TWN4

Director User Guide

DocRev6, September 11, 2014

Elatec GmbH

Contents

1 Introduction

This document describes all features and the usage of the software Director V1.16.
Director is an application that runs on Microsoft Windows. The purpose of this software is to easily test all the internal firmware functions and powerful APIs of the TWN4, a state of the art RFID reader.

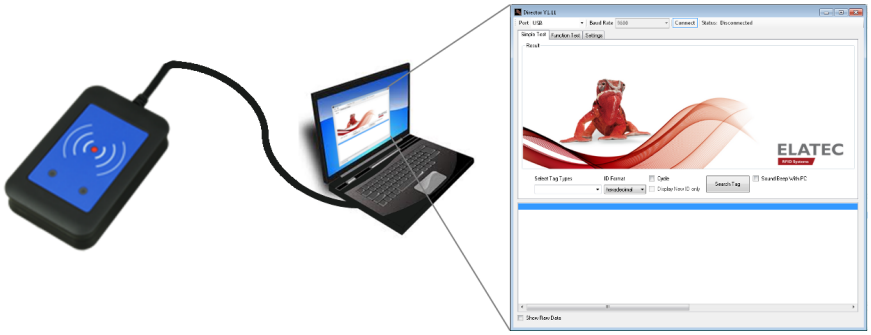
The behavior of TWN4 can be controlled by the user with so called apps. These apps are little programs which are mainly written in C, a common programming language. They have access to several APIs which makes it easy to develop complex RFID applications.

The Director can be used to test the firmware functions and APIs of TWN4 without writing a single line of code. It communicates with the reader over a serial interface like USB or RS232 with an ASCII or binary protocol, which is called simple protocol.

1.1 System Requirements

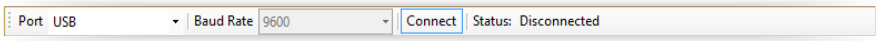
Host:	Director needs Microsoft Windows with .NET Framework 3.5 installed.
TWN4:	Firmware 1.64 or higher (PRS104 or higher) e.g. TWN4_Cx164_PRS104_CDC_Simple_Protocol.bix

2 Usage of Director

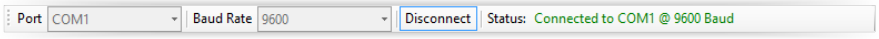


2.1 Startup

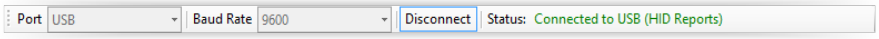
At startup, Director needs to connect to a TWN4 with Simple Protocol. This is done by clicking on the button `Connect`. Before connecting, you may need to select the Port on which TWN4 is cable-connected. If this is a serial COM Port, also the Baud rate needs to be selected (e.g. 9600).



After connecting successfully, the version string will be displayed in the log. Also the connection status in the connection toolbar will change from `Disconnected` to `Connected` and the appropriate COM port will be displayed. In case of a serial COM port (i.e. TWN4 is not connected via USB) also the Baud rate is shown.



If USB is selected on listbox `Ports` and TWN4 firmware supports HID reports, Director will automatically connect with HID reports. This will be displayed on the connection status.

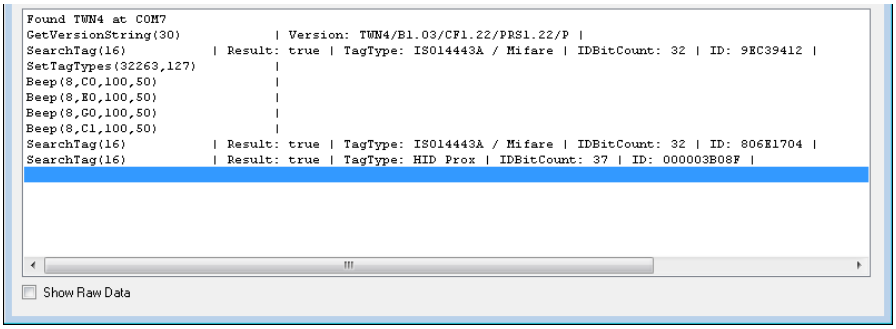


If there is no correct response from the reader, a message will be written in the log box (`Please connect a TWN4 with firmware TWN4_Cx164_PRS104_CDC_Simple_Protocol.bix or higher`). This procedure will also take place if the reader is disconnected and connected

again.

