

## **DMcontrol documentation**

**COLLABORATORS**

	<i>TITLE :</i> DMcontrol documentation		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		March 1, 2023	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>DMcontrol documentation</b>	<b>1</b>
1.1	DMcontrol documentation	1
1.2	Table Of Contents	1
1.3	Introduction	2
1.4	Installation	2
1.5	Using the program.	2
1.6	Commands	3
1.7	LIST	3
1.8	TYPEINFO	4
1.9	ADDDTYPE	5
1.10	REMTYPE	5
1.11	FILEINFO	6
1.12	ADDFILE	6
1.13	ADDDIR	6
1.14	SCANDIR	7
1.15	TESTFILES	7
1.16	CREATEBUNDLE	7
1.17	Flags	8
1.18	NODATACHECK	8
1.19	LONGNAME	9
1.20	NOXPB	9
1.21	SIZE	9
1.22	History	9

---

# Chapter 1

## DMcontrol documentation

### 1.1 DMcontrol documentation

DMcontrol documentation

DMcontrol v1.9

© Alexis "Cyb" Nasr 1996-1997

\$VER: May 1997

[Table Of Contents](#)

[Introduction](#)

[Installation](#)

[Using the program.](#)

[History](#)

### 1.2 Table Of Contents

Table Of Contents

MAIN [DMcontrol documentation](#)

1. [Introduction](#)

2. [Installation](#)

3. [Using the program.](#)

3.1. [Commands](#)

3.1.1. [LIST](#)

3.1.2. [TYPEINFO](#)

3.1.3. [ADDDTYPE](#)

3.1.4. [REMTYPE](#)

3.1.5. [FILEINFO](#)

3.1.6. [ADDFILE](#)

3.1.7. [ADDDIR](#)

3.1.8. [SCANDIR](#)

---

### 3.1.9. TESTFILES

### 3.1.10. CREATEBUNDLE

## 3.2. Flags

### 3.2.1. NODATACHECK

### 3.2.2. LONGNAME

### 3.2.3. NOXPK

### 3.2.4. SIZE

## 4. History

## 1.3 Introduction

### 1. Introduction

DMcontrol is part of the datamaster.library package & therefore should not be separated from it.

This program provides many functions for addition/removal/information of the recognizers/filetypes.

It is useful for the user, to fully control the library, and for the programmer while developing a recognizer. The most directly useful "user-function" is **TESTFILES** (just make a button of it in your favourite Disk-Tool). **CREATEBUNDLE** will surely be useful too.

Note: You'll find the assembler sourcecode of this program in the developer directory.

## 1.4 Installation

### 2. Installation

This is a CLI-tool that will surely go in C:

## 1.5 Using the program.

### 3. Using the program.

Typing the classic "DMcontrol ?" will give you the templates:

L= **LIST** /K/S, RT= **REMTYPE** /K, AT= **ADDTYPE** /K, AF= **ADDFILE** /K, AD= **ADDDIR** /K,  
SD= **SCANDIR** /K, TF= **TESTFILES** /K/M, **NOXPK** /K/S, S= **SIZE** /K/N, DC= **NODATACHECK** /K/S,  
LN= **LONGNAME** /K/S, TI= **TYPEINFO** /K, FI= **FILEINFO** /K, CB= **CREATEBUNDLE** /K:

Explanation of the standard flags:

K means Keyword

S means Switch (works as a ON/OFF switch, no parameter needed)

N means Numeric value as parameter (else it's a string)

---

M means multiple parameters possible.

Some keywords have "short" versions.... For example , you can use just "L" instead of "LIST", etc...

Note: typing just "DMcontrol" will open the library & load all the recognizers.This might be useful no?

Commands

Flags

## 1.6 Commands

### 3.1. Commands

All functions that perform some adding/loading will get a DefaultDir set to LIBS:Recognizers/ so no need to type it each time.

Of course,name may begin with any path such as Store/ or MyDH0:OtherRecos...

IMPORTANT: \*\* There can only be ONE function per command line \*\*

LIST

TYPEINFO

ADDDTYPE

REMTYPE

FILEINFO

ADDFILE

ADDDIR

SCANDIR

TESTFILES

CREATEBUNDLE

## 1.7 LIST

### 3.1.1. LIST

This function lists all the filetypes currently loaded.

These names are the ones that you'll use when making parameter files for programs using datamaster.library.

(example in DMlauncher:

```
TYPE=GFX-JPEG
```

```
ACTION=Sys:Utilities/FastJPEG %s
```

```
END_DEF
```

```
)
```

If you want to know more about the recognizers behind each MAJOR-filetype, just use " **TYPEINFO** filetype" instead.

