieve user's number of uploads (Files).

INPUT:

None.

RETURN:

Data3    -> Contains User's number of uploaded Files.
```

1.62 NR_Downloads 52

```
COMMAND: NR_Downloads        52

Retrieve~user's number of downloads (Files).

INPUT:

None.

RETURN:

Data3    -> Contains User's number of downloaded Files.
```

1.63 NR_BytesUpload 53

```
COMMAND: NR_BytesUpload      53

Retrieve user's uploaded bytes.

INPUT:

None.

RETURN:

Data3    -> Contains User's uploaded bytes.
```

1.64 NR_BytesDownload 54

```
COMMAND: NR_BytesDownload    54

Retrieve user's downloaded bytes.

INPUT:

None.

RETURN:

Data3    -> Contains User's downloaded bytes.
```

1.65 NR_DailyByteLimit 55

```
COMMAND: NR_DailyByteLimit   55

Retrieve user's daily bytes limit.

INPUT:

None.

RETURN:

Data3    -> Contains user's daily byte limit.
```

1.66 NR_DailyFileLimit 56

```
COMMAND: NR_DailyFileLimit   56

Retrieve user's daily files limit.

INPUT:

None.

RETURN:

Data3    -> Contains User's daily file limit.
```

1.67 NR_DailyByteDld 57

```
COMMAND: NR_DailyByteDld     57

Retrieve user's daily bytes downloaded.

INPUT:
```

None.

RETURN:

Data3 -> Contains User's daily bytes downloaded.

1.68 NR_DailyFileDld 58

COMMAND: NR_DailyFileDld 58

Retrieve user's daily files downloaded.

INPUT:

None.

RETURN:

Data3 -> Contains User's daily files downloaded.

1.69 NR_DailyByteBonus 59

COMMAND: NR_DailyByteBonus 59

Retrieve user's daily byte bonus.

INPUT:

None.

RETURN:

Data3 -> Contains User's daily byte bonus.

1.70 NR_DailyFileBonus 60

COMMAND: NR_DailyFileBonus 60

Retrieve user's daily file bonus.

INPUT:

None.

RETURN:

Data3 -> Contains User's daily file bonus.

1.71 NR_Expert 61

```
COMMAND: NR_Expert           61

Retrieve users expert mode.

INPUT:

None.

RETURN:

Data2    -> 1 means Expertmode is set, 0 means it's not set.
```

1.72 NR_NumLines 62

```
COMMAND: NR_NumLines         62

Retrieve users number of lines to view.

INPUT:

None.

RETURN:

Data2    -> Contains User's number of lines.

NOTE:

The number of lines to view is really one line more than you will get
here, because of scrolling texts there should be always one line kept
as a reserve.
```

1.73 NR_Birthday 63

```
COMMAND: NR_Birthday         63

Retrieve users birthday.

INPUT:

None.

RETURN:

IOString -> Contains User's birthday.
```

1.74 NR_MenuPrompt 64

```
COMMAND: NR_MenuPrompt       64

Retrieve users menuprompt.

INPUT:

None.

RETURN:

IOString -> Contains User's menuprompt.
```

1.75 NR_EditorType 65

```
COMMAND: NR_EditorType       65

Retrieve users Editor type.

INPUT:

None.

RETURN:

Data2    -> Contains User's editor type.
```

1.76 NR_XferProt 66

```
COMMAND: NR_XferProt         66

Retrieve users (last) used Transfer-Protocol

INPUT:

None.

RETURN:

Data2    -> Contains the User's (last) used transfer protocol value.
IOString -> Contains the User's (last) used transfer protocol name.
```

1.77 NR_LostCarrier 67

```
COMMAND: NR_LostCarrier       67

Retrieve users number of lost carriers.

INPUT:
```

None.

RETURN:

Data2 -> Contains User's numbers of lost carriers.

1.78 NR_Zoom 68

COMMAND: NR_Zoom 68

Retrieve users ZOOM-Type.

INPUT:

None.

RETURN:

Data2 -> Contain User's Zoom type.

1.79 NR_SysLanguage 69

COMMAND: NR_SysLanguage 69

Retrieve users text language specification.

INPUT:

None.

RETURN:

IOString -> Contains User's text language specification.

1.80 NR_Language 70

COMMAND: NR_Language 70

Retrieve users current language specification.

INPUT:

None.

RETURN:

Data2 -> Contain User's current language setting.

1.81 NR_LineCount 71

```
COMMAND: NR_LineCount        71

Retrieve users current number of lines viewed.

INPUT:

None.

RETURN:

Data2    -> Contains the current number of lines viewed.
```

1.82 NR_AnsiColor 72

```
COMMAND: NR_AnsiColor        72

Retrieve users ANSI-Flag setting.

INPUT:

None.

RETURN:

Data2    -> Contains User's Ansiflag.
```

1.83 NR_SentBy 73

```
COMMAND: NR_SentBy           73

Retrieve users Sent-By line.

INPUT:

Data1    <- The conferencenumber from which you want to get the SentByLine

RETURN:

IOString -> Contains User's Sent-By line.
```

1.84 NR_AutoFileID 74

```
COMMAND: NR_AutoFileID       74

Retrieve users Auto FILE_ID.DIZ Flag.

INPUT:
```

```
None.

RETURN:

Data2    -> Contains User's File-ID flag.
```

1.85 NR_NewMessage 75

```
COMMAND: NR_NewMessage       75

Retrieve users NewMessage Status.

INPUT:

None.

RETURN:

Data2    -> Contain User's NewMessage flag.
```

1.86 NR_Goodbye 76

```
COMMAND: NR_Goodbye          76

Retrieve users GoodBye Flag.

INPUT:

None.

RETURN:

Data2    -> Contain User's (fast)GoodBye flag.
```

1.87 NR_ViewFlag 77

```
COMMAND: NR_ViewFlag         77

Retrieve users view flag.

INPUT:

None.

RETURN:

Data2    -> Contain User's view flag.
```

1.88 NR_ZippyFlag 78

```
COMMAND: NR_ZippyFlag        78

Retrieve the Zippy Flag.

INPUT:

None.

RETURN:

Data2    -> Contain User's zippy flag.
```

1.89 NR_ReplyMSGFlag 79

```
COMMAND: NR_ReplyMSGFlag     79

Retrieve the reply message flag.

INPUT:

Data1    <- The conferencenumber from which you want to get the ReplyFlag.

RETURN:

Data2    -> Contains User's reply msg flag.
```

1.90 NR_NukedFiles 80

```
COMMAND: NR_NukedFiles       80

Retrieve user's number of nuked files.

INPUT:

None.

RETURN:

Data3    -> Contain User's number of nuked files.
```

1.91 NR_NukedBytes 81

```
COMMAND: NR_NukedBytes       81

Retrieve user's number of nuked bytes.

INPUT:
```

```
None.

RETURN:

Data3    -> Contain User's number of nuked bytes.
```

1.92 NR_MinDownCPS 82

```
COMMAND: NR_MinDownCPS       82

Retrieve users min. download CPS.

INPUT:

None.

RETURN:

Data2    -> Contain User's min. download CPS.
```

1.93 NR_GetEditString 83

```
                COMMAND: NR_GetEditString    83

Prompt the user for a specified number of chars and edit string.

INPUT:

Data1    <- Max. chars which can be typed.
Data2    <- Flag for different modes (see below).
IOString <- String to be inserted into the edit array.

RETURN:

IOString -> Filled string with chars typed by the user.
Data3    -> Returncode from internal procedure (usable for Data2 flag 3).

Description of Data2-Flag:

Value | Description
------+-------------------------------------------------------------------
  0   | Simple String-Input without Line-editing and other features
------+-------------------------------------------------------------------
  1   | String-Input for Chatmode. Uses Wordwrapping. (Don't use this to
      | get a complete string, because after wordwrap the string changes!).
------+-------------------------------------------------------------------
  2   | String-Input for Msg-Editor (You can't use this Flag, because it
      | could cause damage in fact of using a list! Flag disabled!).
------+-------------------------------------------------------------------
  3   | String-Input for Bulletin-Viewers (It returns CRSR-Keys left/right,
      | Up&down for the Bullview to switch the BullHelp.txt's). See docs for
```

```
      | more informations about Cycle-Texts. If using this Flag, the field
      | Data3 contains the following values for the CRSR-Keys :
      |           65 -> Up, 66 -> Down, 67 -> Right, 68 -> Left.
------+-------------------------------------------------------------------
  4   | Same as Flag 0, but all typed chars will be displayed as '*'. Useful
      | for Password-Entries etc. NOTE: If the Sysop had set the Flag
      | "Display Passwords to SysOp" in the SystemEditor, the entered Chars
      | are only displayed on serial output as '***', the console gets the
      | real Output!
------+-------------------------------------------------------------------
  5   | Currently unused.
------+-------------------------------------------------------------------
  6   | A real feature! This is the recommended Flag to use everywhere a user
      | could enter anything with the full comfort of a "REAL" Line-Editor,
      | supporting DEL/BACKSPACE, Inserting/Removing of chars etc..please use
      | this Flag as Default-Value.
------+-------------------------------------------------------------------
  7   | The same as 4, but here will be no single char prompted when the user
      | types something. No single char means of course also *NO* stars (*)
      | will be printed! If the SysOp has set the Flag "Display Passwords to
      | Sysop", the typed chars will be printed as real input string, but
      | only to console.
------+-------------------------------------------------------------------
  8   | Numeric mode. Only numeric chars can be typed, no alphabetical chars
      | allowed in this mode.
------^-------------------------------------------------------------------
```

NOTE:

Higher values as 8 for Data2-Flag are reserved and can't be used!

If IOString contains data when calling this command, the user will get this data in the prompt and can edit it (flag 6 only!).

NR_GetEditString is nearly the same as NR_PromptChars. In fact it is the same command, but if you use NR_GetEditString, no Data2 (mode flag) check will be done. Here you can use undocumented, obselete and non existing modes. Beware of what you are doing, it can result in crashes if you use for example flag number 2!

1.94 NR_ResetFlagFile 84

COMMAND: NR_ResetFlagFile 84

Reset and get the first file from the flaglist.

INPUT:

Data1 <- The conferencenumber from which you want to get a flagged file. If Data1 lower than 1 or higher than the maximum conferences, it will be set to the current conference.

RETURN:

```
IOString -> The filename of the first flagged file.

NOTE:

Use NR_ResetFlagFile *FIRST* ! You will reset the internal counter
and get the first flagged file! So to get the whole flagging list of the
specified conference, first call NR_ResetFlagFile to get the first filename
and then repeatly call
                NR_GetFlagFile
                 until IOString is empty.
If NR_ResetFlagFile returns an empty IOString, the flaglist is empty :)
```

1.95 NR_DeleteFlagFile 85

```
COMMAND: NR_DeleteFlagFile   85

Delete file(s) from a flaglist.

INPUT:

Data1    <- The conferencenumber from which you want to delete flagged
            file(s).
Data2    <- If true the file will not be deleted if the FFFL_NODELETE flag
            of the file flag is set. If false flag will be deleted in any
            case.
IOString <- The file or pattern to delete file(s).

RETURN:

None.

NOTE:

You could use the full name or a valid A-DOS filepattern as filename !
If Data1 is lower than 1 or higher than the maximum conferences, it will be
set to the current conference.
```

1.96 NR_DelFlagFileNum 86

```
COMMAND: NR_DelFlagFileNum   86

Delete a file from the flaglist with the fileflagnumber.

INPUT:

Data1    <- The confnumber from which you want to delete a flagged file.
            If Data1 is lower than 1 or higher than the maximum conferences,
            it will be set to the current conference.
Data2    <- The number of the flag you want to delete.

RETURN:
```

None.

1.97 NR_GetFullArg 87

COMMAND: NR_GetFullArg 87

Retrieve the full Argumentstring prior to the door being entered.

INPUT:

None.

RETURN:

IOString -> The string of all arguments given when starting the Door.

NOTE:

The first four arguments were automatically sorted by FAME (argument by argument seperated with a space), all other arguments remain unchanged!

SEE ALSO:

NR_GetArgument1
,
NR_GetArgument2
,
NR_GetArgument3
,
NR_GetArgument4

1.98 NR_GetArgument1 88

COMMAND: NR_GetArgument1 88

Retrieves only the FIRST argument, no other arguments given !

INPUT:

None.

RETURN:

IOString -> Will be the first argument.

SEE ALSO:

NR_GetFullArgs,
NR_GetArgument2
,

NR_GetArgument3
,
NR_GetArgument4

1.99 NR_GetArgument2 89

COMMAND: NR_GetArgument2 89

Retrieves only the SECOND argument, no other arguments given !

INPUT:

None.

RETURN:

IOString -> Will be the second argument.

SEE ALSO:

NR_GetFullArgs,
NR_GetArgument1
,
NR_GetArgument3
,
NR_GetArgument4

1.100 NR_GetArgument3 90

COMMAND: NR_GetArgument3 90

Retrieves only the THIRD argument, no other arguments given !

INPUT:

None.

RETURN:

IOString -> Will be the third argument.

SEE ALSO:

NR_GetFullArgs,
NR_GetArgument1
,
NR_GetArgument2
,
NR_GetArgument4

1.101 NR_GetArgument4 91

```
                COMMAND: NR_GetArgument4     91

Retrieves only the FOURTH argument, no other arguments given !

INPUT:

None.

RETURN:

IOString -> Will be the fourth and all other arguments.

SEE ALSO:

NR_GetFullArgs,
                NR_GetArgument1
                ,
                NR_GetArgument2
                ,
                NR_GetArgument3
```

1.102 NR_WaitChar 92

```
COMMAND: NR_WaitChar         92

Gets a char with waiting for response.

INPUT:

IOString <- String which will be displayed to the user.

RETURN:

Data2    -> The char typed be the user.
Data3    -> Where the char was typed: 0 = from Console, 1 = from Serial.

NOTE:

You get all ASCII-codes and also RAW codes, but the CSI codes are
filtered by FAME, because the BBS has to check for FKeys and others.

