

# DT-X400 Series

## Android 8.1 FLDroid Manual

This document describes how to develop  
an application using FLDroid.



No part of this document may be produced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of CASIO COMPUTER CO., LTD. in Tokyo Japan. Information in this document is subject to change without advance notice.

CASIO COMPUTER CO., LTD. makes no representations or warranties with respect to the contents or use of this manual and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose.

© 2018 CASIO COMPUTER CO.,LTD.

- Wi-Fi is a registered trademark of Wi-Fi Alliance.
- Android, Android Wear, Google, Google Play, Google Now and other marks are trademarks of Google LLC.
- Other company, product and service names used in this manual also may be trademarks or registered trademarks of others.

---

## Preface

This document describes how to develop an application using FLDroid.

# Index

<b>Chapter 1.</b>	<b>Overview</b>	<b>4</b>
1.1	About FLDroid	4
1.2	Software Configuration	5
1.2.1	FLDroid	5
1.2.2	Communication Engine	5
1.2.3	Progress Dialog	6
1.3	How to Use	7
1.3.1	Transports	7
1.3.2	Specifying transport	8
1.3.3	How to Start Communication	8
1.4	Developing Applications	9
1.4.1	Outline	9
1.4.2	Process Flow	9
1.4.3	Differences from Windows CE application (FLCE.exe)	11
<b>Chapter 2.</b>	<b>Intents</b>	<b>13</b>
2.1	Start and Abort Communication	13
2.1.1	Start Communication (Launch Communication Engine)	13
2.1.2	Abort Communication	13
2.2	Query and Reply Communication Status	14
2.2.1	Query Communication Status	14
2.2.2	Reply Communication Status	14
2.3	Finished Communication	15
<b>Chapter 3.</b>	<b>Parameters</b>	<b>16</b>
3.1	Rules of Specifying Parameters	16
3.2	Details of Commands and Options	18
3.2.1	Set Communication Environment	18
3.2.2	Send Files	18
3.2.3	Receive Files	20
3.2.4	Send File (Append)	21
3.2.5	Delete Files	22
3.2.6	Move Files / Change Filenames	23
3.2.7	Launch in Idle Mode	24
3.2.8	Hide Screen	24
3.2.9	Hide Transferring Filename	25
3.3	How to specify a pathname	26
3.4	Behavior when the file which doesn't exist is specified	26
3.5	Function and Display	27
<b>Chapter 4.</b>	<b>Exit Code</b>	<b>28</b>
4.1	Table of Exit Code	28

---

<b>Chapter 5.</b>	<b>Apendix</b>	<b>30</b>
5.1	Priority of Transport	30
5.2	How to Use ConnectivityManager	32
5.2.1	How to Get Connection Status	32
5.2.2	How to Watch Connection Status	33

---

# 1. Overview

## 1.1 About FLDroid

The purpose of FLDroid is to realize the functions of "FLCE" on Android OS.

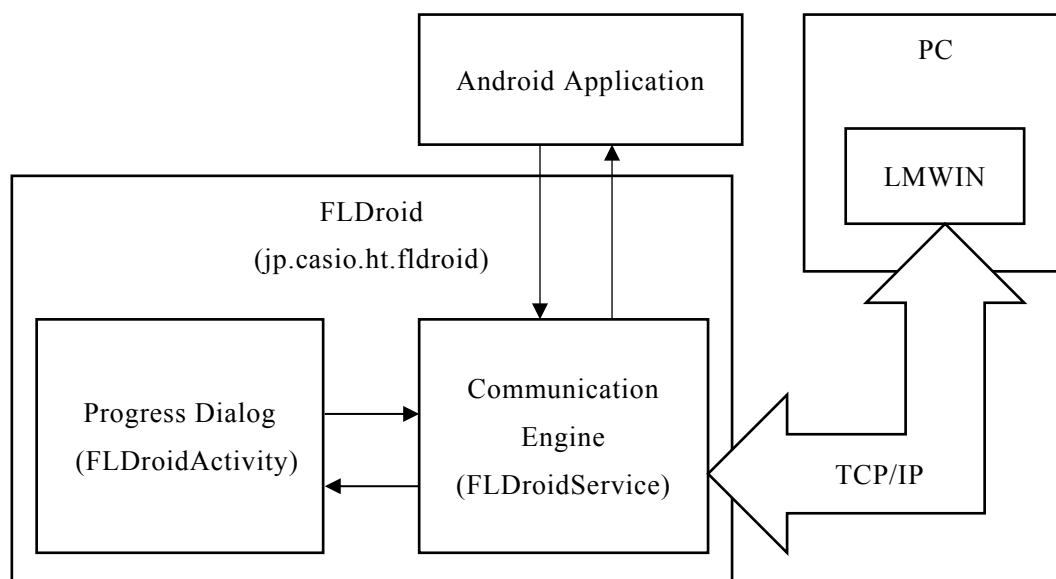
FLDroid is a tool to transfer files between PC and DT-X400. It transfers files in cooperation with the application "LMWIN" running on PC.

FLDroid runs as a service on Android OS and can transfer files via TCP/IP (LAN/WLAN) in cooperation with LMWIN by being called from an Android application.

Please refer to the Manual of LMWIN for the details of PC's application "LMWIN".

## 1.2 Software Configuration

Here is a configuration diagram of FLDroid.



### 1.2.1 FLDroid

FLDroid includes Progress Dialog (Activity) and Communication Engine (Service).