---

Note:If LIST is associated to an addition/removal function ,it will do a listing after the operation,to reflect current list changes.

see **REMTYPE** , **ADDTYPE** , **ADDFILE** , **ADDDIR** , **SCANDIR** .

Example:

-----

```
> DMcontrol LIST
>>17<< Filetypes available:
ANIM-IFF
ASCII-AmigaGuide
ASCII-Text
Executable
Generic
GFX-GIF
GFX-IFF
GFX-JPEG
Icon
MUSIC-Others
MUSIC-PtkClones
MUSIC-Synth
PACKED-LHA
PACKED-LZX
PACKED-XPk
SMPL-IFF
SMPL-Others
```

## 1.8 TYPEINFO

### 3.1.2. TYPEINFO

Gives precise information (version,priority,number of sub-formats recognized etc...) about the recognizer(s) defining a MAJOR-filetype name.

Defining filetype as "all" will give info of ALL the filetypes.

Example:

-----

```
> DMcontrol TYPEINFO GFX-GIF
Recognizer:GFX-GIF (1.0),Pri=1 [1 SubRecons]
Description:GIF Packed graphics
Flags: CHECKPATTERN CHECKDATA
```

If there are SUB-FILETYPES that can be recognized separately, they will also be printed out.

---

..."everything written in BOLD WHITE can be used as filetype" :)

Note for programmers:

~~~~~

The array of the DMR\_SubTypesTable tag is Pri-sorted, \*NOT\* Alpha-sorted.

So if you want to print it alpha sorted like in DMcontrol, well.... you have to make a copy of the array and sort it yourself :-)

## 1.9 ADDTYPE

### 3.1.3. ADDTYPE

Adds a MAJOR-filetype to the list.

This may in fact load MORE THAN ONE recognizer;it will load all recognizers beginning with such a name,which should correspond in fact to a SAME filetype,if conventions are respected ;-)

WARNING:As the library does ON PURPOSE accept multiple recognizers having the same MAJOR-filetype,you may add as many times as you want the same recognizers,but they will ACCUMULATE in memory,leading to a loss of memory,and as many times checking-time multiplying!!!!

This is not a bug,it's just a consequence of the library's philosophy.

If there is enough pressure on me from users ,I'll find a way to fix this ;-)

Putting **LIST** in the command line will do a listing after the operation.

Example:

-----

```
> dmcontrol ADDTYPE=WeirdType
```

this would add/load ALL these files:

WeirdType.data.UpdatedByJoe

WeirdType.patt.

WeirdType.patt.somemore

## 1.10 REMTYPE

### 3.1.4. REMTYPE

Removes a MAJOR-filetype from memory,which means removing all the recognizers defining it.

Defining filetype as "all" will remove...ALL the external recognizers :-)

Putting **LIST** in the command line will do a listing after the operation.

---



## 1.11 FILEINFO

### 3.1.5. FILEINFO

exactly as **TYPEINFO** but on a non-loaded recognizer file.

This will load,analyse,& free the recognizer file.

## 1.12 ADDFILE

### 3.1.6. ADDFILE

Adds a **\*\*single recognizer file\*\***,unlike **ADDDTYPE** .

Note:this time,exact name will be required,of course.

**WARNING:**As the library does **ON PURPOSE** accept multiple recognizers having the same **MAJOR**-filetype,you may add as many times as you want the same recognizers,but they will **ACCUMULATE** in memory,leading to a loss of memory,and as many times checking-time multiplying!!!!

This is not a bug,it's just a consequence of the library's philosophy.

If there is enough pressure on me from users ,I'll find a way to fix this ;-)

Putting **LIST** in the command line will do a listing after the operation.

## 1.13 ADDDIR

### 3.1.7. ADDDIR

Adds all the recognizers of a dir to list.

use "ADDDIR /" to scan default directory (LIBS:Recognizers/)

**WARNING:**As the library does **ON PURPOSE** accept multiple recognizers having the same **MAJOR**-filetype,you may add as many times as you want the same recognizers,but they will **ACCUMULATE** in memory,leading to a loss of memory,and as many times checking-time multiplying!!!!

This is not a bug,it's just a consequence of the library's philosophy.

If there is enough pressure on me from users ,I'll find a way to fix this ;-)

Putting **LIST** in the command line will do a listing after the operation.

**NOTE:** This function takes advantage of a "bundle" file, if present in the directory.(see **CREATEBUNDLE** )

## 1.14 SCANDIR

### 3.1.8. SCANDIR

Flushes all external recognizers then adds all the recognizers of a dir to list.

Use "SCANDIR /" to scan default directory (LIBS:Recognizers/)

Putting **LIST** in the command line will do a listing after the operation.

NOTE: This function takes advantage of a "bundle" file, if present in the directory.(see **CREATEBUNDLE** )

## 1.15 TESTFILES

### 3.1.9. TESTFILES

An easy function the get the type of a file:

You can test several files in the same time.

Examples:

-----