The CRSR-Key-Values from CON: and serial are the following:

                             UP -> 4
                           DOWN -> 5
                          RIGHT -> 3
                           LEFT -> 2
```

1.103 NR_GetConFontSize 93

```
COMMAND: NR_GetConFontSize   93

Retrieves the Font height and width from Console.

INPUT:

None.

RETURN:

Data2    -> Height of font.
Data3    -> Width of font.
```

1.104 NR_ResetANSI 94

```
COMMAND: NR_ResetANSI        94

Resets the console and serial attributes to standard values.

INPUT:

None.

RETURN:

None.
```

1.105 NR_ResetANSIOnExit 95

```
                COMMAND: NR_ResetANSIOnExit  95

Sets a flag to force the node to reset console and serial attributes to
standard values when calling
                MC_ShutDown
                 /
                MC_ShutDownLastWords
                .

INPUT:

None.

RETURN:

None.
```

1.106 NR_CONNumLines 96

```
COMMAND: NR_CONNumLines      96

Retrieves the number of lines to view on Console.

INPUT:

None.

RETURN:

Data2    -> Contains the number of lines to view.

NOTE:

This function is important if you must/want create a so-called "Screen-View"
and the user has more lines available on his screen than the Sysop on the
user's node could display. You may handle the Sysop's Display seperatly
from the user's view !

The number of lines to view is really one line more than you will get here,
because of scrolling texts there should be always one line kept as reserve.
```

1.107 NR_NumberOfChats 97

```
COMMAND: NR_NumberOfChats    97

Retrieve users number of chats.

INPUT:

None.

RETURN:

Data3    -> Contains the number of chats.
```

1.108 NR_NumberOfPages 98

```
COMMAND: NR_NumberOfPages    98

Retrieve users number of pages.

INPUT:

None.

RETURN:

Data2    -> Contains the number of pages.
```

1.109 NR_NumberOfDayPages 99

```
COMMAND: NR_NumberOfDayPages 99

Retrieve users number of pages today.

INPUT:

None.

RETURN:

Data2    -> Contains the number of pages today.
```

1.110 NR_NumOfPagesAllowed 100

```
COMMAND: NR_NumOfPagesAllowed 100

Retrieves the number of pages allowed per day for user's access-level.

INPUT:

None.

RETURN:

Data2    -> Contains the number of pages allowed per day.
```

1.111 NR_NumberOfDayRelogs 101

```
COMMAND: NR_NumberOfDayRelogs 101

Retrieve the number of daily relogins.

INPUT:

None.

RETURN:

Data2    -> Contains the number of daily relogins.
```

1.112 NR_NumOfRelogsAllowed 102

```
COMMAND: NR_NumOfRelogsAllowed 102

Retrieve the number of relogins allowed per day.

INPUT:
```

```
None.

RETURN:

Data2    -> Contains the number of relogins per day.
```

1.113 NR_DoorHelp 103

```
                COMMAND: NR_DoorHelp         103

Try to load and display online user help

INPUT:

IOString  <- Minimum Version.Revision string. Example: "1.23"
StringPtr <- Help file name without any pathes.

RETURN:

None.

NOTE:

Nearly all Arguments are invalid after usage ! Especially
                NR_MainLine
                and
                NR_GetArgument1
                 Commands.
```

1.114 NR_SetDoorReturnCode 104

```
COMMAND: NR_SetDoorReturnCode 104

Set Door Returncode.

INPUT:

Data1    <- The new DoorReturnCode.

RETURN:

None.
```

1.115 NR_GetDoorReturnCode 105

```
COMMAND: NR_GetDoorReturnCode 105

Retrieve Door ReturnCode.
```

```
INPUT:

None.

RETURN:

Data2    -> Contains the DoorReturnCode.
```

1.116 NR_ClrFileFlgLst 106

```
COMMAND: NR_ClrFileFlgLst     106

Delete all file flags in a specified conference.

INPUT:

Data1    <- The conference number.
            If Data1 lower than 1 or higher than the maximum conferences,
            it will be set to the current conference.

RETURN:

None.
```

1.117 NR_ClrFileFlgLsts 107

```
COMMAND: NR_ClrFileFlgLsts    107

Delete all file flags in all conferences.

INPUT:

None.

RETURN:

None.
```

1.118 NR_GetRelativeStatus 108

```
COMMAND: NR_GetRelativeStatus 108

Retrieve the relative status. Has this node got relative conferences set?

INPUT:

None.

RETURN:
```

```
Data2    -> The relative status, if <> 0 then relative is set to on.
```

1.119 NR_AbsToRel 109

```
COMMAND: NR_AbsToRel          109

Convert an absolute conference number (real) to the relative one.

INPUT:

Data1    <- Your absolute~conference number.

RETURN:

Data2    -> The relative conference number.

NOTE:

If Data2 is 0, the user has no access to the selected conference!
If you supply a conference number lower than 0 or higher than the maximum
available conference, the current conference number will be set in Data1.
```

1.120 NR_RelToAbs 110

```
COMMAND: NR_RelToAbs          110

Convert a relative conference number to the absolute (real) one.

INPUT:

Data1    <- Your relative~conference number.

RETURN:

Data2    -> The absolute conference number.

NOTE:

If you supply a conference number lower than 0 or higher than the maximum
available conference, the current conference number will be set in Data1.
```

1.121 NR_GetAbortIOPort 111

```
COMMAND: NR_GetAbortIOPort    111

Get the AbortIO Messageport.

INPUT:

None.
```

```
RETURN:

StructDummy1 -> The AbortIO MsgPort structure
```

1.122 NR_GetDoorDatas 112

```
COMMAND: NR_GetDoorDatas      112

Get datas of your door like the path for example.

INPUT:

None.

RETURN:

IOString     -> The path from where FAME has started your door.
StringPtr    -> The complete path and door filename.
StructDummy1 -> Pointer to struct dOORS (Developer only).

NOTE:

            THIS FUNCTION IS ONLY FOR DEVELOPERS, DO NOT USE IT
                IF YOU ARE NOT AN OFFICIAL FAME DEVELOPER,
                        OR YOUR SYSTEM MAY CRASH !
```

1.123 NR_GetDoorCallName 113

```
COMMAND: NR_GetDoorCallName   113

Get the DoorCallName.

INPUT:

None.

RETURN:

IOString -> The DoorCallName.
```

1.124 NR_GetUserLevelFlags 114

```
                COMMAND: NR_GetUserLevelFlags 114

Get the level flags for the current user.

INPUT:

None.
```

```
RETURN:

Data2    -> MaxPages
Data3    -> MaxRelogins
IOString -> Array of level flags.

NOTE:

You will get an array (IOString) where you can use the level defines to see
if a level flag is set:

                        TRUE / 1 = Flag set
                       FALSE / 0 = Flag not set

If the levelflag you are checking is set to 0, the user isn't able to do
this command. If the checked value is <> 0, the according command is set
and the user is able to do this.

For a complete list of all defines look in the FAMEDefine.h file found in
the developer archive of FAME or click
                HERE
                 to get an overview of all flags
defined for this function.
```

1.125 NC_TimeOut 200

```
COMMAND: NC_TimeOut         200

Change the Door timeout limit.

INPUT:

Data1    <- The new door timeout.

RETURN:

None.
```

1.126 NC_Name 201

```
COMMAND: NC_Name            201

Change users name/handle.

INPUT:

IOString <- The new username of the online user (max. 30 chars).

RETURN:

None.
```

NOTE:

If ReturnCode = 1 the username already exists and nothing changed!

1.127 NC_Password 202

COMMAND: NC_Password 202

Change users password.

INPUT:

IOString <- The new password of the online user (max. 20 chars).

RETURN:

None.

1.128 NC_Location 203

COMMAND: NC_Location 203

Change users location.

INPUT:

IOString <- The new location of the online user (max. 30 chars).

RETURN:

None.

1.129 NC_From 204

COMMAND: NC_From 204

Change users from (Origin).

INPUT:

IOString <- The new origin of the online user (max. 30 chars).

RETURN:

None.

1.130 NC_PhoneNumber 205

```
COMMAND: NC_PhoneNumber     205

Change users phone number.

INPUT:

IOString <- The new phonenumber of the online user (max. 15 chars).

RETURN:

None.
```

1.131 NC_AccessLevel 206

```
COMMAND: NC_AccessLevel     206

Change users access level.

INPUT:

Data1    <- The new level of the online user.

RETURN:

None.
```

1.132 NC_RatioType 207

```
COMMAND: NC_RatioType       207

Change users ratiotype.

INPUT:

Data1    <- The new ratiotype of the online user.

RETURN:

None.
```

1.133 NC_Ratio 208

```
COMMAND: NC_Ratio           208

INPUT:

Data1    <- The new ratio of the online user.
```

```
RETURN:

None.
```

1.134 NC_CompType 209

```
COMMAND: NC_CompType        209

Change users computertype code.

INPUT:

Data1    <- The new computertype of the online user.
IOString <- The new computer.

RETURN:

None.
```

1.135 NC_ModemType 210

```
COMMAND: NC_ModemType       210

Change users modemtype code.

INPUT:

Data1    <- The new modemtype of the online user.
IOString <- The new modem.

RETURN:

None.
```

1.136 NC_MessagePosted 211

```
COMMAND: NC_MessagePosted   211

Change users messages posted.

INPUT:

Data3    <- The new messages posted by the online user.

RETURN:

None.
```

1.137 NC_MessageRead 212

```
COMMAND: NC_MessageRead     212

Change the number of messages read by the User.

INPUT:

Data3    <- The new messages read of the online user.

RETURN:

None.
```

1.138 NC_NoCalls 213

```
COMMAND: NC_NoCalls         213

Change users number of calls.

INPUT:

Data3    <- The new number of calls of the online user.

RETURN:

None.
```

1.139 NC_TimeLastOn 214

```
COMMAND: NC_TimeLastOn      214

Change time the user last called.

INPUT:

Data1    <- The new last time called of the online user.

RETURN:

None.

NOTE:

The Value you pass here has to be a LONG value as given from time() !
```

1.140 NC_TimeUsed 215

```
COMMAND: NC_TimeUsed        215

Change users time used today (seconds).

INPUT:

Data1    <- The new time used today of the online user.

RETURN:

None.
```

1.141 NC_TimeLimit 216

```
COMMAND: NC_TimeLimit       216

Change users time allowed today (seconds).

INPUT:

Data1    <- The new time allowed of the online user.

RETURN:

None.
```

1.142 NC_TimeRemain 217

```
COMMAND: NC_TimeRemain      217

Change users total time remaining for today.

INPUT:

Data1    <- The new total time remaining for the online user.

RETURN:

None.
```

1.143 NC_Uploads 218

```
COMMAND: NC_Uploads         218

Change users number of uploaded files.

INPUT:

Data3    <- The new number of uploaded files of the online user.
```

```
RETURN:

None.
```

1.144 NC_Downloads 219

```
COMMAND: NC_Downloads       219

Change users number of downloaded files.

INPUT:

Data3    <- The new number of downloaded files of the online user.

RETURN:

None.
```

1.145 NC_BytesUpload 220

```
COMMAND: NC_BytesUpload     220

Change users number of uploaded bytes.

INPUT:

Data3    <- The new uploaded bytes of the online user.

RETURN:

None.
```

1.146 NC_BytesDownload 221

```
COMMAND: NC_BytesDownload   221

Change users number of downloaded bytes.

INPUT:

Data3    <- The new downloaded bytes of the online user.

RETURN:

None.
```

1.147 NC_DailyByteLimit 222

```
COMMAND: NC_DailyByteLimit  222

Change users daily byte limit.

INPUT:

Data3    <- The new daily byte limit of the online user.

RETURN:

None.
```

1.148 NC_DailyFileLimit 223

```
COMMAND: NC_DailyFileLimit  223

Change users daily file limit.

INPUT:

Data3    <- The new daily files limit of the online user.

RETURN:

None.
```

1.149 NC_DailyByteDld 224

```
COMMAND: NC_DailyByteDld    224

Change users daily bytes downloaded.

INPUT:

Data3    <- The new daily bytes downloaded of the online user.

RETURN:

None.
```

1.150 NC_DailyFileDld 225

```
COMMAND: NC_DailyFileDld    225

Change users daily files downloaded.

INPUT:
```

```
Data3    <- The new daily files downloaded by the online user.

RETURN:

None.
```

1.151 NC_DailyByteBonus 226

```
COMMAND: NC_DailyByteBonus  226

Change users daily byte bonus.

INPUT:

Data3    <- The new daily byte bonus of the online user.

RETURN:

None.
```

1.152 NC_DailyFileBonus 227

```
COMMAND: NC_DailyFileBonus  227

Change users daily file bonus.

INPUT:

Data3    <- The new daily file bonus of the online user.

RETURN:

None.
```

1.153 NC_Expert 228

```
COMMAND: NC_Expert          228

Change users Expert Mode.

INPUT:

Data1    <- The new expert mode of the online user.

RETURN:

None.
```

1.154 NC_NumLines 229

```
COMMAND: NC_NumLines        229

Change users number of lines to view.

INPUT:

Data1    <- The new number of lines of the online user.

RETURN:

None.

NOTE:

You have to set Data1 to one line less than the full linelength of the
screen, because of scrolling texts you need one line in reserve!
```

1.155 NC_Birthday 230

```
COMMAND: NC_Birthday        230

Change users birthday.

INPUT:

IOString <- The new Birthday of the online user (max. 10 chars).

RETURN:

None.
```

1.156 NC_MenuPrompt 231

```
COMMAND: NC_MenuPrompt      231

Change users Menuprompt.

INPUT:

IOString <- The new Menuprompt of the online user (max. 200 chars).

RETURN:

None.
```

1.157 NC_EditorType 232

```
COMMAND: NC_EditorType      232

Change users Editor type.

INPUT:

Data1    <- The new Editor type of the online user.

RETURN:

None.
```

1.158 NC_XferProt 233

```
COMMAND: NC_XferProt        233

Change users Transfer-Protocol.

