

General Characteristics of the Subscriber Identity Module File System

It used to be you could ask the question “who has a mobile phone?” in a group of people and relatively few hands would go up, now the question is more along the lines of “Who doesn’t have a mobile phone?” or even “Who has more than one mobile phone?”. Indeed one need only take a trip to the local shopping mall to see literally hundreds of people with tiny electronic devices stuck to their ears or the blinking blue lights on headsets to realize that mobile phones are an inseparable part of modern society.

This paper seeks to give an overview of the general file system structure of the subscriber identity module-the smart card that is present in GSM mobile phones. The paper will discuss the layout of the file structure and the some of the various commands used by the mobile equipment to access this file system.

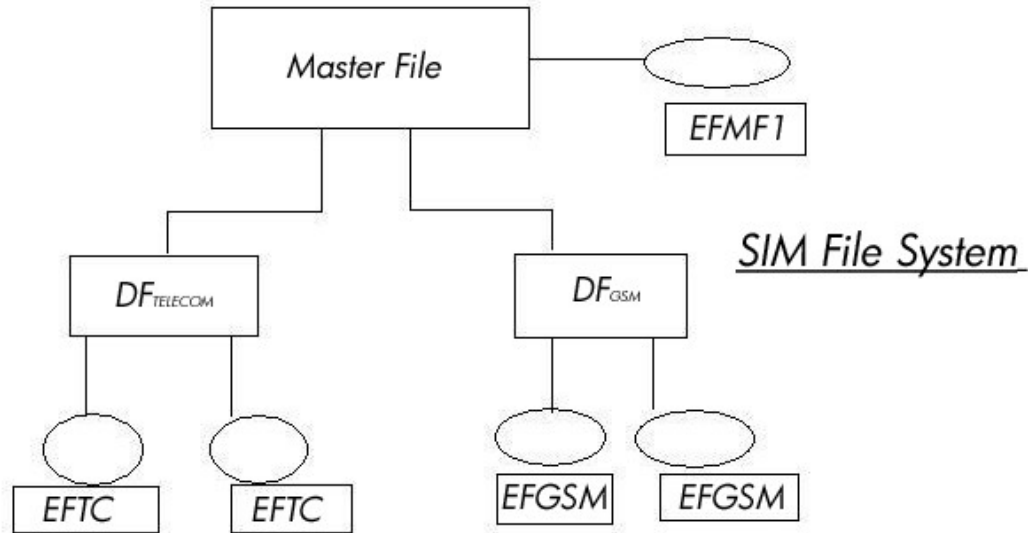
What is a Subscriber Identity Module?

A subscriber identity module or SIM is a smart card that is designed to fit into mobile phone. It provides the identification of a user to a network, allowing him or her to access such services as telephony, email, internet and text messaging. The SIM card contains a microcomputer as well as a certain amount of memory to process commands (Random Access Memory or RAM), and to store user files (Electronically Erasable Programmable Read Only Memory or EEPROM). The SIM also contains an amount of Read Only Memory (ROM) which stores the cards operating system.

When the SIM card is activated the microcomputer loads the operating system from ROM into the RAM of the card and processes commands as requested by the mobile equipment (ME) or card access device (CAD).

SIM Memory Structure

The SIM memory structure is composed of directories that can be said to be roughly analogous to the directories of a traditional computer hard disk drive. These directories are spelled out in detail in GSM 11.11 and 11.14. The file system may be comprised of the following basic forms: a master file (MF), a directory file (DF) and an elementary file (EF).



The Master File

The Master File or MF is the root of the file system. It is analogous to the root directory or “/” in the Linux file system; there is only one MF. An MF may contain one Dedicated File (DF) or many DFs and it may or may not contain one or many Elementary Files (EF).

The master file can be identified by the 2 byte file identifier (and indeed any file can be identified by this 2 byte sequence) of 3F00. This identifier is reserved only for the MF.

It should be noted that there is one EF directly beneath the MF in the hierarchy. This EF identified by the marker EFMF1 in the graphic above is the Integrated Circuit Card Identity (EF_{ICCID}). This EF contains the unique serial number of each individual SIM card and can be used to personalize the ME to the SIM¹. The use of the EF_{ICCID} is open to the individual manufacturer and operator. The EF_{ICCID} is the electronic version of the SIM Serial number recorded on the face of the SIM card body. The SIM Serial Number and EF_{ICCID} can be used by an examiner to identify the origin of the SIM where evidence was obtained².

