**Source:** Nokia

**Title:** Usage of Index Files for extending EF Functionality

Due to limitations in addressing capabilities it is not possible to have more than 255 entries in an EF. In case a feature requires more entries the functionality need to be split into several EFs. In case the record length required is in excess of 255 bytes or new features are to be added to the basic functionality a method of extension fields has been introduced. The extension field is associated with certain EFs, e.g. ADN.

Definitions:

| | |
|---|---|
| Original EF | The original EF is the EF that has contained the initial functionality. The original EF is also called the first master EF. |
| Master EF | This is the first EF identifier in a record. The master EF is to its structure a copy of the original EF. |

In order to remove these restrictions the usage of Index files has been developed. The index files contains two or more records each record contains a set of EF Identifiers. EF identifiers contained in the same record indicate the location of Extension files. The extension file can contain extended data that should be appended to the record in the original file or it may contain other data, such as status information related to the record in the original EF. The contents of a record can be considered as a set of files placed in the horizontal plane. Each record indicates a set of files in the vertical plane, a new row.

When introducing the index file an administration file is required. This administration file may be a separate file or its functionality may be incorporated into a file containing a new feature to be added to the basic functionality, e.g. ADN and Group ID. The administration file shall have as many record entries as the original EF. The administration file contains records with offset pointers where the relevant data can be found in the added files indicated in the Index file. The record structure and content of the administration file is organized in such a way that the file ID entry in the index file has a corresponding entry in the administration file. This means that the offset in the administration file points to a record number in the file ID indicated in the index file.

The location of the Index file can be retrieved by selecting any of the master files and perform STATUS command. The file ID of the index file is indicated in bytes 16-17 in the status information.

## Examples of ADN Extensions

The example with the extension of the ADN feature from 255 entries to 510 entries with the addition of a Group ID file is illustrated as follows. The original ADN file ID is '6F3A'. As the EFEXT1(6F4A) also may contain data related to the ADN feature it should be indicated in index file. Retrieval of data from the Extension file is as specified in GSM 11.11. The basic principles is that any file that may contain data related to the feature, in this case ADN, is to be listed in the index file.

The nickname file can be used as a second name entry for the ADN feature. This means that the index file