INPUT:

Data1    <- The new XferProtocol of the online user.

RETURN:

None.
```

1.159 NC_Zoom 234

```
COMMAND: NC_Zoom            234

Change users ZOOM-Type.

INPUT:

Data1    <- The new ZOOM type of the online user.

RETURN:

None.
```

1.160 NC_SysLanguage 235

```
COMMAND: NC_SysLanguage     235

Change users current System language specifications.

INPUT:

IOString <- The new System language of the online user (max. 30 chars).
```

```
RETURN:

None.
```

1.161 NC_Language 236

```
COMMAND: NC_Language        236

Change users current language specifications.

INPUT:

Data1    <- The new language of the online user.

RETURN:

None.
```

1.162 NC_LineCount 237

```
COMMAND: NC_LineCount       237

Change users current number of lines viewed.

INPUT:

Data1    <- The new number of lines viewed.

RETURN:

None.
```

1.163 NC_AnsiColor 238

```
COMMAND: NC_AnsiColor       238

Change users ANSI-Flag.

INPUT:

Data1    <- The new ANSI flag of the online user.

RETURN:

None.
```

1.164 NC_SentBy 239

```
COMMAND: NC_SentBy          239

Change users Sent-By line.

INPUT:

Data1    <- The conferencenumber in which you want to change the SentByLine.
            If Data1 is lower than 1 or higher than the maximum conferences,
            it will be set to the current conference.
IOString <- The new Sent-By Line of the online user (max 45 chars).

RETURN:

None.
```

1.165 NC_AutoFileID 240

```
COMMAND: NC_AutoFileID      240

Change users Auto FILE_ID.DIZ.

INPUT:

Data1    <- The new Auto FILE_ID.DIZ of the online user.

RETURN:

None.
```

1.166 NC_NewMessage 241

```
COMMAND: NC_NewMessage      241

Change the New Message Status.

INPUT:

Data1    <- The new New Message Status of the online user.

RETURN:

None.
```

1.167 NC_Goodbye 242

```
COMMAND: NC_Goodbye         242
```

```
Change the (Fast) Goodbye flag.

INPUT:

Data1    <- The new Goodbye Flag of the online user.

RETURN:

None.
```

1.168 NC_ViewFlag 243

```
COMMAND: NC_ViewFlag        243

Change users Viewflag.

INPUT:

Data1    <- The new View Flag of the online user.

RETURN:

None.
```

1.169 NC_ZippyFlag 244

```
COMMAND: NC_ZippyFlag       244

Change users Zippyflag.

INPUT:

Data1    <- The new Zippy Flag of the online user.

RETURN:

None.
```

1.170 NC_ReplyMSGFlag 245

```
COMMAND: NC_ReplyMSGFlag    245

Change the ReplyMsg Flag.

INPUT:

Data1    <- The conferencenumber you are going to change the ReplyMSG Flag.
            If Data1 is lower than 1 or higher than the maximum conferences,
            it will be set to the current conference.
Data2    <- The new ReplyMSG Flag of the online user.
```

```
RETURN:

None.
```

1.171 NC_NukedFiles 246

```
COMMAND: NC_NukedFiles      246

Change users number of nuked Files.

INPUT:

Data3    <- The new nuked Files of the online user.

RETURN:

None.
```

1.172 NC_NukedBytes 247

```
COMMAND: NC_NukedBytes      247

Change users number of nuked Bytes.

INPUT:

Data3    <- The new nuked Bytes of the online user.

RETURN:

None.
```

1.173 NC_MinUpCPS 248

```
COMMAND: NC_MinUpCPS        248

Change users min. Upload CPS.

INPUT:

Data1    <- The new min. Upload CPS of the online user.

RETURN:

None.
```

1.174 NC_MinDownCPS 249

```
COMMAND: NC_MinDownCPS      249

Change users min. Download CPS.

INPUT:

Data1    <- The new Min Download CPS of the online user.

RETURN:

None.
```

1.175 NC_LostCarrier 250

```
COMMAND: NC_LostCarrier      250

Change users number of Lost Carriers.

INPUT:

 Data1    <- The new number of Lost Carriers of the online user.

RETURN:

None.
```

1.176 NC_NumberOfChats 251

```
COMMAND: NC_NumberOfChats   251

Change users number of chats.

INPUT:

Data3    <- The new number of chats.

RETURN:

None.
```

1.177 NC_NumberOfPages 252

```
COMMAND: NC_NumberOfPages   252

Change users number of pages.

INPUT:
```

```
Data1    <- The new number of pages.

RETURN:

None.
```

1.178 NC_NumberOfDayPages 253

```
COMMAND: NC_NumberOfDayPages 253

Change users number of daily pages.

INPUT:

Data1    <- The new number of day pages.

RETURN:

None.
```

1.179 NC_NumberOfDayRelogs 254

```
COMMAND: NC_NumberOfDayRelogs 254

Change the number of daily relogins allowed for users Accesslevel.

INPUT:

Data2    <- The new number of relogins allowed per day.

RETURN:

None.
```

1.180 NC_Nuked 255

```
COMMAND: NC_Nuked             255

Called from a Nuker to update the nuked bytes and files of the current user.

INPUT:

Data1    <- The number of files nuked.
Data3    <- The number of bytes nuked.

RETURN:

None.
```

```
NOTE:

Data1 and Data3 are files and bytes nuked by the nuker and they will be
added to the user.data.

This command will also call all doors found under the SYSCMD commands
NUKED and NUKED<x> !
```

1.181 NC_NukedAfter 256

```
                COMMAND: NC_NukedAfter        256

Called from a Nuker to update the nuked bytes and files of the current user,
but they will be updated not BEFORE the nuker has shut down.

INPUT:

Data1    <- The number of files nuked.
Data3    <- The number of bytes nuked.

RETURN:

None.

NOTE:

Data1 and Data3 are files and bytes nuked by the nuker and they will be
added to the user.data, but in difference to command
                NC_Nuked
                 the fields
won't be updated until the nuker has shut down!

This command will also call all doors found under the SYSCMD commands
NUKED and NUKED<x> !
```

1.182 CF_ShowText 400

```
COMMAND: CF_ShowText        400

Shows a Textfile without Suffix.

INPUT:

IOString <- the complete path and filename of the text to be viewed.

RETURN:

None.

NOTE:

You have to pass the complete path and Filename, incl. the suffix,
```

```
i.e.: BBS:BULL.TXT !
```

1.183 CF_ShowTextSuffix 401

```
Shows a Textfile with Suffix.

COMMAND: CF_ShowTextSuffix  401

INPUT:

IOString <- the complete path and filename of the text to be viewed.

RETURN:

None.

NOTE:

You have to pass the complete path & filename, but without Suffix!
I.e.: BBS:BULL could be 'BBS:BULL.TXT' or 'BBS:BULL.GER' etc.
```

1.184 CF_ShowTextSufLvl 402

```
COMMAND: CF_ShowTextSufLvl  402

Shows a Textfile with full suffix & level.

INPUT:

IOString <- the complete path and filename of the text to be viewed.

RETURN:

None.

NOTE:

You have to pass the complete path & filename, but without Suffix!
The level will be automatically searched and inserted for you.
I.e.: BBS:Bull could be 'BBS:BULL100.TXT' or 'BBS:BULL50.GER' etc.
```

1.185 CF_ExecuteCommand 403

```
COMMAND: CF_ExecuteCommand   403

Execute a menu command including doors if available.

INPUT:

IOString <- The command to be executed.
```

```
RETURN:

None.
```

1.186 CF_InternalCmd 404

```
COMMAND: CF_InternalCmd     404

Execute an internal menu command.

INPUT:

IOString <- The internal menu command to be executed.

RETURN:

None.
```

1.187 CF_ZModemSend 405

```
COMMAND: CF_ZModemSend      405

Send a file to the user via ZModem protocol.

INPUT:

IOString <- The File to be sent to the user.

RETURN:

Data2    -> Returns the status:

                 1 means file has been transfered.
                 0 means file has been copied to sysop download dir.
                -1 means xrzmodem.library not open.
```

1.188 CF_ZModemReceive 406

```
COMMAND: CF_ZModemReceive   406

Receive files via ZModem protocol.

INPUT:

IOString <- The path for the file(s) to be written to.

RETURN:

Data2    -> Returns the status:
```

```
                  1 means file has been transfered.
                  0 means file has been copied to sysop download dir.
                 -1 means xrzmodem.library not open.
NOTE:

You have to pass a valid path to IOString, else the transfer won't start !!!
```

1.189 CF_ZModemSendLst 407

```
COMMAND: CF_ZModemSendLst   407

Send files from the Flaglist to the user via ZModem protocol.

INPUT:

StringPtr <- A Stringpointer of nearly unlimited filenames incl. path.
             Between every single path/filename there must be a space.
             Maximum filelength of path+filename is 100 chars.
             Example: "RAM:File1.123 RAM:File2.123 RAM:ENV/TIME"

RETURN:

Data2    -> Returns the status:

                  1 means file has been transfered.
                  0 means file has been copied to sysop download dir.
                 -1 means xrzmodem.library not open.
```

1.190 CF_ReturnCommand 408

```
COMMAND: CF_ReturnCommand  408

Execute a command/door.

INPUT:

IOString <- The command to be executed.

RETURN:

None.

NOTE:

To delete a previous ReturnCommand-Request, overwrite it with a new request
or use "" to delete it completely.
```

1.191 CF_SetFlagFile 409

```
COMMAND: CF_SetFlagFile     409

Add file to the door-internal flaglist.

                    ! NOT IMPLEMENTED YET !
```

1.192 CF_GetFlagFile 410

```
COMMAND: CF_GetFlagFile     410

Get file frrom the door-internal flaglist.

                    ! NOT IMPLEMENTED YET !
```

1.193 CF_CallersLog 411

```
COMMAND: CF_CallersLog      411

Add a line of text to the callers.log of the current node.

INPUT:

IOString <- The string to add to the callers.log.

RETURN:

None.
```

1.194 CF_UDLog 412

```
COMMAND: CF_UDLog           412

Add a line of text to the ud.log of the current node.

INPUT:

IOString <- The string to add to the ud.log.

RETURN:

None.
```

1.195 CF_DoorLog 413

```
COMMAND: CF_DoorLog         413

Add a line of text to the Door.log of the current node.

INPUT:

IOString <- The string to add to the Door.log.

RETURN:

None.
```

1.196 CF_CompType 414

```
COMMAND: CF_CompType        414

Change Computertype with built-in selector.

INPUT:

Data1    <- Old or default computertype to load, 0 to let user select.

RETURN:

None.
```

1.197 CF_ModemType 415

```
COMMAND: CF_ModemType       415

Change Modemtype with built-in selector.

INPUT:

Data1    <- Old or default modemtype to load, 0 to let user select.

RETURN:

None.
```

1.198 CF_Language 417

```
COMMAND: CF_Language        417

Change Textlanguage (Suffix and language) with built-in selector.

INPUT:

Data1    <- Old or default language to load, 0 to let user select.
```

```
RETURN:

None.
```

1.199 CF_NumLines 418

```
COMMAND: CF_NumLines        418

Change Number of lines to view with built-in selector.

INPUT:

Data1    <- 0 = Built-In selector should appear with prompting for numbers,
            higher than 0 means set the given value without prompting.

RETURN:

None.
```

1.200 CF_ShTxtSufLvlCyc 419

```
COMMAND: CF_ShTxtSufLvlCyc  419

Shows a Textfile with full Suffix, level & Cycle.

INPUT:

Data1     <- 1 means use Suffix .TXT only. 0 means use users system language.
IOString  <- The path where the files to be shown are located.
StringPtr <- The filename itself, without path!

RETURN:

IOString  -> The complete filepath and name with cyclenumber and more...

EXAMPLE:

You pass :

IOString  = "BBS:SCREENS"
StringPtr = "Bull"
Data1     = 1

and maybe you get:

IOString  = "BBS:SCREENS/1-Bull.ger" or similar.

If no cycle texts could be found, FAME tries to use:

"BBS:Screens/Bull.ger"
```

```
If this file also doesn't exist, FAME tries to use:

"BBS:Screens/Bull.txt".
```

1.201 CF_ShowTextSetable 420

```
COMMAND: CF_ShowTextSetable 420

Shows a Textfile freely configurable.

INPUT:

Data1        <- 0 means don't cycle, else cycle.
Data2        <- 1 means use users language,
                0 means you have to set the suffix yourself.
Data3        <- 1 means use suffix .TXT only.
                0 means use users system language.
IOString     <- If Data1 is not 0 IOString is the path where the files to
                view are located, else IOString is the full path + filename.
StringPtr    <- The filename itself, without path!
StructDummy1 <- "-1" means only users suffix will be added as suffix.
                "-2" means .GR or if .GR not exists GR1 will be added as
                suffix. This is used for Paragon Door texts.

If the value of StructDummy1 is higher than 0 (Atol() ) i.e. it's 10, it
will be used as the Level, like : Menu10.txt.
If the value of StructDummy1 is 0 (Atol() ), the next lower level text will
be used, like: Menu300.txt or Menu255.txt. It counts down from 1000 to 0 in
steps of 5.

RETURN:

IOString     -> The complete filepath and name (with cyclenumber and more).

NOTE:

StructDummy1 is only available if Data2 is 1!
```

1.202 CF_InternReturnCmd 421

```
COMMAND: CF_InternReturnCmd 421

Execute an internal command.

INPUT:

IOString <- The command to be executed.

RETURN:

None.
```

```
NOTE:

To delete a previous ReturnCommand-Request, overwrite it with a new request
or use "" to delete it completely.
```

1.203 CF_DoCallersLog 422

```
COMMAND: CF_DoCallersLog 422

Write to any of the Logfiles.

INPUT:

Data1    <- Phase (type).
Data2    <- Log type.
Data3
IOString <- The string to be add to the <x>.log.

RETURN:

None.