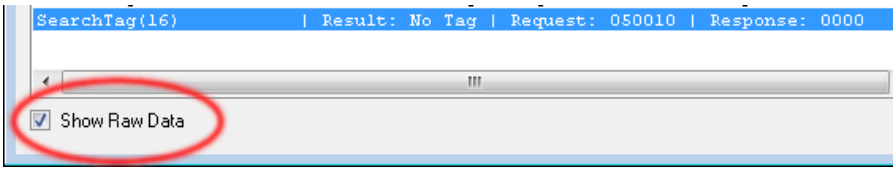
2.2 Log

Every function call and its result are logged in the log box at the bottom of the application. Also every message will be displayed here.



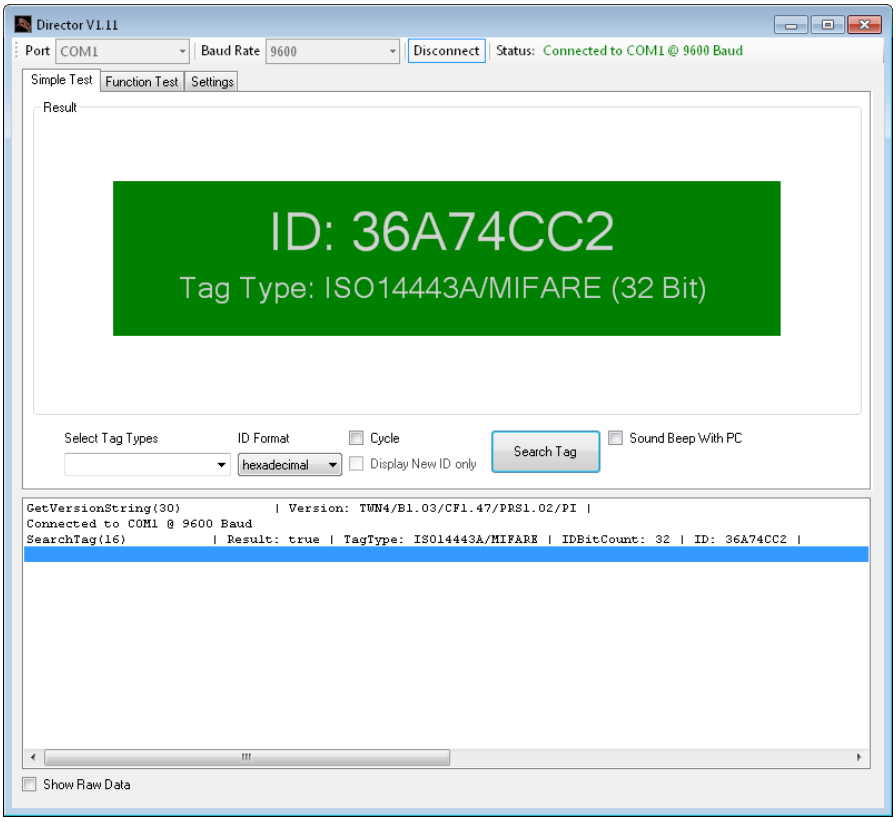
There is a context menu available for the log box which allows the user either to clear or to copy all items to the clipboard.

Beneath the log is a checkbox called *Show Raw Data*. If checked, each function call will also log the request / response which are sent to / from the reader in raw format.