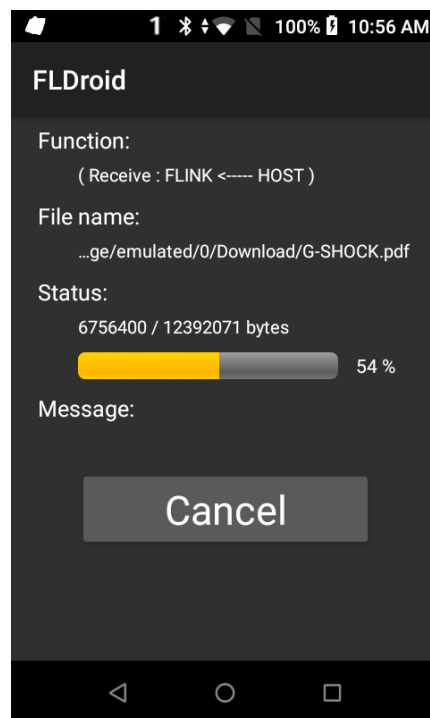
### 1.2.2 Communication Engine

Communication Engine is a Service makes actual communication. It communicates with LMWIN running on the PC by being called from an Android application. Applications using FLDroid communicates with a PC by calling this service.

## 1.2.3 Progress Dialog

Progress Dialog is an activity shows communication status. The communication status is indicated by this Activity when Hide Screen Command (/W) is not specified at the start of communication.

The communication status is not indicated when "/W" command is specified. This makes it possible to communicate in the background and/or to create an original Activity shows the communication status.





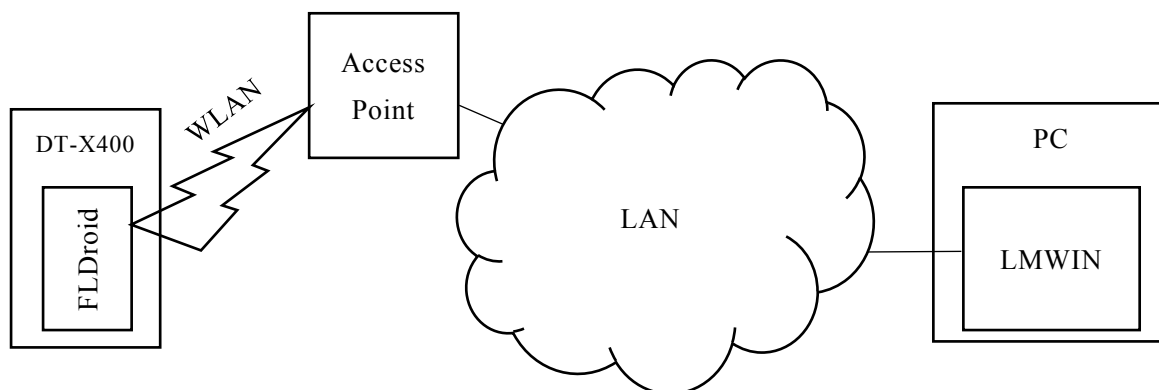
## 1.3 How to Use

### 1.3.1 Transports

There are following transports for FLDroid – LMWIN communication.

#### WLAN

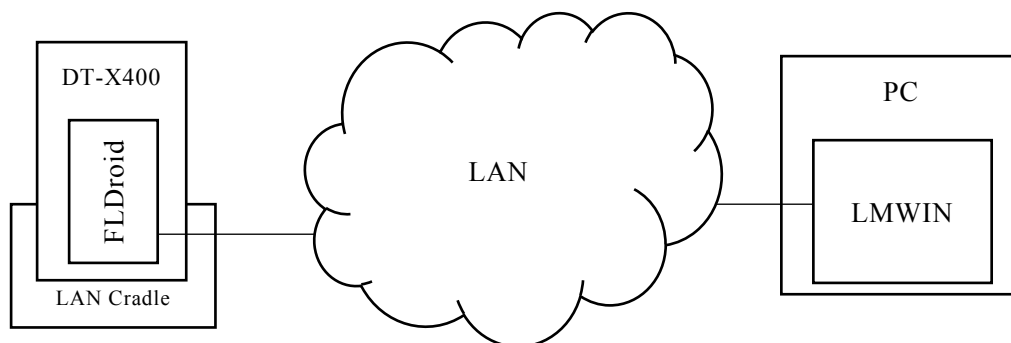
Connect FLDroid and LMWIN using WLAN.



- An access point and a LAN environment to connect DT-X400 to the LAN must be prepared.

#### LAN cradle

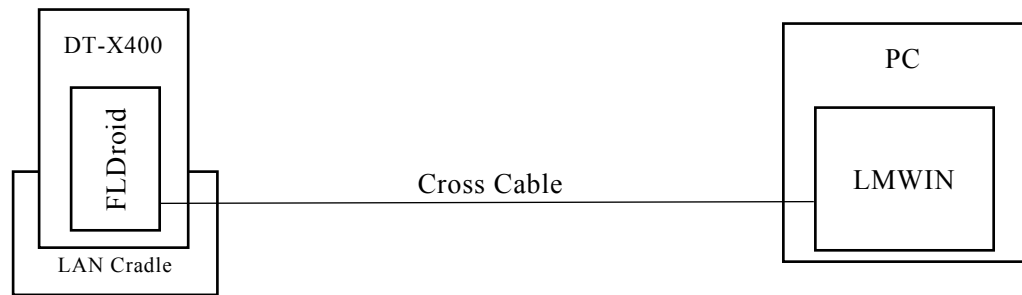
Connect FLDroid and LMWIN using LAN cradle.



- A LAN environment to connect DT-X400 to the LAN must be prepared.

## LAN cradle (Direct PC Connection)

Connect PC and LAN cradle directly using a LAN cable (cross cable).