Dedicated Files

The term Dedicated File is perhaps a bit confusing since the Dedicated File is more akin to a container or a sub directory rather than an actual file in the traditional sense. A dedicated file can also be identified by a two byte identifier. This identifier is assigned by the DF or the MF that contains it. The DF can also be referenced by a name that is between one and sixteen bytes long. The naming conventions for the DF name can be found in the ISO 7816-5 specification³. Two Dedicated Files of interest are the DF_{GSM} and the DF_{TELECOM}.

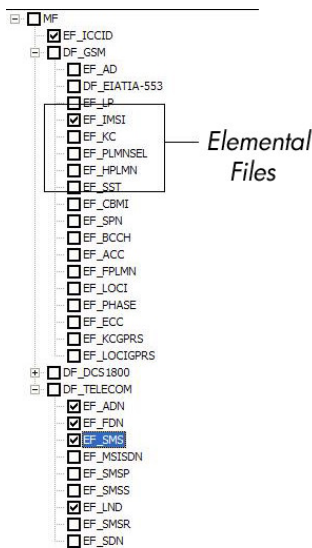
The DF_{TELECOM} contains more common telecom service features and can be used for other telecom applications in multipurpose SIMs⁴. The phonebook EF which falls under this directory is an example of a more general telecom application. The DF_{TELECOM} file can be identified by the 2 byte identifier of 7F10.

The DF_{GSM} contains applications for the GSM900 and GSM1800 MHz respectively. This directory contains EFs that are exclusive to GSM networks. The DF_{GSM} file can be identified by the 2 byte identifier of 7F20.

Elementary Files

Elementary Files (EF) sit below the Dedicated Files in the file system hierarchy (with the exception of the aforementioned EFICCID). These are the files that contain the actual data. An analogy to familiar computer file system terminology would be to say that the EF represents the leaf node of the file system.

Each directory has assigned elementary files that are designed to hold data for a specific use.



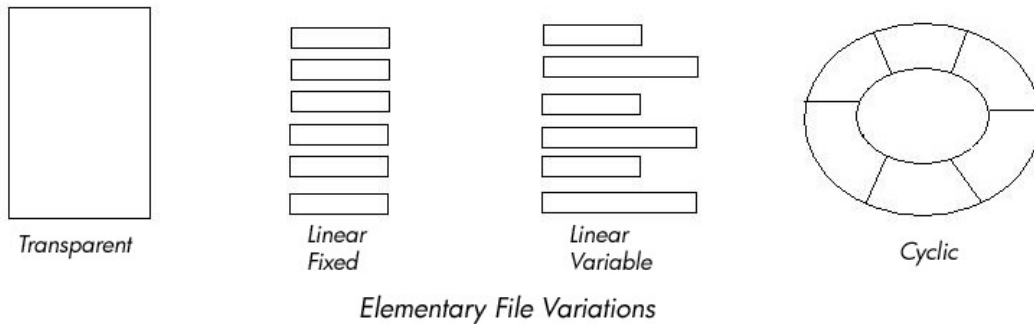
The elementary files do not actually have names per se as they are in electronic form. The names and acronyms assigned to the elementary files are done so to assist the reader in identifying a particular EF that has particular information. Electronic addresses are used instead and can take the form as shown below:

$$\text{ICCID Address} = 3F002FE2^4$$

The EF may hold only one record of information or it may hold many, where a record may be defined as a small unit of information stored on a SIM and consisting of a string of variable size.⁵

There are four variations of elementary files.

- Transparent
- Linear, fixed-length record
- Linear, variable-length record
- Cyclic



A transparent file can be seen as a string of bytes. When reading or updating, the sequence of bytes to be acted upon is referenced by a relative address (offset), which indicates the start position (in bytes), and the number of bytes to be read or updated. The first byte of a transparent EF has the relative address '00 00'. The total data length of the body of the EF is indicated in the header of the EF. This structure was previously referred to as "binary" in GSM⁶. The transparent file structure consists of a header followed by a body that contains a sequence of bytes.