 Log Types:

            0 = Callers.Log only
            1 = UDLog
            2 = StartLog
            3 = DoorLog

NOTE:

Callers.Log will be written in any case!
```

1.204 CF_SaveWherePhase 423

```
COMMAND: CF_SaveWherePhase  423

Save Where I am phase temporary.

INPUT:

None.

RETURN:

None.
```

1.205 CF_RestWherePhase 424

```
COMMAND: CF_RestWherePhase  424

Restore temporary saved Where I am phase.

INPUT:

None.

RETURN:

None.
```

1.206 CF_FileCopyMove 425

```
COMMAND: CF_FileCopyMove    425

Copy or move a file.

INPUT:

Data1    <- if 0 then copy, else move file.
IOString <- Source file to copy/move.

RETURN:

Data2    -> Returncode of operation. If false then there was an error.
```

1.207 CF_SysOpChat 426

```
                COMMAND: CF_SysOpChat       426

Jump into SysOp-Chat.

INPUT:

Data1    <- if 0 last line will be restored.
Data2    <- if TRUE shifted chat.
Data3    <- if TRUE no screen to front will be done, else it will.

RETURN:

None.

NOTE:

Data1 indicates (if = 0) that the last displayed string on screen before
the SysOp-Chat is called will be buffered and shall be re-displayed after
exiting the chat. Often useful for buffering input prompts.

Data2 emulates (if = TRUE) a chat like with SHIFT-F1 called.
If Data2 = FALSE it will be a chat like when using F1 only.
```

```
If the user is already in SysOp-Chat it's not possible to go in the SysOp-
Chat again. It's only possible if no SysOp-Chat is currently running.

If it was not possible to go into SysOp-Chat because the user is already
in a SysOp-Chat, this command return with the
                DoorPort-Error
                 1, which
means "Command not successful executed".
```

1.208 CF_SpecialCmd 427

```
COMMAND: CF_SpecialCmd      427

Execute a special door command (BBSCMD, SYSCMD or CONF<X>CMD).

INPUT:

Data1     <- Type of command to be executed (see below!).
IOString  <- The door command to be executed.
StringPtr <- The arguments for the door.

RETURN:

None.

NOTE:

                           Defines for Data1:

      >0 = CONF<Data1>CMD (i.e. Data1=2 will use CONF2CMD commands!)
      -1 = BBSCMD will be used.
      -2 = SYSCMD will be used.
```

1.209 CF_GetUserConfXS 428

```
                COMMAND: CF_GetUserConfXS   428

Get a string containing the conference access of a special user.

INPUT:

Data1    <- The usernumber of the user you want to get the conf access.

RETURN:

StringPtr -> A buffer filled with the conference access.

NOTE:

Check fdom_ReturnCode for a failure!!!
```

```
StringPtr will be an allocted buffer which will be freed from MainPart
after your door exists!

StringPtr contains the conference access of the user in the following form:

                                "X__X__X_X_"

The first char (byte 0) will be conference 1, the second (byte 1)
conference 2 and so on. There is *NO* relative conference here! The string
contains the physical form of your conferences from conference 1 till the
last conference available. You have to check yourself if the conference
is in the relative list if you require this. If you want to know anything
about the relative conferences, use
                NR_GetRelativeStatus
                 to check if the
relative conferences are activated, and use
                NR_AbsToRel
                 to get the relative
number of an absolute (real/physical) one.

For better understanding: The 'X' chars means that the user has access to
this conference, everything else means that the user has no access. Please
check only for 'X' and not for the '_' (underscores), as the 'X' is the only
real important thing to check.
```

1.210 SR_ConfName 600

```
COMMAND: SR_ConfName        600

Retrieve the conference name.

INPUT:

Data1    <- The conferenumber.
            If Data1 is lower than 1 or higher than the maximum conferences,
            it will be set to the current conference.

RETURN:

IOString -> The requested conference name.
```

1.211 SR_ConfLocation 601

```
COMMAND: SR_ConfLocation    601

Retrieve the conference location.

INPUT:

Data1    <- The conferenumber.
            If Data1 is lower than 1 or higher than the maximum conferences,
            it will be set to the current conference.
```

```
RETURN:

IOString -> The requested conference location.
```

1.212 SR_ConfNum 602

```
COMMAND: SR_ConfNum         602

Retrieve the current conference number.

INPUT:

None.

RETURN:

Data2    -> The actual conference number.
Data3    -> The relative conference number.

NOTE:

If the relative conference number is 0, then the user has no access to this
conference, however this should not normally happen here...
```

1.213 SR_BBSLocation 603

```
COMMAND: SR_BBSLocation     603

Retrieve the BBS Location.

INPUT:

None.

RETURN:

IOString -> The BBS Location.
```

1.214 SR_Status 604

```
COMMAND: SR_Status          604

Retrieve the current status of the node.

INPUT:

None.

RETURN:
```

```
IOString -> Contains the status.
```

1.215 SR_ScreenAdress 605

```
COMMAND: SR_ScreenAdress    605

Retrieve the screen address of the Console.

INPUT:

None.

RETURN:

IOString -> Contains the (hex) screen address.
```

1.216 SR_TaskPri 606

```
COMMAND: SR_TaskPri         606

Retrieve the priority the node is running at.

INPUT:

None.

RETURN:

Data2    -> Contains the Node priority.
```

1.217 SR_RawScreenAdress 607

```
COMMAND: SR_RawScreenAdress 607

Retrieve the raw screen address of the node.

INPUT:

None.

RETURN:

Data2    -> Contains the raw screen adress.
```

1.218 SR_FAMEVersion 608

```
COMMAND: SR_FAMEVersion     608

Retrieve the current Version string of FAME.

INPUT:

None.

RETURN:

Data1    -> Contains the version number of FAME.
Data2    -> Contains the revision number of FAME.
IOString -> Contains the version string of FAME.
```

1.219 SR_ChatSet 609

```
COMMAND: SR_ChatSet         609

Retrieve the Chatstatus.

INPUT:

None.

RETURN:

Data2    -> Contains the Chat flag status.
```

1.220 SR_ENVStat 610

```
COMMAND: SR_ENVStat         610

Retrieve the current ENV stat variable code.

INPUT:

None.

RETURN:

Data2    -> Contains the ENV stat variable code.
```

1.221 SR_NodeDevice 611

```
COMMAND: SR_NodeDevice      611

Retrieve the device name of the node.

INPUT:
```

None.

RETURN:

IOString -> Contains the device name.

1.222 SR_NodeUnit 612

COMMAND: SR_NodeUnit 612

Retrieve the device unit number of the node.

INPUT:

None.

RETURN:

Data2 -> Contains the unit number of the used device.

1.223 SR_NodeBaud 613

COMMAND: SR_NodeBaud 613

Retrieve the initialized Baudrate of the node.

INPUT:

None.

RETURN:

Data2 -> Contains the initial Baudrate of the node.

1.224 SR_NodeNumber 614

COMMAND: SR_NodeNumber 614

Retrieve the node number.

INPUT:

None.

RETURN:

Data2 -> Contains the node number of this node.

1.225 SR_MCI 615

```
COMMAND: SR_MCI             615

Send MCI-Text to FAME.

INPUT:

Data1    <- If non zero a return will be sent at the end.
IOString <- The MCI-string to be interpreted by FAME.

RETURN:

Data2    -> The Returnvalue. 0 if everything is ok, non zero means
            there was an error.
```

1.226 SR_GetTask 616

```
COMMAND: SR_GetTask         616

Finds current node's task address.

INPUT:

None.

RETURN:

StructDummy1 -> Is the (struct Process *) of this Node.
```

1.227 SR_NodeBaudRate 617

```
COMMAND: SR_NodeBaudRate    617

Retrieve users current connect rate.

INPUT:

None.

RETURN:

Data2    -> Contains the connect rate.
```

1.228 SR_LogonType 618

```
COMMAND: SR_LogonType       618

Retrieve the Logontype.
```

```
INPUT:

None.

RETURN:

Data2    -> Contains the Logontype.

Possible values are:

                    0 = Local/Sysop Login
```

1.229 SR_ScrLeft 619

```
COMMAND: SR_ScrLeft         619

Retrieve the screen coordinates.

INPUT:

None.

RETURN:

Data2    -> Contains the coordinate of the left edge of screen.
```

1.230 SR_ScrTop 620

```
COMMAND: SR_ScrTop          620

Retrieve the screen coordinates.

INPUT:

None.

RETURN:

Data2    -> Contains the coordinate of the Top edge of screen.
```

1.231 SR_ScrWidth 621

```
COMMAND: SR_ScrWidth        621

Retrieve the screen coordinates.

INPUT:

None.
```

```
RETURN:

Data2    -> Contains the width of the screen.
```

1.232 SR_ScrHeight 622

```
COMMAND: SR_ScrHeight       622

Retrieve the screen coordinates.

INPUT:

None.

RETURN:

Data2    -> Contains the height of the screen.
```

1.233 SR_PurgeLine 623

```
COMMAND: SR_PurgeLine       623

Abort Serial input.

INPUT:

None.

RETURN:

None.
```

1.234 SR_NonStopText 624

```
COMMAND: SR_NonStopText     624

Retrieve the NONSTOP scrolling text flag.

INPUT:

None.

RETURN:

Data2    -> Contains the nonstop flag.
```

1.235 SR_GoodFile 625

```
COMMAND: SR_GoodFile        625

Retrieve the result of a tested file after upload.

INPUT:

None.

RETURN:

Data2    -> Contains the filestatus.
```

1.236 SR_LastAccountNum 626

```
COMMAND: SR_LastAccountNum  626

Retrieve the Last Account Number.

INPUT:

None.

RETURN:

Data2    -> Contains the number of the last available account.
```

1.237 SR_PurgeLineStart 627

```
COMMAND: SR_PurgeLineStart  627

Begin serial input abort.

INPUT:

None.

RETURN:

None.
```

1.238 SR_PurgeLineEnd 628

```
COMMAND: SR_PurgeLineEnd    628

End serial input abort.

INPUT:
```

None.

RETURN:

None.

1.239 SR_BBSOrigin 629

COMMAND: SR_BBSOrigin 629

Retrieve the BBS Origin.

INPUT:

None.

RETURN:

IOString -> Contains the BBS Origin.

1.240 SR_DefLineLen 630

COMMAND: SR_DefLineLen 630

Retrieve the default number of textlines.

INPUT:

None.

RETURN:

Data2 -> Contains the default number of textlines.

1.241 SR_NumberOfNodes 631

COMMAND: SR_NumberOfNodes 631

Retrieve the number of Nodes.

INPUT:

None.

RETURN:

Data2 -> Contains the number of Nodes.

1.242 SR_FileDescUndLine 632

```
COMMAND: SR_FileDescUndLine 632

Retrieve the Filedescription underline.

INPUT:

None.

RETURN:

IOString -> Contains the Description underline.
```

1.243 SR_NumberOfConfs 633

```
COMMAND: SR_NumberOfConfs   633

Retrieve the number of Conferences.

INPUT:

None.

RETURN:

Data2    -> Contains the number of Conferences.
```

1.244 SR_ConfNameLoc 634

```
COMMAND: SR_ConfNameLoc     634

Retrieve the conference name and location.

INPUT:

Data1    <- The confnumber you are interested in.
            If Data1 is lower than 1 or higher than the maximum conferences,
            it will be set to the current conference.

RETURN:

StructDummy1 -> The requested conference name.
StructDummy2 -> The requested conference location.
```

1.245 SR_FAMEDataFileVers 635

```
COMMAND: SR_FAMEDataFileVers 635
```

```
Retrieve the current Datafile Versionstring of FAME.

INPUT:

None.

RETURN:

Data2    -> Contains the versionnumber used to check for fileheaders.
Data3    -> Contains the revisionnumber used to check for fileheaders.
IOString -> Contains the Datafile version string of FAME.
```

1.246 SR_ConnectString 636

```
COMMAND: SR_ConnectString    636

Retrieve the current users connect string.

INPUT:

None.

RETURN:

IOString -> Contains the connect string.
```

1.247 SC_ChatSet 702

```
COMMAND: SC_ChatSet          702

Change the Chat status.

INPUT:

Data1    <- 0 means chat off, else chat on.

RETURN:

None.
```

1.248 SC_ENVStat 703

```
COMMAND: SC_ENVStat          703

Change the current ENV stat variable code.

INPUT:

Data1    <- the new ENV stat variable code.
```

```
RETURN:

None.
```

1.249 SC_NonStopText 704

```
COMMAND: SC_NonStopText     704

Change the NONSTOP scrolling textflag.

INPUT:

Data1    <- 0 means NONSTOP text is off, else NONSTOP text is on.

RETURN:

None.
```

1.250 SC_DefLineLen 705

```
COMMAND: SC_DefLineLen     705

Change the default number of textlines.

INPUT:

Data1    <- The new default number of textlines.

RETURN:

None.
```

1.251 SC_FileDescUndLine 706

```
COMMAND: SC_FileDescUndLine 706

Change the Filedescription underline.

INPUT:

None.

RETURN:

IOString <- The new File Description underline (max. 45 chars).
```

1.252 SC_GoodFile 707

```
COMMAND: SC_GoodFile        707

Set the result of a tested file after upload.

INPUT:

Data1    <- The new filestatus.

RETURN:

None.

NOTE:

Data1 Values:

-1 means the file failed the file check.
 0 means the file passed the file check.
 1 means the file was not checked.
```

1.253 AR_GetKey 800

```
COMMAND: AR_GetKey          800

Check for a keypress without waiting for it.

INPUT:

None.

RETURN:

Data2    -> 0 means nothing was pressed, 1 means a key was pressed.
Data3    -> Where the char was typed: 0 = from Console, 1 = from Serial.
```

1.254 AR_WaitRAWChar 801

```
               COMMAND: AR_WaitRAWChar     801

Gets a char with waiting for it, also get ALL REAL RAW keys!

INPUT:

IOString <- String will be displayed to the user.