```
> dmcontrol testfiles disk.info s:startup-sequence
```

File disk.info : Icon

File s:startup-sequence : ASCII-Text

```
> dmcontrol Mods-4:Modules/Synth/FC13/FC13.Astaroth-2-4
```

File Mods-4:Modules/Synth/FC13/FC13.Astaroth-2-4 : Future Composer 1.3

(subtype of MUSIC-Synth)

Some option flags can be added when using TESTFILES:

(They are OFF by default)

**NODATACHECK**

**LONGNAME**

**NOXPK**

**SIZE**

## 1.16 CREATEBUNDLE

### 3.1.10. CREATEBUNDLE

This function takes the same parameter as **ADDDIR** or **SCANDIR** : the name of a directory containing recognizers.

use "CREATEBUNDLE /" for default dir (LIBS:Recognizers/)

It will merge all the recognizers of the directory into one "big" file.

If the library finds such a file when scanning a recognizers directory

(mainly on initialization), it will use it instead of loading the recognizer

---

one by one.

This will be slightly faster (specially if you have lots of recognizers).

Of course don't forget to "refresh" this file each time you install new recognizers.

"tech" info follows, for those who'd be interested (?)

First the program loads all the recognizers to memory.(it LoadSeg()s the files, checks if they are correct recognizers, and if so, loads them "again").

Once everything is done, the bundle is created.

What is more weird, is the way the bundle is LOADED by datamaster.library.

I'd have liked to make the LoadSeg()s directly on memory but this doesn't seem possible and I don't want to rewrite a hunk-reallocator for that ;(

So the solution was to just create a temp dir in RAM: and split back all the recognizers to it, and THEN, loadseg them with a dmScanRecoDir "classic" call(!) . Once it's done, temp files are deleted and basta...

Well, it still is faster than loading small files from HD.

(of course if you have a SCSI-II Barracuda HD, and AFS, ok ok... I only have a poor IDE with FFS :-))

## 1.17 Flags

### 3.2. Flags

**NODATACHECK**

**LONGNAME**

**NOXPK**

**SIZE**

## 1.18 NODATACHECK

### 3.2.1. NODATACHECK

When **TESTFILES** is used alone,it will first check the file by DATA (most powerful mode,but a bit slow as it needs to load some/all data of the file).Then if the type is still 'generic',it will use PATTERN checking as last chance.

adding NODATACHECK will force the test to use ONLY the pattern-matching check on the filename,which is quicker...but not always very precise.

Example:

-----

```
> dmcontrol testfiles S:startup-sequence
```

File S:startup-sequence : ASCII-Text

> dmcontrol testfiles S:startup-sequence NODATACHECK

File S:startup-sequence : Generic

explanation:as this file doesn't match one of the patterns (\*.doc,\*.readme etc...) supported by the ASCII-Text recognizer,it WON'T be recognized by the pattern-matching...simple,no?

## 1.19 LONGNAME

### 3.2.2. LONGNAME

By default, **TESTFILES** strips out the path of the tested file when printing the results. (I think it's more readable)

Set this switch to print the full path.

## 1.20 NOXPK

### 3.2.3. NOXPK

By default, **TESTFILES** XPK-loads the tested file (or part of it).

So all the 'XPK-packed' files will automatically be recognized correctly.

If, for some strange reason, you do not want to xpk-load, set this switch.

## 1.21 SIZE

### 3.2.4. SIZE

By default, **TESTFILES** loads the 5000 bytes of the file for testing.You can change this value if you want (in bytes)

Note:set it to -1 if for loading the whole file.

## 1.22 History

v1.9:

~~~~~

\* DMR\_SubTypesTable etc support... Output much enhanced I think :)

\* Removed the "PATT" switch (useless now).

v1.8:

~~~~~

\* DMR\_HookName support.

v1.7:

~~~~~

---

- \* Adapted for v2.0 datamaster library. (subtypes support, new functions)
- \* better output
- \* Changed switches XPK to **NOXPK** & SHORTNAME to **LONGNAME**
- \* Added bundle-reco support ( **CREATEBUNDLE** command).

(some beta versions)

~~~~~

v1.0:

~~~~~

Initial release.

---