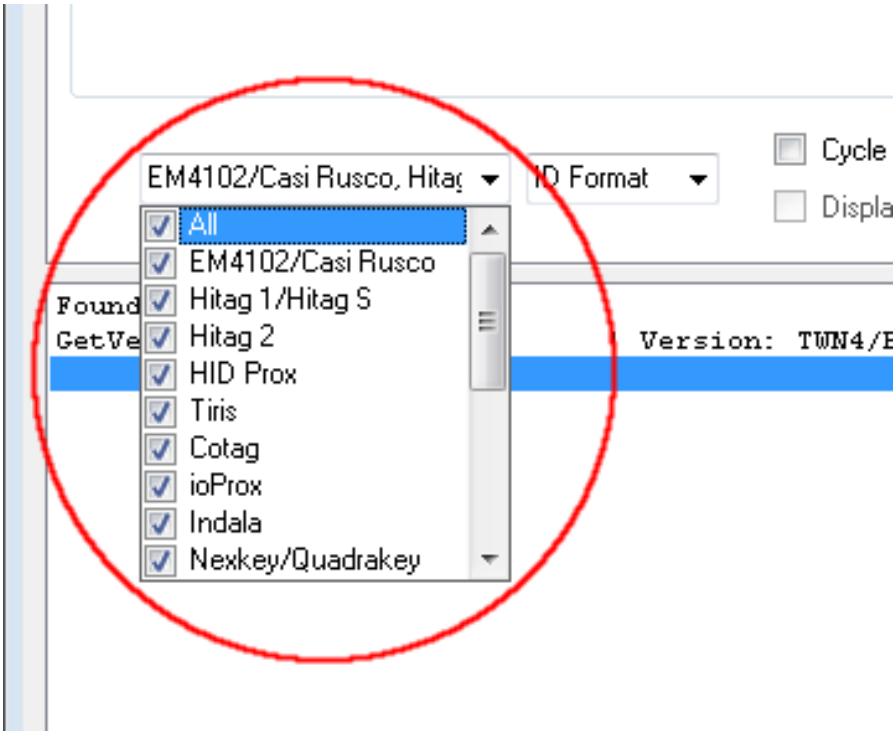


2.3 Simple Test

Clicking the button *Search Tag* lets the TWN4 search for a transponder. If one is found, the ID, the tag type and the bit count are displayed in the result box. Furthermore the reader sounds a short beep. To search for transponders continuously, just check the *Cycle* checkbox.
If the checkbox *Sound beep with PC* is checked, the PC sounds a beep additionally to the beep of the TWN4. This is useful for a TWN4 without speaker.
You can control the format of the ID with the combo box *ID format* to hexadecimal, decimal or binary output.

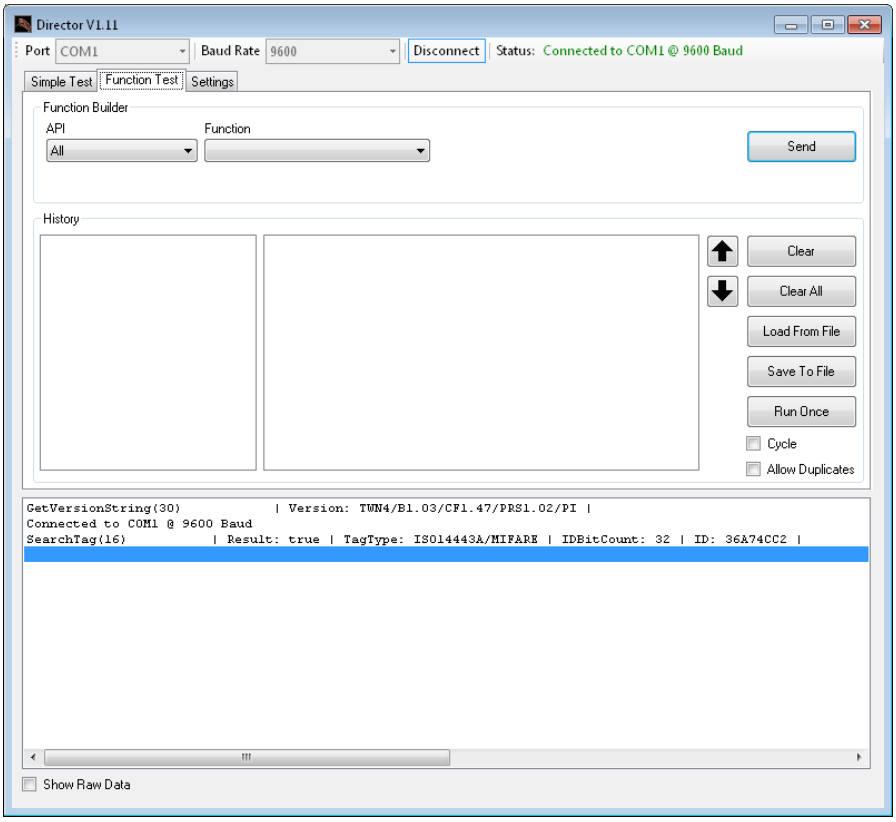

Director V1.11
Port
COM1
Baud Rate
9600
Disconnect
Status: Connected to COM1 @ 9600 Baud
Simple Test
Function Test
Settings
Result
ID: 36A74CC2
Tag Type: ISO14443A/MIFARE (32 Bit)
Select Tag Types
ID Format
hexadecimal
Cycle
Display New ID only
Search Tag
Sound Beep With PC
GetVersionString(30) | Version: TWN4/B1.03/CF1.47/PRS1.02/PI |
Connected to COM1 @ 9600 Baud
SearchTag(16) | Result: true | TagType: ISO14443A/MIFARE | IDBitCount: 32 | ID: 36A74CC2 |
Show Raw Data

There is also a combo box *Select Tag Types* which lets you comfortably enable or disable the tag types the reader is searching for. This is also possible in cyclic operation (checkbox *Cycle* is checked).