The linear fixed-length elementary file is a sequence of records all having an identical (fixed) length. The first record is record number 1. The length of a record as well as this value multiplied by the number of records is indicated in the header of the EF⁷. So the structure of the linear fixed EF consists of a header which is then followed by a body containing from one to n number of records that each have an identical length.

The linear variable-length elementary file then is simply the same as the above only the records in the body of the file can have a variable number of bytes. This type of file has a much higher overhead in read/write access time and data storage overhead required of the file system⁸.

A cyclic elementary file can be thought of as a ring of records. These files are used for storing records in chronological order. When all records have been used for storage, then the next storage of data shall overwrite the oldest information. An EF with a cyclic structure consists of a fixed number of records with the same (fixed) length. In this file structure there is a link between the last record (n) and the first record⁹. Each successive write to the file performs the operation on the next physical record in the ring¹⁰. The structure then of a cyclic EF is very similar to the structure of a linear fixed-length EF having a header and a body consisting of fixed length records.

ACCESSING FILES

Lastly, we are going to consider how the files in the memory structure are accessed. There are a number of commands used to access the files, some of which are detailed briefly below.

Answer to Reset

This information is given to the mobile equipment at the start of each card session and spells out the operational parameters of the card. Once this is done the Master File becomes the current directory.

Select File

In order for a file to be available for updating a logical pointer to that file must be established. This is achieved through the use of the Select File command. When a file is selected by this command, any subsequent commands on this file such as reading or writing will operate on the logical pointer established by this command.

Binary Commands

The following commands all operate on transparent files-i.e. non record oriented files. If these commands are attempted on record-type files, the command will abort with an error indicated returned to the calling application.

- Read Binary-This command will retrieve some segment from an EF.
- Write Binary-This command will put information into some segment of an EF. Three things can be done with this command: 1) Set a series of bytes 2) clear a series of bytes or 3) do a one time write of a series of bytes
- Update Binary-This command will directly erase and store byte information into a segment of an EF
- Erase Binary-This command will set the value of a string of bits on an EF to zero

Record Commands

The following commands are designed to be executed against record oriented EFs, i.e. linear fixed length EFs. Like the binary commands above if one of these commands is executed against the wrong type of file-in this case a transparent one-the command will abort with an error indicated returned to the calling application.

- Read Record-This command will read and return the contents of one or more records in the selected EF.
- Write Record-This command will write a record into the selected EF. It can be used to achieve one of three results 1) a one time write of a record into the EF 2) setting specific bits within a specific record in the EF or 3) clearing specific bits within a specific record of an EF.
- Append Record-This command will add an additional record to the end of a linear record oriented EF.
- Update Record-This command will write a record into a EF. This will write a specific record into an EF and the net effect is that the specific record is erased and the new specified record is written into the EF.

While not an exhaustive treatment of the memory structure on a SIM, this paper has attempted to paint a general picture of how directories and files on a SIM card are laid out. In addition, some of the commands used to access and update the files have been mentioned. The examiner is encouraged to read the cited works and references that follow for additional detail and information.

Citations

1. Smith, Greg, *Mobile Telephone SIM Examination Course: Subscriber Identity Module: Memory Structure*, p.4
2. Smith, Greg, *Mobile Telephone SIM Examination Course: Subscriber Identity Module: Memory Structure*, p.4
3. Smith, Greg, *Mobile Telephone SIM Examination Course: Subscriber Identity Module: Memory Structure*, p.5
4. Smith, Greg, *Mobile Telephone SIM Examination Course: Subscriber Identity Module: Memory Structure*, p.5
5. Smith, Greg, *Mobile Telephone SIM Examination Course: Subscriber Identity Module: Memory Structure*, p.5
6. 3GPP TS 11.11, 6.4.1, p24

7. 3GPP TS 11.11, 6.4.2, p25
8. Jurgensen, Timothy M., Scott B. Guthery, *Smart Cards:The Developer's Toolkit*, 2002, p.118
9. 3GPP TS 11.11, 6.4.3, p25
10. Jurgensen, Timothy M., Scott B. Guthery, *Smart Cards:The Developer's Toolkit*, 2002, p.119

References

Jurgensen, Timothy M., Scott B. Guthery, *Smart Cards:The Developer's Toolkit*, 2002

3GPP TS 11.11, 1999

Smith, Greg *Mobile Telephone SIM Examination Course*, Trew and Co