RETURN:

Data2    -> The char typed by the user.
Data3    -> Where the char was typed: 0 = from Console, 1 = from Serial.
```

```
NOTE:

You have to check for FKey/Shift-FKey/Helpkey Rawkey codes yourself. If such
a Rawkeycode appears, you have to use the
                AR_EmulateFKeyHelp
                 Command.
```

1.255 AR_EmulateFKeyHelp 802

```
COMMAND: AR_EmulateFKeyHelp 802

Emulates a pressed FKey/Shift-FKey/Helpkey on console.

INPUT:

Data1    <- The keytype.

RETURN:

None.

NOTE:

Data1-Values are:

 1-10 = F1-F10
11-20 = Shift-F1 - Shift-F10.
   21 = Helpkey.

If ReturnCode is 1, the node is iconified and this command won't be executed.
This is needed for more security, as this function only emulates the keys
pressed by the SysOp and should not be used for other things...
```

1.256 AR_GetCharHex 803

```
COMMAND: AR_GetCharHex      803


                   ! NOT IMPLEMENTED YET !
```

1.257 AR_EditFile 804

```
COMMAND: AR_EditFile        804

Edit a File with the internal Texteditor.

INPUT:

IOString <- The file to be edited.
```

```
RETURN:

Data2    -> Returncode: 0 successful, -2 aborted by user.
```

1.258 AR_Dump 805

```
COMMAND: AR_Dump            805

Dump the user's data-structure to a specified file.

INPUT:

IOString <- The filename of the generated user structure.

RETURN:

None.
```

1.259 AR_UserStatus 806

```
COMMAND: AR_UserStatus      806

Retrieve the User's status (Active/Deleted/Inactive).

INPUT:

Data2    -> The user status flag.

RETURN:

None.
```

1.260 AR_NewScan 807

```
COMMAND: AR_NewScan         807

Retrieve the Newscan Flag.

INPUT:

Data1    <- The confnumber from which you want to get the NewScan flag.
            If Data1 lower than 1 or higher than the maximum conferences,
            it will be set to the current conference.

RETURN:

Data2    -> 1 if NewScan is on, 0 if off.
```

1.261 AR_Hacks 808

```
COMMAND: AR_Hacks           808

Retrieve the Hackattempts of the user.

INPUT:

None.

RETURN:

Data2    -> Contains the User's number of hacks.
```

1.262 AR_LastConf 809

```
COMMAND: AR_LastConf        809

Retrieve the Last conference which was joined by the User.

INPUT:

None.

RETURN:

Data3    -> Contains the User's last conference on.
```

1.263 AR_ConfReJoin 810

```
COMMAND: AR_ConfReJoin      810

Retrieve the conference which should be joined at every logon.

INPUT:

None.

RETURN:

Data3    -> Contains the Auto-Rejoin conference of the user.
```

1.264 AR_HighUpCPS 811

```
COMMAND: AR_HighUpCPS       811

Retrieve the highest CPS during uploads.

INPUT:
```

None.

RETURN:

Data2 -> Contains the User’s highest CPS during Uploads.

1.265 AR_HighDownCPS 812

COMMAND: AR_HighDownCPS 812

Retrieve the highest CPS during Downloads.

INPUT:

None.

RETURN:

Data2 -> Contains the User’s highest CPS during Downloads.

1.266 AR_Baud 813

COMMAND: AR_Baud 813

Retrieve the Baudrate of the last Logon.

INPUT:

None.

RETURN:

Data2 -> Contains the Baudrate of the User’s last Logon.

1.267 AR_MsgCLS 814

COMMAND: AR_MsgCLS 814

Retrieve the ClearScreen-Setting on user’s Messageflag.

INPUT:

None.

RETURN:

Data2 -> Contains the ClearScreen-Setting on user’s Messageflag.

1.268 AR_FileCLS 815

```
COMMAND: AR_FileCLS         815

Retrieve the ClearScreen-Setting on user's Filelist-Flag.

INPUT:

None.

RETURN:

Data2    -> Contains the ClearScreen-Setting on Filelist-Flag.
```

1.269 AR_UGlobal 816

```
COMMAND: AR_UGlobal         816

Retrieve the Global Uploadflag.

INPUT:

None.

RETURN:

Data2    -> Contains the User's Global uploadflag.
```

1.270 AR_DGlobal 817

```
COMMAND: AR_DGlobal         817

Retrieve the Global Downloadflag.

INPUT:

None.

RETURN:

Data2    -> Contains the User's Global downloadflag.
```

1.271 AR_FileBase 818

```
COMMAND: AR_FileBase        818

Retrieve the access to the User File Base.

INPUT:
```

None.

RETURN:

Data2 -> Contains the User's access to the User File Base.

1.272 AR_Hide 819

COMMAND: AR_Hide 819

Retrieve users hide flag.

INPUT:

None.

RETURN:

Data2 -> Contains the User's hide flag.

1.273 AR_MsgRooming 820

COMMAND: AR_MsgRooming 820

Retrieve the Messagerooming flag.

INPUT:

None.

RETURN:

Data2 -> Contains the User's Messagerooming flag.

1.274 AR_StringEdit 821

COMMAND: AR_StringEdit 821

Retrieve the StringEdit flag.

INPUT:

None.

RETURN:

Data2 -> Contains the User's StringEdit flag.

1.275 AR_CryptPW 822

```
COMMAND: AR_CryptPW         822

Retrieve the CryptPW.

INPUT:

None.

RETURN:

IOString -> Contains User's CryptPW.
```

1.276 AR_CryptFlag 823

```
COMMAND: AR_CryptFlag       823

Retrieve the Cryptflag.

INPUT:

None.

RETURN:

Data2    -> Contains User's Crypt flag.
```

1.277 AR_UserFileBPW 824

```
COMMAND: AR_UserFileBPW     824

Retrieve UserFileBase Password.

INPUT:

None.

RETURN:

IOString -> Contains User's UserFileBasePW.
```

1.278 AR_DropDTR 825

```
COMMAND: AR_DropDTR         825

Drop Carrier on an user.

INPUT:
```

None.

RETURN:

None.

1.279 AR_Userlimit 826

COMMAND: AR_Userlimit 826

Retrieve the UserLimit of the BBS.

INPUT:

None.

RETURN:

Data3 -> Contains BBS Userlimit.

1.280 AR_MaxNameFailure 827

COMMAND: AR_MaxNameFailure 827

Retrieve the Maximum number of failures allowed when entering a name.

INPUT:

None.

RETURN:

Data2 -> Contains BBS MaxNameFailure.

1.281 AR_MaxUserPWFail 828

COMMAND: AR_MaxUserPWFail 828

Retrieve the maximum number of failures allowed when entering a UserPassword.

INPUT:

None.

RETURN:

Data2 -> Contains the BBS MaxUserPWFail.

1.282 AR_MaxSysPWFailure 829

```
COMMAND: AR_MaxSysPWFailure 829

Retrieve the maximum number of SystemPassword failures allowed.

INPUT:

None.

RETURN:

Data2    -> Contains the BBS MaxSysPWFailure.
```

1.283 AR_MaxNUPFailure 830

```
COMMAND: AR_MaxNUPFailure   830

Retrieve the maximum number of NewUser Password failures allowed.

INPUT:

None.

RETURN:

Data2    -> Contains the BBS MaxNUPFailure.
```

1.284 AR_MaxUserHacks 831

```
COMMAND: AR_MaxUserHacks    831

Retrieve the maximum number of BBS Userhacks allowed.

INPUT:

None.

RETURN:

Data2    -> Contains the BBS MaxUserHacks.
```

1.285 AR_ScreensPath 832

```
COMMAND: AR_ScreensPath     832

Retrieve the Screens path of the BBS.

INPUT:
```

None.

RETURN:

IOString -> Contains the BBS ScreensPath.

1.286 AR_SysPWPrompt 833

COMMAND: AR_SysPWPrompt 833

Retrieve the SystemPassword prompt of the BBS.

INPUT:

None.

RETURN:

IOString -> Contains the BBS SysPWPrompt.

1.287 AR_NewUserPWPrompt 834

COMMAND: AR_NewUserPWPrompt 834

Retrieve the NewUserPassword prompt of the BBS.

INPUT:

None.

RETURN:

IOString -> Contains the BBS NewUserPWPrompt.

1.288 AR_UsernamePrompt 835

COMMAND: AR_UsernamePrompt 835

Retrieve the UserName prompt of the BBS.

INPUT:

None.

RETURN:

IOString -> Contains the BBS UsernamePrompt.

1.289 AR_UserPWPrompt 836

```
COMMAND: AR_UserPWPrompt    836

Retrieve the UserPassword prompt of the BBS.

INPUT:

None.

RETURN:

IOString -> Contains the BBS UserPWPrompt.
```

1.290 AR_PausePrompt 837

```
COMMAND: AR_PausePrompt     837

Retrieve the Pauseprompt.

INPUT:

None.

RETURN:

IOString -> Contains the BBS PausePrompt.
```

1.291 AR_SysOpChatColor 838

```
COMMAND: AR_SysOpChatColor  838

Retrieve the Sysop-Chat color.

INPUT:

None.

RETURN:

IOString -> Contains the BBS SysOpChatColor.
```

1.292 AR_UserChatColor 839

```
COMMAND: AR_UserChatColor   839

Retrieve the User-Chat color.

INPUT:
```

None.

RETURN:

IOString -> Contains the BBS UserChatColor.

1.293 AR_UploadPathI 840

COMMAND: AR_UploadPathI 840

Retrieve the UploadPathI of the actual Conference.

INPUT:

None.

RETURN:

IOString -> Contains the UploadPathI of the actual conference.

1.294 AR_DownloadPathI 841

COMMAND: AR_DownloadPathI 841

Retrieve the DownloadPathI of the actual conference.

INPUT:

None.

RETURN:

IOString -> Contains the DownloadPathI of the actual conference.

1.295 AR_AdditioUlPaths 842

COMMAND: AR_AdditioUlPaths 842

Retrieve the Additional Upload Paths.

INPUT:

None.

RETURN:

IOString -> Contains the AdditionalUlPaths of the actual conference.

1.296 AR_AdditioDlPaths 843

```
COMMAND: AR_AdditioDlPaths 843

Retrieve the Additional Download Paths.

INPUT:

None.

RETURN:

IOString -> Contains the AdditionalDlPaths of the actual conference.
```

1.297 AR_NumberofDirs 844

```
COMMAND: AR_NumberofDirs    844

Retrieve the Number of Dirs.

INPUT:

Data1    <- The conference number.

RETURN:

Data2    -> Contains the number of dirs in the conference.

NOTE:

If Data1 is < 1 or higher than the maximum available conference number,
you will recieve the Number of Dirs for the current conference!
```

1.298 AR_GiveUlBytes 845

```
COMMAND: AR_GiveUlBytes     845

Retrieve the GiveUlBytes.

INPUT:

None.

RETURN:

Data2    -> Contains the number of ULBytes given for the actual conference.
```

1.299 AR_TakeDlBytes 846

```
COMMAND: AR_TakeDlBytes     846

Retrieve the TakeDlBytes.

INPUT:

None.

RETURN:

Data2    -> Contains the number of ULBytes taken for the actual conference.
```

1.300 AR_FlagFilePath 847

```
COMMAND: AR_FlagFilePath    847

Add a file on a Conference Fileflaglist with or without path.

INPUT:

Data1     <- Conferencenumber or if < 1 the actual conference will be used.
             If Data1 is lower than 1 or higher than the maximum conferences,
             it will be set to the current conference.
Data2     <- FreeDl Flag. If <> 0 it will be a FreeDownloadfile.
IOString  <- The Filename (without Path!).
StringPtr <- The Path for the file, "" if no path is known and the BBS
             should find the Path.

RETURN:

Data3     -> Errormessage. 0 on success.

NOTE:

You can also flag filenames longer than 12 Chars. Filenames can be up to
31 chars long, the Pathname is limited to 101 chars.

Data3 Returnvalues:

       0 -> successfully flagged
       1 -> file is already flagged
       2 -> length of IOString is less than 1 Byte, i.e. a Null-String.
      -1 -> list element memory allocation failed.
      -2 -> given conference not found. normally won't happen here, because
            of the Data1 check.
```

1.301 AR_StringToNode 848

```
                COMMAND: AR_StringToNode    848

Send a Textstring to a Node (depends on the users settings in "W 18").
```

```
INPUT:

Data1    <- The Node which the String should be sent to.
IOString <- The String to be sent to the Node.

RETURN:

None.

NOTE:

If ReturnCode=1, the String wasn't send because the selected node doesn't
exist ! To avoid this, use
                NR_ActiveNode
                 to check if selected node exists.

AR_StringToNode depends on users settings under command "W 18". This means,
that the User is able to disable the recieving of messages to him at several
actions. If you sent it to an user, and he has disabled it in his current
status (Reading Mail,etc.), the String you've sent will be added to a list
which will be displayed when the user comes to a position where he will or
must recieve the messages. All users *MUST* recieve the messages in idle
mode (Menuprompt).
```

1.302 AR_GetNodeEnv 849

```
COMMAND: AR_GetNodeEnv      849

Retrieve the Status of a Node.

INPUT:

Data1    <- The Node you want the status of.

RETURN:

Data2    -> The status of the Node. Use this one to get a real result!
Data3    -> The full last Node to Server Command, (it's best to not use this)
IOString -> The Actionstring displayed at the Server.
```

1.303 AR_FullEdStruct 850

```
COMMAND: AR_FullEdStruct    850

Retrieve the (filled) struct for external Message-Editor.

INPUT:

None.

RETURN:
```

```
StructDummy1 -> Pointer to struct ExternEditor.
```

1.304 AR_SendStr 851

```
COMMAND: AR_SendStr         851

Sends an unlimited length String to the user.

INPUT:

Data1     <- if <> 0 a "CR/LF" combination will be sent after the string
             has been sent.
StringPtr <- The string to be sent to the user.

RETURN:

None.
```

1.305 AR_SendStrCon 852

```
COMMAND: AR_SendStrCon      852

Sends an unlimited length String to the console.