2.4 Function Test

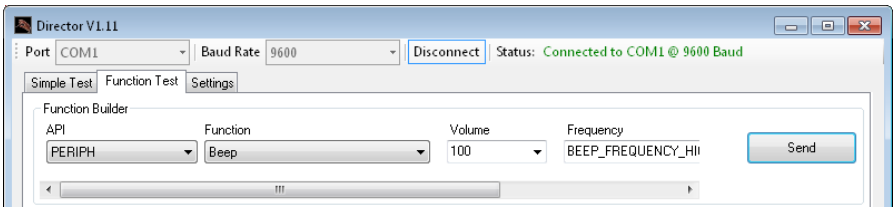
This is the section for advanced users and is the heart of the application. Here you have access to nearly all firmware functions of TWN4.



2.4.1 Function Selection

First we need to build our function call, which consists of the functions name and its parameters. The function can be selected by the drop down list *Function*. The huge amount of available functions can be filtered by the drop down list *API*, which lists all APIs.

If a function is selected, a combo box appears for each parameter (if any). The combo boxes are hopefully filled with some default values. Then you can just click the send button on the right and see what happens. If a default value is missing, you have to fill the box by yourself. Of course you can also overwrite the default values.



2.4.2 Formats for Parameter Values

The parameter values can be typed in different formats which are listed below.

Type	Format	Example
Decimal value	Number	`110`
Hexadecimal	Prefix 0x	`0x6e, 0x6E`
Binary	Prefix 0b	`0b01101110`
ASCII character	Quotes	`‘n‘`
ASCII string	Double quotes	`"A string"`
Byte array	Prefix $	`$4120737472696E67`

Table 2.1: Formats for parameter values

Each single number or character can be combined to a list of values, if a parameter needs more than one. Just separate the values with a space. Each value can have a different format. It is also possible to type in a list of hexadecimal values as one block (see *Byte array* in table above)

For example,

```
0x41 0x20 0x73 0x74 0x72 0x69 0x6E 0x67
```

is the same like

```
‘A‘ 0x20 0x73 0x74 0x72 0x69 0b01101110 147
```

or

```
$4120737472696E67
```

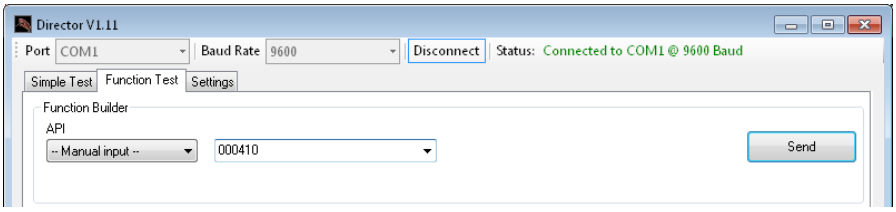
In case of a byte array, it is also possible to leave the field blank. This results in a byte array with length 0.
If the mouse hovers over a parameter field, a small hint appears which type and how many values are expected.

When a function is executed by clicking the *Send* button or just hitting the Enter key, the request is sent to the reader. Hopefully we get a response from the reader, which is then

parsed and split into its return values. The function call with its parameters and the return values are displayed in the log box, see chapter 2.2

2.4.3 Manual input

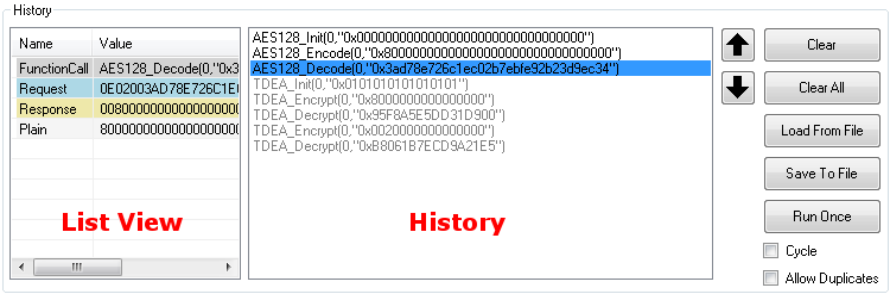
It is possible to type in a command manually by selecting – *Manual input* – in the drop down box *API*. You have to type in the whole command with API number, function number and parameters, e.g. 000420