- A LAN environment doesn't have to be prepared because PC and LAN cradle connected directory using a LAN cross cable.
- Use fixed IP addresses for PC and DT-X400 and they should be included in the same subnet.

### 1.3.2 Specifying transport

DT-X400 has some kinds of network transports such as WLAN / LAN cradle / WAN, and System changes active transport automatically.

Therefore, when using FLDroid in the status with multiple kinds of transports connected to networks, a transport must be specified for FLDroid to communicate using intended transport.

On FLDroid, an application can specify the transport by Set Communication Environment Command (/Y).

For example, the transport can be fixed to LAN cradle by the following command even when both WAN and LAN cradle are connected to networks.

```
/Y={ETHERNET, IP address, }
```

For Set Communication Environment Command, see "3.2.1 Set Communication Environment (p.18)".

For Priority of Transport, see "5.1 Priority of Transport (p.30)".

### 1.3.3 How to Start Communication

Please start communication after the network connection established. When starting the communication before established, FLDroid will wait for the connection for a certain period (60 seconds as default). When the connection wouldn't be established, Fldroid will exit in error.

To get and/or watch connection status, ConnectionManager can be used.

For the way to get and/or watch connection status, see "5.2 How to Use ConnectivityManager (p.32)".

For more information on ConnectivityManager, please refer to the Android Developer website.

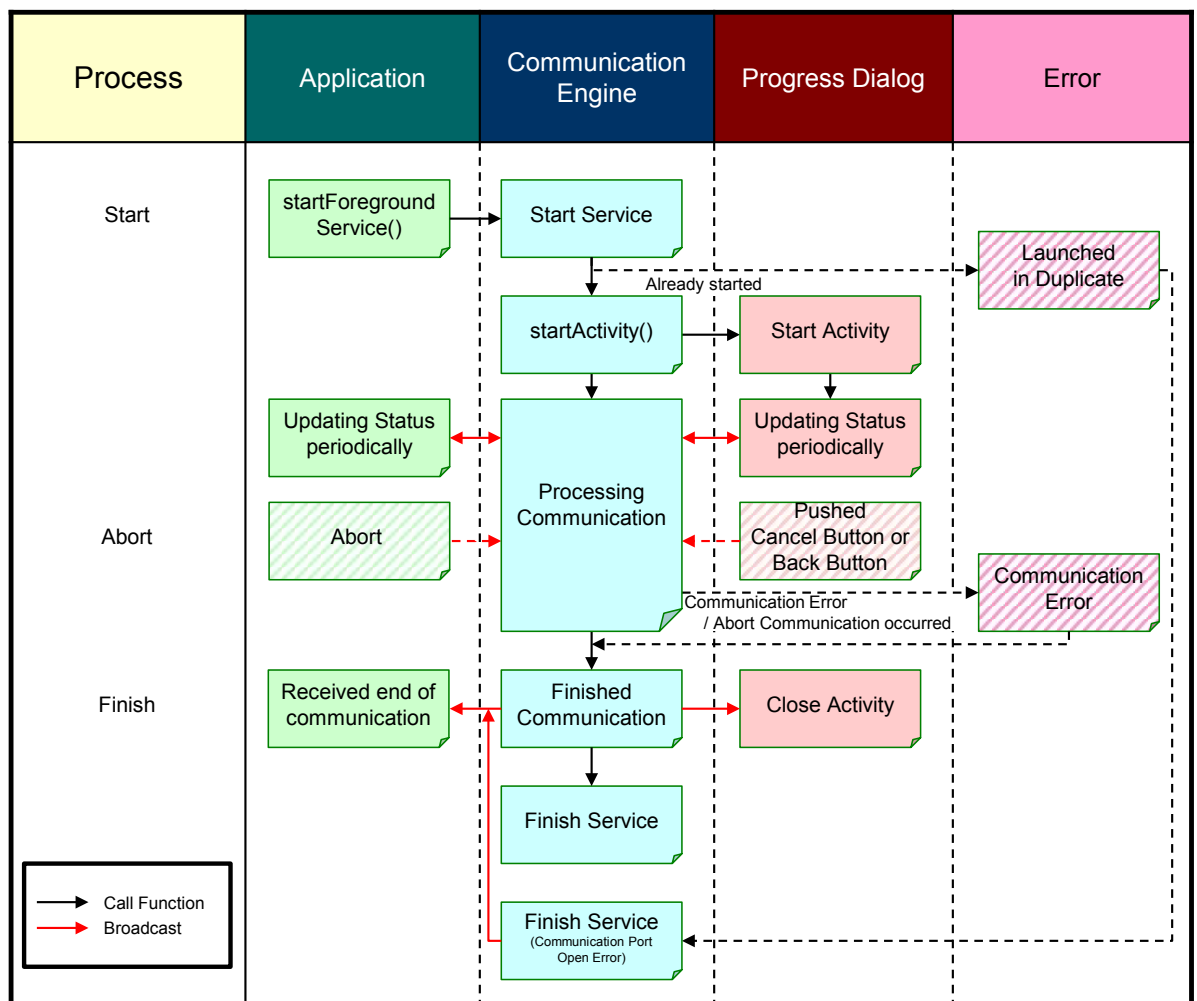
## 1.4 Developing Applications

### 1.4.1 Outline

- To use FLDroid, you must develop an application and call FLDroid from the application.
- startForegroundService() or startService() can be used to start communication.
- "The communications between the application and FLDroid" and "the end of the communication" are done via Broadcast.

### 1.4.2 Process Flow

Here is a process flow to call FLDroid from an application.