INPUT:

Data1     <- if <> 0 a "CR/LF" combination will be sent after the string
             has been sent.
StringPtr <- The string to be sent to the console.

RETURN:

None.
```

1.306 AR_SendStrSer 853

```
COMMAND: AR_SendStrSer      853

Sends an unlimited length String to the serial.

INPUT:

Data1     <- if <> 0 a "CR/LF" combination will be sent after the string
             has been sent.
StringPtr <- The string to be sent to the serial.

RETURN:

None.
```

1.307 AR_ResetNumFlags 854

```
                COMMAND: AR_ResetNumFlags   854

Reset NumberFlagList.

INPUT:

None.

RETURN:

Data3   -> Returncode.

NOTE:

Data3 must be TRUE (<>0) or an internal error has been detected. In this
case you don't have to use
                AR_SetNumFlag
                 and
                AR_GetNumFlag
                 !!!
```

1.308 AR_SetNumFlag 855

```
COMMAND: AR_SetNumFlag      855

Flag an entry in the NumberFlagList.

INPUT:

Data1    <- The number of the new entry. It's up to you to count correctly!
IOString <- The entry (FileName).

RETURN:

Data3    -> Returncode.

NOTE:

Data3 Returnvalues:

        0 -> successfully flagged.
       -1 -> memory allocation failed.
       -2 -> Base not initialized.
```

1.309 AR_GetNumFlag 856

```
COMMAND: AR_GetNumFlag      856

Get an entry from the NumberFlagList.
```

```
INPUT:

Data1    <- The number of the entry you want to have.

RETURN:

Data3    -> Returncode. See below...
IOString -> The entry (FileName).

NOTE:

Data3 Returnvalues:

        0 -> successfully flagged.
       -1 -> entry number not found.
       -2 -> Base not initialized.
```

1.310 AR_NodeVersion 857

```
COMMAND: AR_NodeVersion     857

Get the version and revision of a specific node.

INPUT:

Data1    <- The Node number of your choice.

RETURN:

Data2    -> The version of this Node.
Data3    -> The revision of this Node.
```

1.311 AR_NodeStartTime 858

```
COMMAND: AR_NodeStartTime   858

Get the start time of a specific Node.

INPUT:

Data1    <- The Node number

RETURN:

Data2    -> The start time/date (UNIX) of this Node.
IOString -> The start time/date (String) of this Node.
```

1.312 AR_ServerVersion 859

```
COMMAND: AR_ServerVersion   859

Get the version and revision of the Server.

INPUT:

None.

RETURN:

Data2    -> The version of the Server.
Data3    -> The revision of the Server.
```

1.313 AR_ServerStartTime 860

```
COMMAND: AR_ServerStartTime 860

Get the start time of the Server.

INPUT:

None.

RETURN:

Data2    -> The start time/date (UNIX) of the Server.
IOString -> The start time/date (String) of the Server.
```

1.314 AR_HotKey 861

```
                COMMAND: AR_HotKey          861

Gets a char without waiting for it, gets also self definable cursor keys.

INPUT:

Data1    <- Cursor key begining return code. (If 0 they begin with 2)

RETURN:

Data2    -> The char typed by the user.
Data3    -> Where the char was typed: 0 = from Console, 1 = from Serial.

NOTE:

AR_Hotkey is the same as
                NR_HotKey
                , but like
                NR_WaitChar
                 it returns also
cursor keys. Beware, the cursor key returns are maybe others than with the
```

```
command NR_WaitChar !!!!

You get all ASCII codes, also RAW codes, only the CSI codes and escape
sequences will be filtered, because the BBS has to check for Function keys
and more.

You will get the cursorkeys as the following values:

                            Data1 + 2 for UP
                            Data1 + 3 for DOWN
                            Data1 + 1 for RIGHT
                            Data1     for LEFT

                                Definition:

                                 LEFT  = 0
                                 RIGHT = 1
                                 DOWN  = 2
                                 UP    = 3

You now have to add the defined values above to the returncode from Data2
to checkout if the result is a cursor key.

If you specify that LEFT (the first cursor key) begins with value 160 (set
Data1 to 160), the cursorkeys will be the following ones:


                            CURSOR LEFT  = 160
                            CURSOR RIGHT = 161
                            CURSOR UP    = 163
                            CURSOR DOWN  = 162
```

1.315 AR_DropDtrOnNode 862

```
COMMAND: AR_DropDtrOnNode   862

Drop the Carrier on a specific node.

INPUT:

Data1    <- The Node where the carrier shall be droped.

RETURN:

Nothing.
```

1.316 AC_UserStatus 900

```
COMMAND: AC_UserStatus      900

Change the Userstatus (Active/Deleted/Inactive).
```

```
INPUT:

Data1    <- The new Userstatus.

RETURN:

None.
```

1.317 AC_NewScan 901

```
COMMAND: AC_NewScan         901

Change the NewScan Flag.

INPUT:

Data1    <- The conferencenumber for which you want to set the NewScan flag.
Data2    <- The new NewScan Flag. 1 = On, 0 = Off.

RETURN:

None.
```

1.318 AC_HackCount 902

```
COMMAND: AC_HackCount       902

Change the Hackcounter.

INPUT:

Data1    <- The new hackcounter.

RETURN:

None.
```

1.319 AC_LastConf 903

```
COMMAND: AC_LastConf        903

Change the last conference which was joined.

INPUT:

Data3    <- The new last conf joined.

RETURN:

None.
```

1.320 AC_ConfReJoin 904

```
COMMAND: AC_ConfReJoin      904

Change the Conf which should be joined everytime at logon.

INPUT:

Data3    <- The new conf which should be joined everytime at logon.

RETURN:

None.
```

1.321 AC_HighUpCPS 905

```
COMMAND: AC_HighUpCPS       905

Change the highest CPS on Uploads.

INPUT:

Data1    <- The new highest CPS on Uploads.

RETURN:

None.
```

1.322 AC_HighDownCPS 906

```
COMMAND: AC_HighDownCPS     906

Change the highest CPS on Downloads.

INPUT:

Data1    <- The new highest CPS on Downloads.

RETURN:

None.
```

1.323 AC_Baud 907

```
COMMAND: AC_Baud            907

Change the Baudrate of the last logon.

INPUT:
```

```
Data1    <- The new Baudrate of the last Logon.

RETURN:

None.
```

1.324 AC_MsgCLS 908

```
COMMAND: AC_MsgCLS          908

Change the ClearScreen on Message flag.

INPUT:

Data1    <- The new ClearScreen on Msg's flag.

RETURN:

None.
```

1.325 AC_FileCLS 909

```
COMMAND: AC_FileCLS         909

Change the ClearScreen on Filelist flag.

INPUT:

Data1    <- The new ClearScreen on Filelist flag.

RETURN:

None.
```

1.326 AC_UGlobal 910

```
COMMAND: AC_UGlobal         910

Change the global upload flag.

INPUT:

Data1    <- The new global upload flag.

RETURN:

None.
```

1.327 AC_DGlobal 911

```
COMMAND: AC_DGlobal         911

Change the global download flag.

INPUT:

Data1    <- The new global download flag.

RETURN:

None.
```

1.328 AC_FileBase 912

```
COMMAND: AC_FileBase        912

Change the access to the User File Base.

INPUT:

Data1    <- The new access to the User File Base.

RETURN:

None.
```

1.329 AC_Hide 913

```
COMMAND: AC_Hide            913

Change the Hide flag.

INPUT:

Data1    <- The new Hide flag.

RETURN:

None.
```

1.330 AC_MsgRooming 914

```
COMMAND: AC_MsgRooming      914

Change the Message-Rooming Flag.

INPUT:
```

```
Data1    <- The new Msgrooming flag.

RETURN:

None.
```

1.331 AC_StringEdit 915

```
COMMAND: AC_StringEdit      915

Change the StringEdit flag.

INPUT:

Data1    <- The new StringEdit flag.

RETURN:

None.
```

1.332 AC_CryptPW 916

```
COMMAND: AC_CryptPW         916

Change the CryptPW.

INPUT:

IOString <- The new CryptPW (max 11 chars).

RETURN:

None.
```

1.333 AC_CryptFlag 917

```
COMMAND: AC_CryptFlag       917

Change the CryptFlag.

INPUT:

Data1    <- The new CryptFlag.

RETURN:

None.
```

1.334 AC_UserFileBPW 918

```
COMMAND: AC_UserFileBPW     918

Change User's File Base PW.

INPUT:

IOString <- The new User File Base PW.

RETURN:

None.
```

1.335 AC_Userlimit 919

```
COMMAND: AC_Userlimit       919

Change the Userlimit of the BBS.

INPUT:

Data3    <- The new BBS Userlimit.

RETURN:

None.
```

1.336 AC_MaxNameFailure 920

```
COMMAND: AC_MaxNameFailure  920

Change the max. Name failures.

INPUT:

Data1    <- The new BBS MaxNameFailure.

RETURN:

None.
```

1.337 AC_MaxUserPWFail 921

```
COMMAND: AC_MaxUserPWFail   921

Change the max. User Password failures.

INPUT:
```

```
Data1    <- The new BBS MaxUserPWFailure.

RETURN:

None.
```

1.338 AC_MaxSysPWFailure 922

```
COMMAND: AC_MaxSysPWFailure 922

Change the max. System-Password failures.

INPUT:

Data1    <- The new BBS MaxSysPWFailure.

RETURN:

None.
```

1.339 AC_MaxNUPFailure 923

```
COMMAND: AC_MaxNUPFailure   923

Change the max. NewUser-Password failures.

INPUT:

Data1    <- The new BBS MaxNUPFailure.

RETURN:

None.
```

1.340 AC_MaxUserHacks 924

```
COMMAND: AC_MaxUserHacks    924

Change max. allowed user hacks.

INPUT:

Data1    <- The new BBS MaxUserHacks.

RETURN:

None.
```

1.341 AC_ScreensPath 925

```
COMMAND: AC_ScreensPath     925

Change the ScreensPath.

INPUT:

IOString <- The new BBS ScreensPath.

RETURN:

None.
```

1.342 AC_SysPWPrompt 926

```
COMMAND: AC_SysPWPrompt     926

Change the System-Password prompt.

INPUT:

IOString <- The new BBS SysPWPrompt.

RETURN:

None.
```

1.343 AC_NewUserPWPrompt 927

```
COMMAND: AC_NewUserPWPrompt 927

Change the NewUser-Password prompt.

INPUT:

IOString <- The new BBS NewUserPWPrompt.

RETURN:

None.
```

1.344 AC_UsernamePrompt 928

```
COMMAND: AC_UsernamePrompt  928

Change the Username prompt.

INPUT:
```

```
IOString <- The new BBS UsernamePrompt.

RETURN:

None.
```

1.345 AC_UserPWPrompt 929

```
COMMAND: AC_UserPWPrompt    929

Change the User-Password prompt.

INPUT:

IOString <- The new BBS UserPWPrompt.

RETURN:

None.
```

1.346 AC_PausePrompt 930

```
COMMAND: AC_PausePrompt     930

Change the Pause prompt.

INPUT:

IOString <- The new BBS PausePrompt.

RETURN:

None.
```

1.347 AC_SysOpChatColor 931

```
COMMAND: AC_SysOpChatColor  931

Change the SysopChat color.

INPUT:

IOString <- The new BBS SysOpChatColor.

RETURN:

None.
```

1.348 AC_UserChatColor 932

```
COMMAND: AC_UserChatColor   932

Change users SysopChat color.

INPUT:

IOString <- The new BBS UserChatColor.

RETURN:

None.
```

1.349 AC_NumberofDirs 933

```
COMMAND: AC_NumberofDirs    933

Change the Number of dirs. For the current conference.

INPUT:

Data1    <- The new BBS NumberofDirs.

RETURN:

None.
```

1.350 AC_GiveUlBytes 934

```
COMMAND: AC_GiveUlBytes     934

Change the GiveUlBytes. For the current conference.

INPUT:

Data1    <- The new Ul bytes given by the BBS.

RETURN:

None.
```

1.351 AC_TakeDlBytes 935

```
COMMAND: AC_TakeDlBytes     935

Change the TakeDlBytes. For the current conference.

INPUT:
```

```
Data1    <- The new Dl bytes taken by the BBS.

RETURN:

None.
```

1.352 AC_ServerAction 936

```
COMMAND: AC_ServerAction    936

Write action String to the Server nodeline.

INPUT:

IOString <- The string to be written to server's action field.

RETURN:

None.
```

1.353 AC_ServerActionCPS 937

```
COMMAND: AC_ServerActionCPS 937

Write action string and Baud/CPS to the server nodeline.

INPUT:

Data1    <- The Baud/CPS to the server.
IOString <- The string to the server.

RETURN:

None.
```

1.354 RD_ShowFlags 1000

```
COMMAND: RD_ShowFlags       1000

Shows the Flaglist of the current Conference.

                    ! NOT IMPLEMENTED YET !
```

1.355 RD_ShowAllFlags 1001

COMMAND: RD_ShowAllFlags   1001

Shows the Flaglists of all Conferences.

                        ! NOT IMPLEMENTED YET !

1.356 RD_ConfigLoad 1002

COMMAND: RD_ConfigLoad   1002

Reloads all Configdatas.

                        ! NOT IMPLEMENTED YET !

1.357 RD_IconifyIs 1003

COMMAND: RD_IconifyIs   1003

Retrieve Iconify Status.

INPUT:

None.

RETURN:

Data2   -> The iconify status flag.

1.358 RD_Iconify 1004

COMMAND: RD_Iconify   1004

Switch between Iconify and UnIconify.

INPUT:

None.

RETURN:

None.

NOTE:

No datas needed here, as this Function toggles.

1.359 RD_Spy 1005

```
COMMAND: RD_Spy             1005

Retrieve or toggles the Spy flag.


                      ! NOT IMPLEMENTED YET !
```

1.360 RD_NewUser 1006

```
COMMAND: RD_NewUser         1006

Retrieve or toggles the NewUser flag.