2.4.4 History

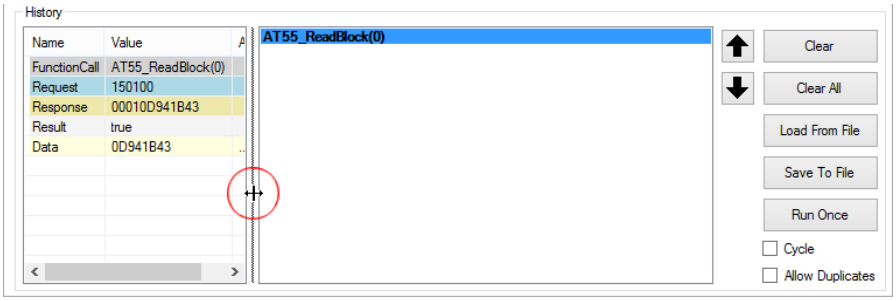
Each executed function with its request and corresponding response will be stored in the history box. If a function call with same parameters is already stored, it will be overwritten with the currently executed function. If identical function calls should be stored, the checkbox *Allow Duplicates* must be checked. Double clicking a stored function will execute it again with same parameters.

If a function has (not yet) got a response, it is displayed in gray color, otherwise it is black.



When a function call is selected by mouse click, the function builder shows the function with its parameters, to make it easy to start the function again with the same or with changed parameters. When a function call is selected by mouse or keyboard, the function call and the return values are displayed in the list view on the left of the history box.

If either the list view or history box is too small, it is possible to resize it with the middle bar that separates both list view and history box.



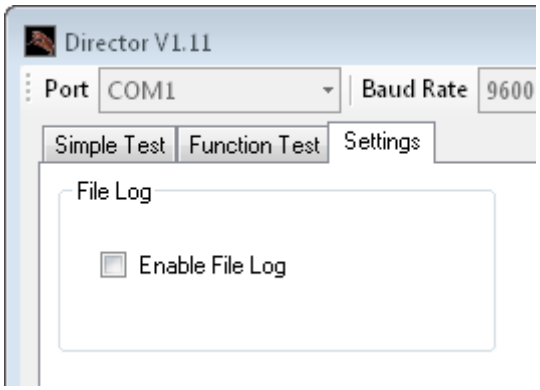
The position of a function call in the history box can be changed by clicking the buttons with the up or down arrow.

The *Clear* button deletes a selected item from the list while the *Clear All* button deletes the complete list.

The items of the history box can be saved to a text file and loaded again with the buttons *Save To File* and *Load From File*.

Providing that one or more items are in the history, they can be executed one time in sequence by clicking the button *Run Once* or in a cycle by checking the *Cycle* checkbox.

2.5 Settings



2.5.1 Protocol

The Simple Protocol can be used in two ways. The simplest way is to send ASCII characters which encodes the data to transmit. The other way is to send the data directly in binary format.
You can select the behavior of Simple Protocol with the drop-down box *Protocol*.
It is also possible to use Simple Protocol with a CRC. This can enabled or disabled with the drop-down box *CRC*.

2.5.2 File Log

If the checkbox *Enable File Log* is checked, the log (see chapter 2.2) will also be written to the file `log.txt` in the same directory as the application.

2.6 Constants

There are a lot of predefined constants that can be used for function parameters (you can either type in the constant or the respective value). Many of these constants are also used in the TWN4 firmware and have the same value there.

Constant	**Value**
true	1
false	0
ON	1
OFF	0
CHANNEL_NONE	0
CHANNEL_USB	1
CHANNEL_COM1	2
CHANNEL_COM2	3
CHANNEL_I2C	4
CHANNEL_CCID_DATA	0x10
continued on next page...	