- "Application" can start a communication by calling "Communication Engine" using `startForegroundService()`.  
Call `startService()` instead when the compile API level is 25 or less.
- Only one communication can be processed in same time. When FLFroid is launched in duplicate "Communication Engine" broadcasts "Finish Service" and terminates immediately.
- When Hide Screen Command (/W) is not specified, "Communication Engine" launches an Activity of "Progress Dialog".
- "Communication Engine" communicates with "Application" and "Progress Dialog" via broadcasts and notifies the status of communication during the communication process.
- When the cancel button of "Progress Dialog" or back button is pushed, the communication is aborted.
- "Application" can abort the communication by sending a broadcast of "Abort" to "Communication Engine".
- "Application" and "Progress Dialog" receive a broadcast "Finished Communication" from "Communication Engine" when the communication is finished.
- "Progress Dialog" finishes soon when it receives "Finished Communication" from "Communication Engine".

### 1.4.3 Differences from Windows CE application (FLCE.exe)

When comparing FLDroid and Windows CE application (FLCE.exe), there are the following differences due to differences in OS etc.

#### Support status of FLCE commands

Here is the support status of FLCE commands on FLDroid.

Type	Function	Command	FLCE	FLDroid	Note
Settings Command	Set Communication Environment	/Y	x	x	
Common Commands	Send Files	/S	x	x	
	Receive Files	/R	x	x	Option 'O' is ignored.
	Send File (Append)	/A	x	x	
	Delete Files	/D	x	x	
	Move Files	/N	x	x	
	Send Date and Time	/T	x	-	
	Launch in Idle Mode	/Y	x	x	
Optional Commands	Hide Screen	/W	x	x	
	Hide Transferring Filename	/F	x	x	

#### Support status of LMWIN commands

Here is the support status on FLDroid of commands specified by LMWIN.

Function	Command	Support	Note
Send Files	/S	x	Option 'O' is ignored. Overwriting read-only files makes an error.
Receive Files	/R	x	
Send File (append)	/AS	x	Option 'O' is ignored. Overwriting read-only files makes an error.
Receive File (Append)	/AR	x	
Delete	/D	x	Option 'O' is ignored. Read-only files are also deleted.
Move Files / Change Filenames	/N	x	
Ring Buzzer	/B	x	
Set Date and Time	/T	x	
Show String	/P	x	
End Session	//	x	
Launch Child Process	/C	x	Function of LMWIN
Get Disk Information	/I	x	The disk capacity is in megabytes.
Get File Information	/I	x	
Set File Information	/X	x	File attributes can't be changed.
Format	/F	x	

## Screen for inputting commands

FLDroid has no screens for inputting commands.

To start communication by FLDroid, you must develop an application and call FLDroid from it.

## Folders which can be accessed

FLDroid can access only following folders and sub-folders of them.

Drive Name	Folder Name
Internal storage	/storage/emulated/0/
SD card	/storage/sdcard1/
USB drive	/storage/usbotg/

## Handling read-only files on Android devices

The read-only attribute of files can't be changed on Android OS. So that option 'O' is ignored on receiving files. Overwriting read-only files makes "File Write Error".

The file can be deleted even when it is read-only. When overwriting a read-only file, it can be realized by deleting the corresponding file and then writing it.

## Unit of the disk capacity in command "Get Disk Information" (/I)

The disk capacity by the command "Get Disk Information" (/I) from LMWIN is in megabytes. (In case of FLCE.exe, it is in Bytes.)

## Changing the file attributes by the command "Set File Information" (/X)

FLDroid does not support changing file attributes by the command "Set File Information" (/X).

## 2. Intents

The application and FLDroid use intents to communicate through `startForegroundService()` or `sendBroadcast()` each other.

This section describes the intents used in the communication process.

### 2.1 Start and Abort Communication

#### 2.1.1 Start Communication (Launch Communication Engine)

An application uses Communication Engine to start a communication. To launch Communication Engine, the application calls `startForegroundService()` using the following intent.

Name		Type		Value / Description
packageName	Mandatory	packageName	String	"jp.casio.ht.fldroid" specified in <code>setClassName()</code>
className	Mandatory	className	String	"jp.casio.ht.fldroid.FLDroidService" specified in <code>setClassname()</code>
arg	Mandatory	Extra	String	Parameter String described in "3.Parameters (p.16)"
uuid	Optional	Extra	String	UUID generated and specified
CONWAIT	Optional	Extra	int	Session establishment timeout time (seconds) default : 60 seconds
RECWAIT	Optional	Extra	int	Reception wait timeout time (seconds) default : 30 seconds

Note: about uuid

The intents "Finished Communication" and "Reply Communication Status" include this value and the Application can confirm the target of broadcasts.

Please specify a UUID when starting a communication.

UUID can be made as follows.

```
String uuid = UUID.randomUUID().toString();
```

#### 2.1.2 Abort Communication

The application can abort the communication by broadcasting this intent.

Name		Type		Value / Description
Action	Mandatory	Action	String	"jp.casio.ht.fldroid.abort"

## 2.2 Query and Reply Communication Status

The application can query the communication status while processing communication.

The application queries the status by broadcasting "Query Communication Status".

Communication Engine replies by broadcasting "Reply Communication Status".

### 2.2.1 Query Communication Status

The application can query the communication status by broadcasting this intent.

Communication Engine replies "Reply Communication Status" for the response.

Name		Type		Value / Description
Action	Mandatory	Action	String	"jp.casio.ht.fldroid.querystatus"

### 2.2.2 Reply Communication Status

Communication Engine replies the communication status by broadcasting "Reply

Communication Status" when it receives "Query Communication Status".