INPUT:

Data1    <- 0 = retrieve, <> 0 = set the NewUser flag.
Data3    <- If Data1=1 the new NewUser flag.

RETURN:

Data2    -> If Data1=0 the NewUser flag.
```

1.361 RD_LoadAccount 1007

```
COMMAND: RD_LoadAccount     1007

Load Userdatas.

INPUT:

Data1        <- The Usernumber of the User you want to load.
StructDummy1 <- The (allocated) Userdata structure.
StructDummy2 <- The (allocated) Userkey structure.

RETURN:

Data2        -> An internal error code.
StructDummy1 -> The (filled) Userdata structure.
StructDummy2 -> The (filled) Userkey structure.

NOTE:

If you pass either for StructDummy1 or StructDummy2 a NULL pointer, your
System may crash!
```

1.362 RD_SaveAccount 1008

```
COMMAND: RD_SaveAccount     1008

Write Userdatas back.

INPUT:

Data1        <- The Usernumber of the User you want to save.
StructDummy1 <- The (filled) Userdata structure.
StructDummy2 <- The (filled) Userkey structure.
```

1.363 RD_LoadConfDat 1009

```
COMMAND: RD_LoadConfDat     1009

Load UserCnf.data.

INPUT:

Data1        <- The Usernumber of the User for whom you want to load the
                actual conference UserConf.data struct.
Data2        <- The selected conference.
                If Data2 lower than 1 or higher than the maximum conferences,
                it will be set to the current conference.

RETURN:

StructDummy1 -> The Userconfdata structure.
```

1.364 RD_SaveConfDat 1010

```
COMMAND: RD_SaveConfDat     1010

Write UserCnf.data.

INPUT:

Data1        <- The Usernumber of the User for whom you want to save the
                actual conference UserConf.data struct.
Data2        <- The selected conference.
                If Data2 lower than 1 or higher than the maximum conferences,
                it will be set to the current conference.
StructDummy1 <- The Userconfdata structure.

RETURN:

None.
```

1.365 RD_AppendAccount 1011

```
COMMAND: RD_AppendAccount   1011

Add a new Account.

INPUT:

StructDummy1 <- The filled Userdata structure.
StructDummy2 <- The filled Userkey structure.

RETURN:

None.

NOTE:

The UserConf.data structures will be automatically generated and set to
default values.
```

1.366 RD_DoOnMaxUserHack 1012

```
COMMAND: RD_DoOnMaxUserHack 1012


                        ! NOT IMPLEMENTED YET !
```

1.367 RD_ServerCommand 1013

```
COMMAND: RD_ServerCommand   1013

Send Server-command.

INPUT:

Data1    <- Data1 of the Server Msg.
Data2    <- Data2 of the Server Msg.
IOString <- String (if needed) for a command.

RETURN:

None.
```

1.368 RD_GetMciFlag 1014

```
COMMAND: RD_GetMciFlag      1014

Retrieve the MCI-Flag.


                        ! NOT IMPLEMENTED YET !
```

1.369 RD_MciFlag 1015

```
COMMAND: RD_MciFlag         1015

Change the MCI-Flag.

                        ! NOT IMPLEMENTED YET !
```

1.370 RD_LoadAccountMisc 1016

```
COMMAND: RD_LoadAccountMisc 1016

Load Misc datas.

INPUT:

Data1        <- The Usernumber of the User you want to load.
Data2        <- The selected conference.
                If Data2 islower than 1 or higher than the maximum
                conferences, it will be set to the current conference.

RETURN:

Data3        -> An internal error code.
StructDummy1 -> The Userdata structure.
StructDummy2 -> The Userdata structure.
StructDummy3 -> The Userconfdata structure.

NOTE:

If Data2 is lower than 1 or higher than the last conference available, Data2
will be set to the current conference number.
```

1.371 RD_SaveAccountMisc 1017

```
COMMAND: RD_SaveAccountMisc 1017

Write Misc datas.

INPUT:

Data1        <- The Usernumber of the User you want to load.
Data2        <- The selected conference.
                If Data2 is lower than 1 or higher than the maximum
                conferences, it will be set to the current conference.
StructDummy1 <- The Userdata structure.
StructDummy2 <- The Userdata structure.
StructDummy3 <- The Userconfdata structure.

RETURN:
```

```
None.
```

1.372 RD_ASLLocalUlPath 1018

```
COMMAND: RD_ASLLocalUlPath  1018

Retrieve or change ASL local upload path.

INPUT:

Data1    <- 0 = retrieve, <> 0 = change ASLLocalUlPath.
IOString <- If Data1 is non-zero, this is the new path (max 21 chars).

RETURN:

IOString -> If Data1=0, it contains ASLLocalUlPath.
```

1.373 RD_ASLTextViewPath 1019

```
COMMAND: RD_ASLTextViewPath 1019

Retrieve or change ASL local textview path.

INPUT:

Data1    <- 0 = retrieve, <> 0 = change ASLTextViewPath.
IOString <- If Data1 is non-zero, this is the new path (max 21 chars).

RETURN:

IOString -> If Data1=0, it contains ASLTextViewPath.
```

1.374 RD_ASLSendFilePath 1020

```
COMMAND: RD_ASLSendFilePath 1020

Retrieve or change ASL local Sendfile path.

INPUT:

Data1    <- 0 = retrieve, <> 0 = change ASLSendFilePath.
IOString <- If Data1 is non-zero, this is the new path (max 21 chars).

RETURN:

IOString -> If Data1=0, it contains ASLSendFilePath.
```

1.375 RD_StartNodeCmd 1021

```
COMMAND: RD_StartNodeCmd    1021

Start a command (Menu/Door) on another node.

INPUT:

Data1    <- The Node you mean.
IOString <- The command incl. Arguments.

RETURN:

None.
```

1.376 RD_BeADoorOnNode 1022

```
                COMMAND: RD_BeADoorOnNode   1022

This command lets you be a door on another node. This means in detail, this
command emulates on the given node that you are a Door started on it.
The Node opens, if needed, it's own FAMEDoorPort(x).
Now you can access this port like a normal door (which you already are!),
meaning there is no difference if you were directly started on this node.
This also means that you have to use the Mustcommands
                MC_DoorStart
                &
                MC_ShutDown
                 !!!

INPUT:

Data1    <- The Node you mean.
IOString <- Arguments given on doorstart (only a joke 8)...but true!)

RETURN:

None.
```

1.377 RD_OpenDoorPort 1023

```
COMMAND: RD_OpenDoorPort    1023

Opens a DoorPort on a Node.

INPUT:

Data1    <- The Node you mean.
Data2    <- The DoorPorttype.

RETURN:
```

```
None.

NOTE:

DoorPortTypes currently supported:

                                    0 = FIM
                                    1 = XIM
                                    2 = TIM
```

1.378 RD_CloseDoorPort 1024

```
COMMAND: RD_CloseDoorPort   1024

Closes a DoorPort on a Node.

INPUT:

Data1    <- The Node you mean.
Data2    <- The DoorPorttype.

RETURN:

None.

NOTE:

DoorPortTypes currently supported:

                                    0 = FIM
                                    1 = XIM
                                    2 = TIM
```

1.379 RD_GetServerList 1025

```
COMMAND: RD_GetServerList   1025

Get the internal Serverliststamm.

INPUT:

None.

RETURN:

StructDummy1 -> The internal Serverlist-base (struct NodeListStamm).
```

1.380 RD_StringToNode 1026

```
                COMMAND: RD_StringToNode    1026

Send a Textstring to a Node.

INPUT:

Data1    <- The Node to which the String should go.
IOString <- The String to be sent.

RETURN:

None.

NOTE:

If ReturnCode=1, the String wasn't sent ! Maybe the selected node doesn't
exist ! To avoid this, use
                NR_ActiveNode
                 to check if selected node exists
and
                AR_GetNodeEnv
                 to retrieve the status of the node. The best way is to use
the internal ServerNodeList.

This String appears at every position/action on the Node.
```

1.381 RD_StringToNodes 1027

```
                COMMAND: RD_StringToNodes   1027

Send a Textstring to a list of Nodes.

INPUT:

IOString     <- The String to be displayed on the Nodes.
StructDummy1 <- The list of Nodes from type struct FAMEDestNodes.

RETURN:

None.

NOTE:

Use
                NR_ActiveNode
                 to check if the Nodes exists and
                AR_GetNodeEnv
                 to retrieve
the status of the nodes. The best way is to use the internal ServerNodeList.

This String appears at every position/action on the nodes.
```

1.382 RD_CONStatus 1028

```
COMMAND: RD_CONStatus       1028

Get or set the CON Outputflag.

INPUT:

Data1    <- 1 means set, else get flag.
Data2    <- Data1 = 1, sets this value.

RETURN:

Data2    -> Data1 <> 1 gets this value.
```

1.383 RD_SERStatus 1029

```
COMMAND: RD_SERStatus       1029

Get or set the SER Outputflag.

INPUT:

Data1    <- 1 means set, else get flag.
Data2    <- Data1 = 1, sets this value.

RETURN:

Data2    -> if Data1 <> 1 gets this value.
```

1.384 RD_SaveMsgFile 1030

```
COMMAND: RD_SaveMsgFile     1030

Save a Msg from.

INPUT:

Data1        <- Conferencenumber to post the message to.
Data3        <- File to message flag (see below).
IOString     <- Path & Filename of message to save.
StructDummy1 <- Allocated and filled struct MailHeader.
StringPtr    <- File or directory contents to message.

RETURN:

Data2        -> Returncode (Result, see below).

Results are: 0 = Successful.
             1 = Your given file doesn't exists.
             2 = Can't get conference datas from your given conf number.
             3 = MsgBase lock failed.
```

```
             4 = MsgStatus.dat can't be opened.
             5 = MsgHeader.dat can't be opened.
             6 = Filecopy of message failed.
             7 = Your struct MailHeader isn't valid.


Data3 flags: 0 = Nothing to attach.
             1 = One single file to attach (pass filename via StringPtr).
             2 = All files of a directory to attach (pass Dirname via
                 StringPtr field).

NOTE:

Minimum filled elements in struct MailHeader are:

 - fmah_ToName
 - fmah_FromName
 - fmah_Subject
 - fmah_MsgStatus <- *MUST* be set to TRUE !
 - fmah_Private

If StringPtr is a valid path to a file it will be copied to the mail and a
flag in the struct MailHeader will be set to mark that there is an
attached file.
```

1.385 RD_SaveMsgList 1031

```
               COMMAND: RD_SaveMsgList     1031

Save a MessageList from External Editor.

INPUT:

Data1        <- Conferencenumber to post the message to.
StructDummy1 <- Filled struct ExternEditor.

RETURN:

None.

NOTE:

Get the Pointer to struct ExternEditor with
               AR_FullEdStruct
                !
Also call
               RD_FreeMsgListFEd
                before
               AR_FullEdStruct
                and make sure that *NO*
message is written on the node until you use this !
```

1.386 RD_FreeMsgListFEd 1032

```
COMMAND: RD_FreeMsgListFEd  1032

Free the internal MessageList.

INPUT:

None.

RETURN:

None.
```

1.387 RD_ActASLULPath 1033

```
COMMAND: RD_ActASLULPath    1033

Retrieve or change Actual ASL Upload Path.

INPUT:

Data1    <- 0 = retrieve, <> 0 = change ActASLULPath.
IOString <- If Data1 <>0, this will be the new path (max. 100 chars).

RETURN:

IOString -> If Data1 = 0, this contains the path.
```

1.388 RD_ActASLTextViewP 1034

```
COMMAND: RD_ActASLTextViewP 1034

Retrieve or change Actual ASL TextView Path.

INPUT:

Data1    <- 0 = retrieve, <> 0 = change ActASLTextViewPath.
IOString <- If Data1 <>0, this will be the new path (max. 100 chars).

RETURN:

IOString -> If Data1 = 0, this contains the path.
```

1.389 RD_ActASLDlPath 1035

```
COMMAND: RD_ActASLDlPath    1035

Retrieve or change Actual ASL Download Path.
```

```
INPUT:

Data1    <- 0 = retrieve, <> 0 = change ActASLDlPath.
IOString <- If Data1 <> 0, this will be the new path (max. 100 chars).

RETURN:

IOString -> If Data1 = 0, this contains the path.
```

1.390 RD_NodeScrToFront 1036

```
COMMAND: RD_NodeScrToFront  1036

Brings node Screen to Front.

INPUT:

None.

RETURN:

None.

NOTE:

If the Screen is iconified, it will be uniconified.
```

1.391 RD_SetServerActCol 1037

```
COMMAND: RD_SetServerActCol 1037

Change Textcolor of the Serveraction.

INPUT:

Data1    <- Color
Data2    <- Node

RETURN:

None.
```

1.392 RD_InitNumFlag 1038

```
               COMMAND: RD_InitNumFlag     1038

Initialize your own NumberFlagList.

INPUT:
```

```
StructDummy1 <- Your own not initialized struct NumFlagList NumFlagListBase
                pointer.

RETURN:

None.

NOTE:

You must supply a pointer to this struct in StructDummy1:

struct NumFlagList
  {
  struct NumFlagList *Next,
                     *Prev;
  char               FileName[102];
  long               FileNumber;
  };

Remember to set NumFlagListBase=NULL; !

If StructDummy1 is FALSE an internal error has been detected. In this
case you don't have to use
                RD_RemoveNumFlag
                ,
                RD_ResetNumFlags
                ,

                RD_SetNumFlag
                 and
                RD_GetNumFlag
                 !!!

You can have as many NumFlagListBases as you need! There are no limits on
the FAME side.

The only thing to do is to take another NumFlagListBase like this:

     struct NumFlagList *NumFlagListBase1,
                        *NumFlagListBase2,
                        *NumFlagListBase3;
```

1.393 RD_RemoveNumFlag 1039

```
                COMMAND: RD_RemoveNumFlag   1039

Remove your own NumberFlagList.