Constant	Value
CHANNEL_CCID_CTRL	0x11
CHANNEL_SAM1	0x20
CHANNEL_SAM2	0x21
CHANNEL_SAM3	0x22
CHANNEL_SAM4	0x23
CHANNEL_SC1	0x28
CHANNEL_RNG	0x30
DIR_IN	1
DIR_OUT	0
COM_WORDLENGTH_8	8
COM_PARITY_NONE	0
COM_PARITY_ODD	1
COM_PARITY_EVEN	2
COM_STOPBITS_0_5	1
COM_STOPBITS_1	2
COM_STOPBITS_1_5	3
COM_STOPBITS_2	4
COM_FLOWCONTROL_NONE	0
ALL_HFTAGS	0xFFFFFFFF
ALL_LFTAGS	0xFFFFFFFF
REDLED	0x01
GREENLED	0x02
YELLOWLED	0x04
ALLLEDS	0x07
GPIO0	0x01
GPIO1	0x02
GPIO2	0x04
GPIO3	0x08
GPIO4	0x10
GPIO5	0x20

continued on next page. . .

Constant	Value
GPIO6	0x40
GPIO7	0x80
GPIO_PUPD_NOPULL	0
GPIO_PUPD_PULLUP	1
GPIO_PUPD_PULLDOWN	2
GPIO_OTYPE_PUSHPULL	0
GPIO_OTYPE_OPENDRAIN	1
BEEP_FREQUENCY_HIGH	2400
BEEP_FREQUENCY_LOW	2057
C0	523
C#0	554
D0	587
D#0	622
E0	659
F0	698
F#0	740
G0	784
G#0	831
A0	880
Bb0	932
H0	988
C1	1047
C#1	1109
D1	1175
D#1	1245
E1	1319
F1	1397
F#1	1480
G1	1568
G#1	1661

continued on next page. . .

Constant	Value
A1	1760
Bb1	1865
H1	1976
I2CMODE_MASTER	0x0000
I2CMODE_SLAVE	0x1000
I2CMODE_CHANNEL	0x2000
I2CMODE_ADDRESS_MASK	0x007F
KEYA	0
KEYB	1
DESF_AUTHMODE_COMPATIBLE	0
DESF_AUTHMODE_EV1	1
DESF_COMMSET_PLAIN	0
DESF_COMMSET_PLAIN_MACED	1
DESF_COMMSET_FULLY_ENC	3
DESF_KEYTYPE_3DES	0
DESF_KEYTYPE_3K3DES	1
DESF_KEYTYPE_AES	2
DESF_KEYLEN_3DES	16
DESF_KEYLEN_3K3DES	24
DESF_KEYLEN_AES	16
DESF_FILETYPE_STDDATAFILE	0
DESF_FILETYPE_BACKUPDATAFILE	1
DESF_FILETYPE_VALUEFILE	2
DESF_FILETYPE_LINEARRECORDFILE	3
DESF_FILETYPE_CYCLICRECORDFILE	4
CRYPTO_ENV0	0
CRYPTO_ENV1	1
CRYPTO_ENV2	2
CRYPTO_ENV3	3
CRYPTO_ENV_CNT	4

continued on next page. . .

Constant	Value
CRYPTOMODE_AES128	0
CRYPTOMODE_3DES	3
CRYPTOMODE_3K3DES	4
CRYPTOMODE_CBC_DES	5
CRYPTOMODE_CBC_DFN_DES	6
CRYPTOMODE_CBC_3DES	7
CRYPTOMODE_CBC_DFN_3DES	8
CRYPTOMODE_CBC_3K3DES	9
CRYPTOMODE_CBC_AES128	10
WAKEUP_BY_USB_MSK	0x01
WAKEUP_BY_COM1_MSK	0x02
WAKEUP_BY_COM2_MSK	0x03
ISO7816_POWERSELECT_AUTO	0
ISO7816_POWERSELECT_5V	1
ISO7816_POWERSELECT_3V	2
ISO7816_POWERSELECT_1V8	3
SID_INTERNALFLASH	1
FS_READ	0
FS_WRITE	1
FS_APPEND	2
FILE_ENV0	0
FILE_ENV1	1
FILE_ENV2	2
FILE_ENV3	3
FILE_ENV_CNT	4
FS_POSABS	0
FS_POSREL	1
FS_POSEND	2
FS_MOUNT_NONE	0
FS_MOUNT_READONLY	1

continued on next page. . .

Constant	Value
FS_MOUNT_READWRITE	2
FS_FORMATMAGICVALUE	0x74496F44

Table 2.2: Constants used in Director