Name		Type		Value / Description
Action	Mandatory	Action	String	"jp.casio.ht.fldroid.replystatus"
function	Optional	Extra	String	Processing Function It isn't included when there are no strings to show.
filename	Optional	Extra	String	Name of Transferring File It isn't included when there are no strings to show.
bytes	Optional	Extra	int	Transferred Bytes It isn't included when there are no strings to show.
totalbytes	Optional	Extra	int	Total Bytes of Transferring File It isn't included when there are no strings to show.
progress	Optional	Extra	int	Transferred Progress in Percent It isn't included when there are no strings to show.
message	Optional	Extra	String	String Specified by Command "/P" It isn't included when there are no strings to show.
uuid	Mandatory	Extra	String	"uuid" specified in "Start Communication".



## 2.3 Finished Communication

Communication Engine finishes a communication when the communication completed successfully, something error occurred or abort was specified. In that case, Communication Engine broadcasts this intent to notify the communication finished.

Name	M/O	Type		Value / Description
Action	Mandatory	Action	String	"jp.casio.ht.fldroid.finished"
result	Mandatory	Extra	int	Exit Code For the detail, see "4 Exit Code (p.28)".
uuid	Mandatory	Extra	String	"uuid" specified in "Start Communication".

## 3. Parameters

### 3.1 Rules of Specifying Parameters

There are "Settings Command", "Action Commands" and "Optional Command" in Parameters. Multiple "Action Commands" can be specified in a Parameter String. However, Parameter String must be 250 characters or shorter. When multiple "Action Commands" are specified, they are processed in order from leading one. When an error occurs during processing intermediate "Action Commands", FLDroid stops with the error immediately and the following won't be processed. There are other rules for specifying Parameters as following.

1. All Parameters must be separated by space character.
2. "Settings Command" can be specified only once in the lead of the string.
3. "Action Commands" can be specified up to 20 items.
4. "Options" must be specified after "Command" without any spaces.
5. When multiple "Options" are specified, they must be specified continuously without any spaces.
6. "Options" can be specified in any orders.
7. "Commands" and "Options" can be specified in both upper and lower cases.

#### Command Types

Type	Function	Command	Specifiable Options	Example
Settings Command	Set Communication Environment	/Y	none	/Y={LAN,192.168.0.1,}
Action Commands	Send Files	/S	O,R	/SOR
	Receive Files	/R	O,R	/ROR
	Send File (Append)	/A	none	/A
	Delete Files	/D	O,R	/D
	Move Files	/N	none	/N
Optional Commands	Launch in Idle Mode	/Y	Script Filename	
	Hide Screen	/W	none	/W
	Hide Transferring Filename	/F	none	/F

#### Restrictions of Optional Commands

1. The Optional Commands "/W" and "/F" cannot be specified together because of their functionality.
2. "Optional Commands" must be specified before or after "Action Commands".  
When specified between "Action Commands", the operation is not guaranteed.

## Outline of Options

1. O(Over Write): Forced overwriting of read-only files  
When this option is specified, the overwrite operation is also performed for the read-only files.  
When this option is not specified, the overwrite operation to the read-only files fails.  
The read-only attribute of the files can't be changed on Android OS. So that, "O" option is ignored in access to the files on the Android device. The behavior when accessing read-only files is as follows.  
"File Write Error" will occur when trying to overwrite to the read-only files.  
The read-only files will be deleted when trying to delete them even without option "O".
2. R(Recursive Call):  
All files under specified directory will be targets of the process. When there are subdirectories, also they will be targets of the process.  
The depth of subdirectories is limited up to 16.  
Only files specified in path name will be targets when this option is not specified.

## Examples

Here are some examples of specific parameters.

These examples assume to communicate with a PC with IP address of 192.168.0.1.

1. Launch in Idle Mode (No Script File)  
Connect to the PC and launch in Idle Mode. (The script file specified in LMWIN on the PC will be executed.)  

```
/Y={LAN,192.168.0.1, }
```
2. Launch in Idle Mode (Script File specified)  
Connect to the PC and execute 0001.scr. (0001.scr on the PC will be executed.)  

```
/Y={LAN,192.168.0.1, } "0001.scr"
```
3. Send Files  
Send a file "/storage/emulated/0/FLDroid/FLDroid.dat" to a folder "c:\FLDroid" on the PC.  

```
/Y={LAN,192.168.0.1, } /S  
"/storage/emulated/0/FLDroid/FLDroid.dat" "c:/FLDroid"
```
4. Receive Files  
Receive a file "C:\FLdroid\FLDroid.dat" on the PC to a folder  
"/storage/emulated/0/FLDroid".  

```
/Y={LAN,192.168.0.1, } /R "c:/FLDroid/FLDroid.dat"  
"/storage/emulated/0/FLDroid "
```

## 3.2 Details of Commands and Options

The specification and meaning of Parameter String are described here.

In the following explanation, the parameter enclosed by [] indicates what can be specified arbitrarily.

Other parameters must be specified.

### 3.2.1 Set Communication Environment

#### Calling Sequences

```
/Y={ Transport, IP address, }
```

#### Explanation

Specify the transport and the IP address of the host PC used for communication.

Be sure to write this command first. When it is used in other places, Parameter Error will occur.

#### Parameters

##### *Transport*

Specify the route of the network to use for the communication. One of the following can be specified.

LAN	: FLDroid uses the active network.
ETHERNET	: FLDroid uses a LAN cradle to connect a network.
WIFI	: FLDroid uses WLAN to connect a network.

##### *IP address*

Specify the IP address of the host PC.