INPUT:

StructDummy1 <- Your own initialized struct NumFlagList NumFlagListBase
                pointer.

RETURN:
```

```
None.

NOTE:

Don't forget to set your NumFlagListBase=NULL; if you use it again with

                RD_InitNumFlag
                !
```

1.394 RD_ResetNumFlags 1040

```
                COMMAND: RD_ResetNumFlags   1040

Reset your own NumberFlagList.

INPUT:

StructDummy1 <- Your own initilized struct NumFlagList NumFlagListBase
                pointer.

RETURN:

None.

NOTE:

This is the combination of
                RD_RemoveNumFlag
                 and
                RD_InitNumFlag
                .

If StructDummy1 is FALSE an internal error has been detected. In this
case you don't have to use
                RD_RemoveNumFlag
                ,
                RD_SetNumFlag
                and
                RD_GetNumFlag
                 !!!
```

1.395 RD_SetNumFlag 1041

```
COMMAND: RD_SetNumFlag      1041

Flag an entry in your own NumberFlagList.

INPUT:

Data1        <- The number of the new entry. It's up to you to count it
                correctly!
```

```
IOString     <- The entry (FileName).
StructDummy1 <- Your own initialized struct NumFlagList NumFlagListBase
                pointer.

RETURN:

Data3    -> Returncode. Possible values:

                        0 -> successfully flagged.
                       -1 -> memory allocation failed.
                       -2 -> Base not initialized.
```

1.396 RD_GetNumFlag 1042

```
COMMAND: RD_GetNumFlag      1042

Get an entry from your own NumberFlagList.

INPUT:

Data1        <- The number of the entry you want to have.
StructDummy1 <- Your own initialized struct NumFlagList NumFlagListBase
                pointer.

RETURN:

IOString -> The entry (FileName).
Data3    -> Returncode. Possible values:

                        0 -> success.
                       -1 -> entry number not found.
                       -2 -> Base not initialized.
```

1.397 RD_GetUserDataLoc 1043

```
COMMAND: RD_GetUserDataLoc  1043

Get the location path of the user.datas

INPUT:

None.

RETURN:

IOString  -> Name of User.data.
StringPtr -> Locationpath of all user datas.
```

1.398 RD_ConfName 1044

```
COMMAND: RD_ConfName        1044

Change a conference name.

INPUT:

Data1    <- The conference number.
IOString <- The new conference name.

RETURN:

None.

NOTE:

If Data1 is lower than 1 or higher than the last conference number available,
Data1 will be set to the current conference number!
```

1.399 RD_ConfLocation 1045

```
COMMAND: RD_ConfLocation    1045

Change a conference location.

INPUT:

Data1    <- The conference number.
IOString <- The new conference location.

RETURN:

None.

NOTE:

If Data1 is lower than 1 or higher than the last conference number available,
Data1 will be set to the current conference number!
```

1.400 RD_FGetNextConf 1046

```
COMMAND: RD_FGetNextConf    1046

Update your conference pointer to the next available conference.

INPUT:

StructDummy1 <- Your struct ConfList pointer.

RETURN:

Data2        -> The new conference number. If relative is on, this value is
```

```
                also relative, else it's the real conference number.

NOTE:

If no next conference is available for the user, this pointer remains un-
changed, else it will be set to the new conference pointer.

If Data2 has a negative value no next conference is available for the user.

If you pass a NULL pointer in StructDummy1, it will be set to point to the
first available conference that the user has access to.
```

1.401 RD_FGetPrevConf 1047

```
COMMAND: RD_FGetPrevConf    1047

Update your conference pointer to the previous available conference.

INPUT:

StructDummy1 <- Your struct ConfList pointer.

RETURN:

Data2        -> The new conference number. If relative is on, this value is
                also relative, else it's the real conference number.

NOTE:

If no previous conference is available for the user, this pointer remains un-
changed, else it will be set to the new conference pointer.

If Data2 has a negative value no previous conference is available.

If you pass a NULL pointer in StructDummy1, it will be set to point to the
first available conference that the user has access to.
```

1.402 RD_FGetConfNum 1048

```
COMMAND: RD_FGetConfNum     1048

Get a pointer to a given conference (relative).

INPUT:

Data1        <- The conference number for which you want to get the ptr.

RETURN:

Data2        -> The new conference number. If relative is on, this value is
                also relative, else it's the real conference number.
StructDummy1 -> Your new struct ConfList pointer.
```

```
NOTE:

This functions checks for the relative flag. If relative conferences are on,
the number used for conference searching (Data1) is handled as a relative
conference number, else it will be used to search for the real conference.

ReturnValues for Data2:

 -2  = Given conferencenumber is not available.
 -3  = Given conferencenumber is available, but user has no access.

In case of -3 your pointer remains unchanged, but if you pass a NULL pointer
it will now be pointing to the first available conference that the user
has access to.

If you pass a value lower than 1 or higher than the last available
conference to Data1, the current conference number will be set instead.
```

1.403 RD_FGetRealConfNum 1049

```
COMMAND: RD_FGetRealConfNum 1049

Get a pointer to a given conference (not relative).

INPUT:

Data1        <- The conference number for which you want to get the ptr.

RETURN:

Data2        -> The new conference number, it's always the real one !
StructDummy1 -> Your new struct ConfList pointer.

NOTE:

If Data2 is negative, your given conference number is not available.

Your pointer normally remains unchanged if Data2 is negative, but if you pass
a NULL pointer, it will be set to the first available conference if Data2
is negative.

If you pass a value lower than 1 or higher than the last available conference
to Data1, the current conference number will be set instead.
```

1.404 RD_FGetConfBase 1050

```
COMMAND: RD_FGetConfBase    1050

Get the conference base pointer.

INPUT:
```

```
None.

RETURN:

StructDummy1 -> The conference base pointer.

NOTE:

You can only use the element RelativeNext to get the first available
conference for the user.

Example:

struct ConfList *MyConfPtr = DoorMsg -> StructDummy1 -> RelativeNext;

while(MyConfPtr)
  {
  ...
  ...
  MyConfPtr = MyConfPtr -> RelativeNext;
  }
```

1.405 RD_FGetAktConf 1051

```
COMMAND: RD_FGetAktConf     1051

Get the current conference pointer.

INPUT:

None.

RETURN:

StructDummy1 ->~The~current conference pointer.

NOTE:

You can use this pointer to check for following conferences (RD_FGetNextConf)
or previous conferences~(RD_FGetPrevConf).

Example:

while(DoorMsg -> StructDummy1)
  {
  ...
  ...
  DoorMsg -> StructDummy1 = DoorMsg -> StructDummy1 -> RelativeNext;
  }
```

1.406 RD_GiveNumFromConf 1052

```
COMMAND: RD_GiveNumFromConf 1052

Get the relative/non-relative number from a conference pointer.

INPUT:

StructDummy1 <- Your struct ConfList pointer.

RETURN:

Data2    -> The conference number. If relative is on, this value is
            also relative, else it's the real conference number.
```

1.407 RD_SaveUFlagList 1053

```
                COMMAND: RD_SaveUFlagList   1053

Forbid Upload procedure from clearing the Upload flag after the transfer.

INPUT:

Data1    <- 0 = clear, else don't clear.

RETURN:

None.

NOTE:

If Data1 is non zero, the Upload procedure won't clear the list of uploaded
files and therefor a door may use it. If Data1 is set to zero, you tell the
upload procedure to clear the uploaded list of files now.

Normally the Upload procedure clears the list after it has finished with
the filechecking. The FileID procedure as the last thing in the upload chain
then normally frees the list to give the memory back to system. But this
is not really a must, because when starting the next upload this list will
be cleared automatically before the real transfer begins. It's a question
of memory usage. The best way is to free the list after the upload because
we don't know how many files are in the list and when the next transfer
will be performed.

If a door wants to know which files were uploaded after the upload procedure
it's neccessary to get this list. To get the filled list use the command

                RD_SaveUFlagList
                 with Data1 set to TRUE. After
this get the Base pointer of this list with RD_GetUFlagList. After you
have read all neccessary infos from the list, use RD_ClearUFlagList to
clear the list. Please notice that this list is strictly read-only!

If the door no longer needs any information about the uploaded list, use
RD_SaveUFlagList again with Data1 set to FALSE (0). This sets the upload
procedure to it's origin state. It's not really bad if you forget to set
```

```
it back, because if your door shuts down this flag will be automatically
set back by FAME.
```

1.408 RD_ClearUFlagList 1054

```
COMMAND: RD_ClearUFlagList  1054

Clear the Upload flag list now.

INPUT:

None.

RETURN:

None.
```

1.409 RD_GetUFlagList 1055

```
COMMAND: RD_GetUFlagList    1055

Retrieve the basepointer of the upload flag list.

INPUT:

None.

RETURN:

StructDummy1 -> The base pointer of the upload flag list.
```

1.410 RD_GetMailHeader 1056

```
COMMAND: RD_GetMailHeader   1056

Retrieve the Mailheader Pointer (SYSCMD: MAILHEADER).

INPUT:

None.

RETURN:

StructDummy1 -> The MailHeader pointer.
```

1.411 RD_ShowMailHeader 1057

```
COMMAND: RD_ShowMailHeader  1057

Inform the Node to show the MailHeader (SYSCMD: MAILHEADER).

INPUT:

Data1 -> TRUE to show the MailHeader, otherwise set it to FALSE.

RETURN:

None.
```

1.412 How to contact the authors...

```
  ------------------------------------------------------------------------
  -                        CONTACTS & NUMBERS                            -
  ------------------------------------------------------------------------

If you want to contact the author of this document, SieGeL/tRSi, try the
following:

1. Call pUNISHMENT iNC. BBS, which is my board. It's also the FAME Support
   BBS. Numbers are:

                         +49(0) 30 694 84 70 (v.34)
                         +49(0) 30 694 85 70 (v.34+)

2. Or contact me via E-Mail:

                              siegel@trsi.de

3. You may also visit our homepage:

                           http://www.trsi.de/inno


If you want to contact Strider, simply call my bbs, too, or try this E-Mail:

                              strider@trsi.de

Also all other (tRSi-iNNOVATiONs) members are available on pUNISHMENT iNC.
 and pARALYSIS BBS, but FAME-specific questions should be written only on
         the pUNISHMENT iNC. BBS, as this is the FAME support BBS.


   You could also reach us on IRC, channel #FAME. if you have Internet
  access, you may visit this channel and talk directly to the developers!
```

1.413 Function currently not implemented !

```
                THIS FUNCTION IS CURRENTLY NOT AVAILABLE !
```

1.414 FLVL_FLAG Defines for NR_GetUserLevelFlags

```
                          These are the flags returned by
               NR_GetUserLevelFlags
                 #define FLVL_ReadMessages   1
#define FLVL_EnterMessage   2
#define FLVL_CommenttoSysOp   3
#define FLVL_PageSysOp      4
#define FLVL_ZoomMail     5
#define FLVL_ReadBulletins    6
#define FLVL_WHOisOnline    7
#define FLVL_UserStatus     8
#define FLVL_ZippyTextSearch    9
#define FLVL_JoinConference   10
#define FLVL_FileListing    11
#define FLVL_NewFiles     12
#define FLVL_Upload     13
#define FLVL_Download     14
#define FLVL_ViewFiles      15
#define FLVL_FreeResuming   16
#define FLVL_EditUserAccount    17
#define FLVL_EditPassword   18
#define FLVL_EditUserLocation   19
#define FLVL_EditUserCityState    20
#define FLVL_EditUserName   21
#define FLVL_EditPhonenumber    22
#define FLVL_Relogin      23
#define FLVL_UploadStatus   24
#define FLVL_PublicMessageFiles   25
#define FLVL_PrivateMessageFiles  26
#define FLVL_AttachFile     27
#define FLVL_MCIMessages    28
#define FLVL_BreakChat      29
#define FLVL_OverideChat    30
#define FLVL_OverideTime    31
#define FLVL_EAllMessages   32
#define FLVL_EditFiles      33
#define FLVL_EditDirs     34
#define FLVL_DeleteMessages   35
#define FLVL_AccountEditing   36
#define FLVL_SysOpCommands    37
#define FLVL_Shell      38
#define FLVL_SysOpRead      39
#define FLVL_SysOpView      40
#define FLVL_SysOpDownload    41
#define FLVL_OverideDefaults    42
#define FLVL_GlobalUD     43
#define FLVL_FastRelogin    44
#define FLVL_FastGoodbye    45
#define FLVL_ViewBinaryFiles    46
#define FLVL_ZoomFilelist   47
#define FLVL_NodeChat     48
```

```
#define FLVL_Hydra      49
#define FLVL_RIPGfx     50
#define FLVL_CryptMsgs      51
#define FLVL_EditUserBirthday   52
#define FLVL_DeleteMsginDays    53
#define FLVL_OverwriteFiles   54
#define FLVL_EditMessages   55
#define FLVL_KeepotherMessages    56
#define FLVL_OthUsonotherMsg    57
#define FLVL_SModem     58
#define FLVL_BBSHelp      59
#define FLVL_QuietMode      60
#define FLVL_OnLineMessageOLM   61
#define FLVL_CaptureOnlineUserOLC 62
#define FLVL_DisconOnlineUserOLD  63
#define FLVL_OnlineUserChatOLG    64
#define FLVL_SpyOnlineUserOLS   65
#define FLVL_OnlineUserTimeOLT    66
#define FLVL_SysOpBinaryView    67
#define FLVL_ViewEditCallersLog   68
#define FLVL_ViewEditDoorLog    69
#define FLVL_ViewEditUDLog    70
#define FLVL_AmigaDOSList   71
#define FLVL_AmigaDOSDir    72
#define FLVL_AmigaDOSInfo   73
#define FLVL_ConfAccessEditor   74
#define FLVL_ForceWHO     75
#define FLVL_RealHidden     76
#define FLVL_RenewedSubmission    77
#define FLVL_PriorityMsg    78
#define FLVL_Obsolete     79
#define FLVL_EditFileFlags    80
```