### 3.2.2 Send Files

#### Calling Sequences

```
/S[Option] SourceFilePathName [SourceFilePathName] [...]  
TargetDirectoryPathName
```

#### Explanation

This is a function to transfer files existing on DT-X400 to the PC.

When the sent files already exist on the PC before transferring, they will be overwritten.

When the directory specified as the target doesn't exist, it will be created automatically.

SourceFilePathNames are checked first, and when one of them doesn't exist, FLDroid will finish in error. (In this case, also existed files won't be transferred.)

The progress rate is shown while the files are transferred.

## **Parameters**

### ***Command***

/S(Sending) : Sending Files Process

### ***Option***

O(Over Write) : Forced overwriting of read-only files

When this option is specified, even the files are read-only, they will be overwritten.

When this option isn't specified, when overwriting read-only files are occurred, FLDroid exit in error.

R(Recursive Call) :

All files under the directory specified in SourceFilePathName are included in targets to send. When there are sub-directories, the name of sub-directories will be added to the filename.

The depth of directories is limited up to 16.

Even when this option is specified, specify SourceFilePathName in full-path.

When this option isn't specified, only the files specified by SourceFilePathName will be the targets.

### ***SourceFilePathName***

Specify files which exist on DT-X400 in full-path.

Wild-card can be used as the filename.

Use "\*.\*)" as the filename to specify all files as the target.

### ***TargetDirectoryPathName***

Specify TargetDirectoryPathName as the last of the parameters to this command.

When specified directory doesn't exist, a new directory will be created with the specified name.

Follow the naming rules of the target PC's OS.

## 3.2.3 Receive Files

### Calling Sequences

```
/R[Option] SourceFilePathName [SourceFilePathName] [...]
TargetDirectoryPathName
```

### Explanation

This is a function to receive files which exist on the PC. The file names are specified by SourceFilePathNames.

When the receiving files already exist, they will be overwritten.

When the directory specified as the target doesn't exist, it will be created automatically.

The progress rate is shown while the files are received.

### Parameters

#### *Command*

/R(Receive) : Receiving Files Process

#### *Option*

O(Over Write) : Forced overwriting of read-only files

The option "O" is ignored on Android OS. "File Write Error" will occur when read-only files are overwritten.

R(Recursive Call) :

All files under the directory specified in SourceFilePathName are included in targets to receive. When there are sub-directories, the name of sub-directories will be added to the filename.

When this option isn't specified, only the files specified by SourceFilePathName will be the targets.

Even when this option is specified, specify SourceFilePathName in full-path.

#### *SourceFilePathName*

Specify files which exist on the PC in full-path.

Wild-card can be used as the filename.

Use "\*. \*" as the filename to specify all files as the target.

Follow the naming rules of the PC's OS.

#### *TargetDirectoryPathName*

Specify TargetDirectoryPathName as the last of the parameters to this command.

Specify the directory that stores received files as TargetDirectoryPathName.

When specified directory doesn't exist, a new directory will be created with the specified name.

## 3.2.4 Send File (Append)

### Calling Sequences

```
/A SourceFilePathName TargetFilePathName
```

### Explanation

This function is to send the contents of the file specified by SourceFilePathName from DT-X400 and append it to the file specified by TargetFilePathName on the PC.

When the file specified by TargetFilePathName doesn't exist, it will be created automatically.

The timestamp of the target file will be the system time of the PC.

The file is appended in binary mode.

(When EOF code exists the last of the target file, it is appended after that.)

The progress rate is shown while the files are transferred.

### Parameters

#### *Command*

/A(Append) : Sending and Appending a File Process

#### *SourceFilePathName*

Specify a file which exists on DT-X400 in full-path.

Wild-card cannot be used as the filename.

#### *TargetFilePathName*

Specify the full path of a file which exists on the PC and will be appended.

When the file doesn't exist, the file will be created in the specified file name.

Wild-card cannot be used as the filename.

Follow the naming rules of the target PC's OS.

## 3.2.5 Delete Files

### Calling Sequences

```
/D[Option] PathNameToDelete [PathNameToDelete] [...]
```

### Explanation

This function is to delete PC's files and/or directories specified.

The progress rate is not shown.

### Parameters

#### *Command*

/D(Delete) : Deleting files and/or directories specified by PathNameToDelete

#### *Option*

O(Over Write) : Forced overwriting of read-only files

This option specifies to delete files and/or directories even when they are read-only.

When this option isn't specified, when deleting read-only files and/or directories are occurred, FLDroid exits in error.

R(Recursive Call) :

Directories specified by PathNameToDelete and all directories and files under it will be the targets for deletion.

The depth of directories is limited up to 16.

When this option is specified, specify PathNameToDelete in full-path.

When this option is not specified, only files specified by PathNameToDelete will be the targets for deletion.

#### *PathNameToDelete*

Without The Option "R"

Specify the full path name of a file which exists on the PC and you want to delete.

Wild-card can be used as the filename.

Use "\*.\*)" as the filename to specify all files as the target.

With The Option "R"

Specify the full path name of a directory which exists on the PC and you want to delete.

Follow the naming rules of the target PC's OS.



## 3.2.6 Move Files / Change Filenames

### Calling Sequences

```
/N SourcePathName DestinationPathName
```

### Explanation

This function is to move a file specified by SourcePathName to a file specified by DestinationPathName on the PC.

When the DestinationPathName is a directory, the filename won't be changed.

(SourcePathName's filename will be used.) When the DestinationPathName is a file, the file name will be changed to DestinationPathName's filename.

The progress rate is not shown.

### Parameters

#### *Command*

/N(reName) : Moving a file specified by SourcePathName to DestinationPathName

#### *SourcePathName*

Specify the full path of a file which exists on the PC and you want to move.

Wild-card cannot be used as the filename.

Follow the naming rules of the target PC's OS.

#### *DestinationPathName*

Specify the full path of the destination on the PC.

When there is already a same file of DestinationPathName, FLDroid ends in error.

Specify '/' at the end of the path name when DestinationPathName is a directory.

The filename will be changed as specified when DestinationPathName is a file.

When specified directory doesn't exist, a new directory will be created with the specified name.

Wild-card cannot be used as the filename.

Follow the naming rules of the target PC's OS.

## 3.2.7 Launch in Idle Mode

### Calling Sequences

```
/Y={ Transport, IP address, } [ScriptFileName]
```

Note:

Action Command other than "/Y" cannot be specified.

### Explanation

Idle Mode is a mode that FLDroid gives a right to request to a PC and acts according to the requests from the PC.

When FLDroid is launched in this mode, the action commands other than "/Y" cannot be specified. (When other options are specified, FLDroid will launch in normal mode or will finish in parameter error when ScriptFileName is specified.)

This function finishes when the indication to finish is received.

When the ScriptFileName is specified, FLDroid acts according to the contents of the script file on the PC.

When the script file doesn't exist on the PC, FLDroid will return an error.

### Parameters

#### *ScriptFileName*

Specify a script file which exists on the PC.

The longest script file name is 8 characters for the file name and 3 characters for the extension.

ScriptFileName must be enclosed by "".

## 3.2.8 Hide Screen

### Calling Sequences

```
/w
```

### Explanation

When this command is specified, Progress Dialog won't be shown while FLDroid is running.

## 3.2.9 Hide Transferring Filename

### Calling Sequences

```
/F
```

### Explanation

When this command is specified, transferring filename won't be shown on the progress dialog while FLDroid is running.

### 3.3 How to specify a pathname

1. A pathname must be enclosed by "". A pathname must be 255 characters or shorter including "".
2. Please follow the following rules about the delimiter to express a pathname.  
When you write a pathname on the parameter strings of FLDroid, use "/" even the pathname is a filename on the PC.  
When you write a pathname for LMWIN on the PC, use "\" even the pathname is a filename on the Android device.
3. Please follow the following rules about the drive letter to express a pathname.  
Don't use a drive letter to express the pathname of the Android device. Specify it from the root directory. (It is the same to specify a pathname of the Android device from the LMWIN on the PC.)  
A drive letter must be specified to express a pathname of a file or a directory on the PC from the Android device.  
To specify a device on the Android device for formatting or getting disk info from the LMWIN on the PC, the drive letter means as follows.

Drive	Drive Name
C:	Internal Memory
D:	USB drive
E:	SD card
F:	Internal Memory

### 3.4 Behavior when the file which doesn't exist is specified

When the file or the path which does not exist on the PC is specified, FLDroid acts as follows.

Command	Behavior
Receive	When even one of the specified path names does not exist, FLDroid will finish abnormally. (Existing files won't be received either.)
Delete	When some of the specified path names do not exist, the pathnames will be ignored. (All other paths which exist will be deleted.)
Move	When the specified path name does not exist, FLDroid will finish abnormally. (Communication process won't be performed.)
Send, Send (Append)	The file will be created newly.

## 3.5 Function and Display

Following contents are shown in Progress Dialog when each function is performed.

No.	Function(Command on Protocol)	Specify from FLDroid	Request from PC	Remarks
1	Send Files	C	C	
2	Receive Files	C	C	
3	Append File	C	C	
4	Delete Files or Directories	A	B	
5	Move Files or Change Filenames	A	B	
6	Create Directory	-	B	
7	Set Date and Time	-	A	
8	Request Date and Time	-	A	
9	Show String	-	D	
10	Ring Buzzer	-	A	
11	Get File Information	-	A	
12	Set File Information	-	A	
13	Get Disk Information	-	A	
14	Get Session ID and System Information			No Display (Internal Command)
15	Notify Idle Mode			
16	Indication to Finish			

Status :

- A Currently performing or requesting command is shown.
- B The name of the processing file or directory is shown in addition to A.
- C The name of the transferring file and the progress are shown in addition to A.
- D The text message sent from the PC is shown.

## 4. Exit Code

FLDroid returns Exit Code described in the following table when it finishes processing. Exit Code is returned as the "result" in the broadcast "Finished Communication". The Application must receive the broadcast "Finished Communication" and do the appropriate action referring to the "result".

### 4.1 Table of Exit Code

Category Code (Upper byte) means the category of the error and Detail Code (Lower byte) means the detail of the error.

Error Code		Meaning	Cause	Workaround
Category Code	Detail Code			
00H	00H	Completed Successfully	Normal	-
DCH-F5H	00H	Completed Successfully	The format command for the drive from 'A:' to 'Z:' was received from the PC.	-
F6H	00H	Completed Successfully	The PC has instructed to turn off the power.	-
F7H	00H	Completed Successfully	The PC has indicated to reboot.	-
F8H	00H	Aborted	Aborted by pushing the abort button of the PC or the device.	Re-execute as necessary.
01H	00H	Protocol Error	Data Error (Data error has occurred on the communication line.)	Confirm the connection of the communication line.
02H	80H	File Not Found	A nonexisting file was specified.	Confirm specified files and directories.
02H	81H	Delete Current Directory Error	Tried to delete current directory.	Confirm the directory trying to delete.
02H	82H	Write File Error	The file couldn't be written for some reasons.	Confirm whether the file can be written.
02H	83H	Read File Error	The file couldn't be read for some reason.	Confirm whether the file can be read.
02H	84H	Access Readonly File Error	Tried to overwrite or to delete a read-only file.	Specify the other filename or disable the read-only attribute.
0FH	01H	Parameter Error (Sender)	The parameters were wrong.	Confirm the parameters.
0FH	80H	Parameter Error (Receiver)	The parameters were wrong.	Confirm the parameters.
0FH	02H	Parameter Length Error	The parameter string was too long.	Set the parameter string 250 characters or shorter.

A0H	10H	Open Port Error	FLDroid is already running.	Stop running FLDroid.
A0H	20H	Disconnected Communication Line Error	An error occurred in Socket function.	Confirm whether it's in the communicable state.
A0H	30H	Wait Connection Timeout Error	The connection was not completed within Session establishment timeout time after startup.	Confirm whether it's in the communicable state.

Note:

Please also refer to LMWIN's error code table when the Exit Code can't be found on this table.

(LMWIN's error code may be transmitted and returned. In this case, "Category Code" will be LMWIN's "Error Contents" and "Detail Code" will be LMWIN's "Error Code".)

## 5. Appendix

### 5.1 Priority of Transport

When multiple kinds of transports are connected to networks, System changes active transport automatically by the situation of the connection. Normally, an Application communicates to this active transport.

System changes the active transport according to the following rules.

#### 1. Priority of Transports

Following is the order of the priority of transports from higher.

LAN cradle > WLAN > WAN

#### 2. Connection to The Internet

The transport connected to the Internet has higher priority.

When there is a connected transport which has higher priority than the highest prioritized transport connected to the Internet, the connection will be retained. FLDroid can use this transport by specifying it as the transport in Set Communication Environment Command (/Y).

When there is no transport connected to the Internet, the highest connected transport becomes active. In this case, other connection will also be retained and can be used by specifying it as the transport in Set Communication Environment Command (/Y).

Based on the above rule, the table of the next page shows the connection status of the LAN (LAN cradle) / WLAN / WAN, the active transport and the specifiable transport.



The Internet Connection						Active Transport	Specifiable Transport by /Y		
Connected			Not Connected				LAN	WLAN	WAN
LAN	WLAN	WAN	LAN	WLAN	WAN				
						-			
					x	WAN			x
		x				WAN			x
				x		WLAN		x	
				x	x	WLAN		x	x
		x		x		WAN		x	x
	x					WLAN		x	
	x				x	WLAN		x	
	x	x				WLAN		x	
			x			LAN	x		
			x		x	LAN	x		x
		x	x			WAN	x		x
			x	x		LAN	x	x	
			x	x	x	LAN	x	x	x
		x	x	x		WAN	x	x	x
	x		x			WLAN	x	x	
	x		x		x	WLAN	x	x	
	x	x	x			WLAN	x	x	
x						LAN	x		
x					x	LAN	x		
x		x				LAN	x		
x				x		LAN	x		
x				x	x	LAN	x		
x		x		x		LAN	x		
x	x					LAN	x		
x	x				x	LAN	x		
x	x	x				LAN	x		

## 5.2 How to Use ConnectivityManager

This section describes how to get and watch connection status by ConnectivityManager. For more information about ConnectivityManager, please refer to the Android Developer website.

### 5.2.1 How to Get Connection Status

The connection status of DT-X400 can be gotten by using ConnectivityManager.

- To use ConnectivityManager, the permission "android.permission.ACCESS\_NETWORK\_STATE" is needed. A manifest of an application must include the following definition.

```
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

- You can get an Instance of ConnectivityManager by using getSystemService() as follows.

```
ConnectivityManager cm =
    (ConnectivityManager) getSystemService (Context.CONNECTIVITY_
SERVICE);
```

- By using getAllNetworks(), you can get a list of connected networks. This list includes information about whether a specific network is connected.

```
Network[] networks = cm.getAllNetworks();
for (Network network : networks) {
    NetworkInfo info = cm.getNetworkInfo(network);

    .
    .
    .
}
```

- By using NetworkCapabilities.hasTransport(), it can be tested whether specific transport is supported on the network.

Following is testing whether LAN (LAN cradle) supported.

```
NetworkCapabilities capa =
cm.getNetworkCapabilities(network);
if (capa.hasTransport(capa.TRANSPORT_ETHERNET)) {
    // LAN cradle
}
```

- info.geExtraInfo() returns the extra information such as the name of access point.
- The values used in hasTransport() and the response of getExtraInfo() are as follows.

	hasTransport()	getExtraInfo()
LAN cradle	TRANSPORT_ETHERNET	MAC address
WLAN	TRANSPORT_WIFI	Access Point Name
WAN	TRANSPORT_CELLULAR	APN

- By using getActiveNetwork() / getActiveNetworkInfo(), you can get the status of active network.

```
Network network = cm.getActiveNetwork();
NetworkInfo info = cm.getActiveNetworkInfo();
```

## 5.2.2 How to Watch Connection Status

Callback functions can be registered by using registerNetworkCallback().

ConnectivityManager calls these functions when the status of the network changed. An application can watch the connection of the network by implementing the Callback functions.

To finish use of the Callback functions, use unregisterNetworkCallback().

Following shows how to register callback functions by registerNetworkCallback().

```
NetworkRequest.Builder builder = new
NetworkRequest.Builder();
NetworkRequest nr = builder.build();
networkCallback = new NetworkCallback();
cm.registerNetworkCallback(nr, networkCallback);

        .
        .
        .

class NetworkCallback extends
ConnectivityManager.NetworkCallback {
    // Implement Callback Functions here.
}
```