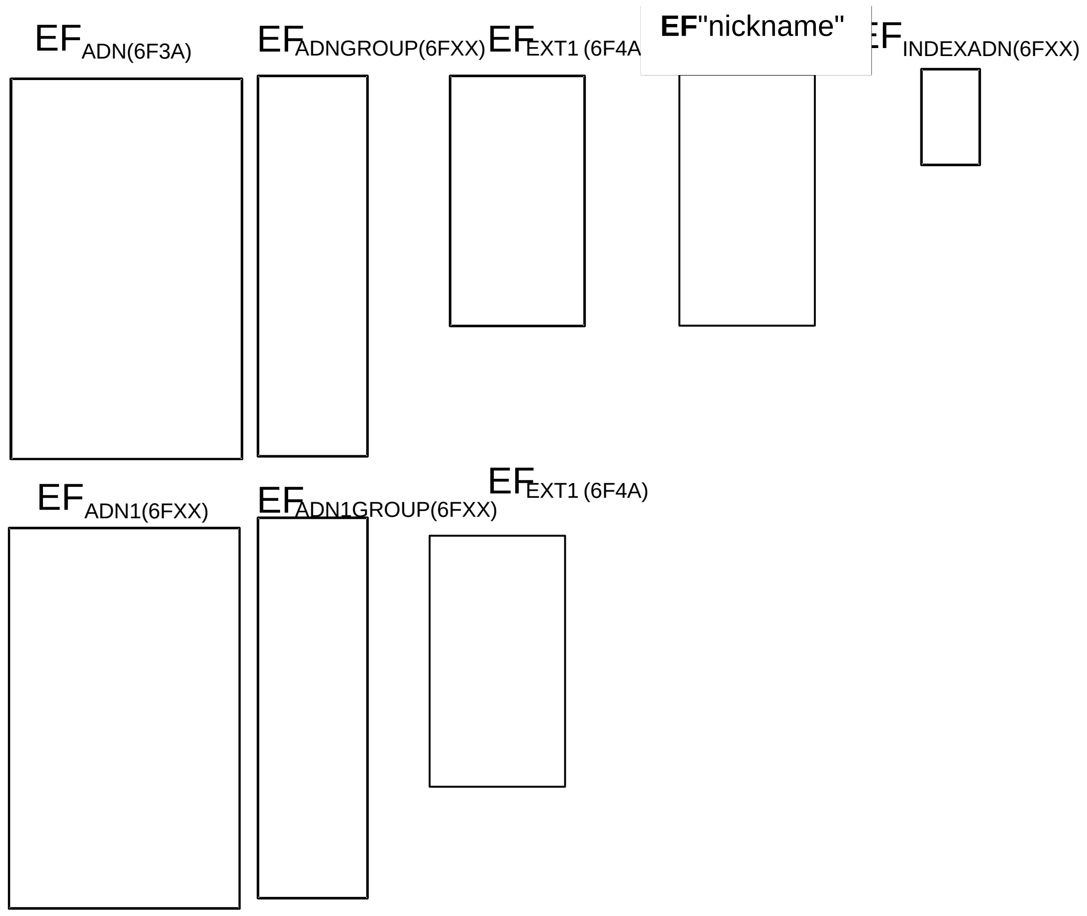

Figure 1 Example of adding grouping $EF_{ADNGROUP(6FXX)}$ and "Nicknames" to $EF_{ADN(6F3A)}$ and extend the ADN and grouping feature beyond 255 records

The structure of the Index EF is shown in the table 1.

Table 1 Structure Index File

| Identifier: 'XXXX' | Structure: linear fixed | | Optional |
|---|---|---|---|
| Record Length: X bytes | | Update activity: low | |
| Access Conditions:<br>READ Original EF<br>UPDATE ADM<br>INVALIDATE ADM<br>REHABILITATE ADM | | | |
| **Bytes** | **Description** | **M/O** | **Length** |
| 1 - 2 | EF Identifier of master EF | M | 2 bytes |
| 2 - 4 | EF Identifier of first Extension/Administration EF | M | 2 bytes |
| 4 - X | EF Identifier of additional Extension EF | O | 2 bytes |

The content of the index file $EF_{INDEXADN}$ record would in this case be as indicated in Table 2.

Table 2 Example of record contents in the Index File

| | | | | |
|---|---|---|---|---|
| Record 1 | $EF_{ADN(6F3A)}$ | $EF_{ADNGROUP(6FXX)}$ | $EF_{EXT1(6F4A)}$ | $EF_{"nicknames"}$ |
| Record 2 | $EF_{ADN1(6FXX)}$ | $EF_{ADN1GROUP(6FXX)}$ | $EF_{EXT1(6F4A)}$ | RFU |

Performing a STATUS command after having selected any of the files described in table 2 will return the identifier of the $EF_{INDEXADN}$ in bytes 16-17 of the status information.

The structure of the combined $EF_{ADNGROUP}$ and the Index administration file in this example is as described in the following table

Table 3 Structure of the combined Group/Administration File

| Identifier: '6FXX' | Structure: linear fixed | | Optional |
|---|---|---|---|
| Record Length: X bytes | | Update activity: low | |
| Access Conditions:<br>READ Original EF<br>UPDATE ADM<br>INVALIDATE ADM<br>REHABILITATE ADM | | | |
| Bytes | Description | M/O | Length |
| 1 - X | Grouping Information | M | 2 bytes |
| X – X+1 | Pointer to record number in $EF_{EXT1(6F4A)}$ | M | 1 byte |
| X+2 – X+3 | Pointer to record number in $EF_{\text{"nicknames"}}$ | O | 1 byte |

ADN example in case of two names for one single telephone number. This is a special case of the pervious example as in this case $EF_{ADN(6F3A)}$ has the same number of records as the $EF_{\text{"second name"}}$. This is the basic difference between this example and the example with $EF_{\text{"nicknames"}}$. In this example a second name is also possible on the $EF_{ADN1(6FXX)}$ where as "nicknames" are only possible for $EF_{ADN(6F3A)}$ in the example described in Figure 1.

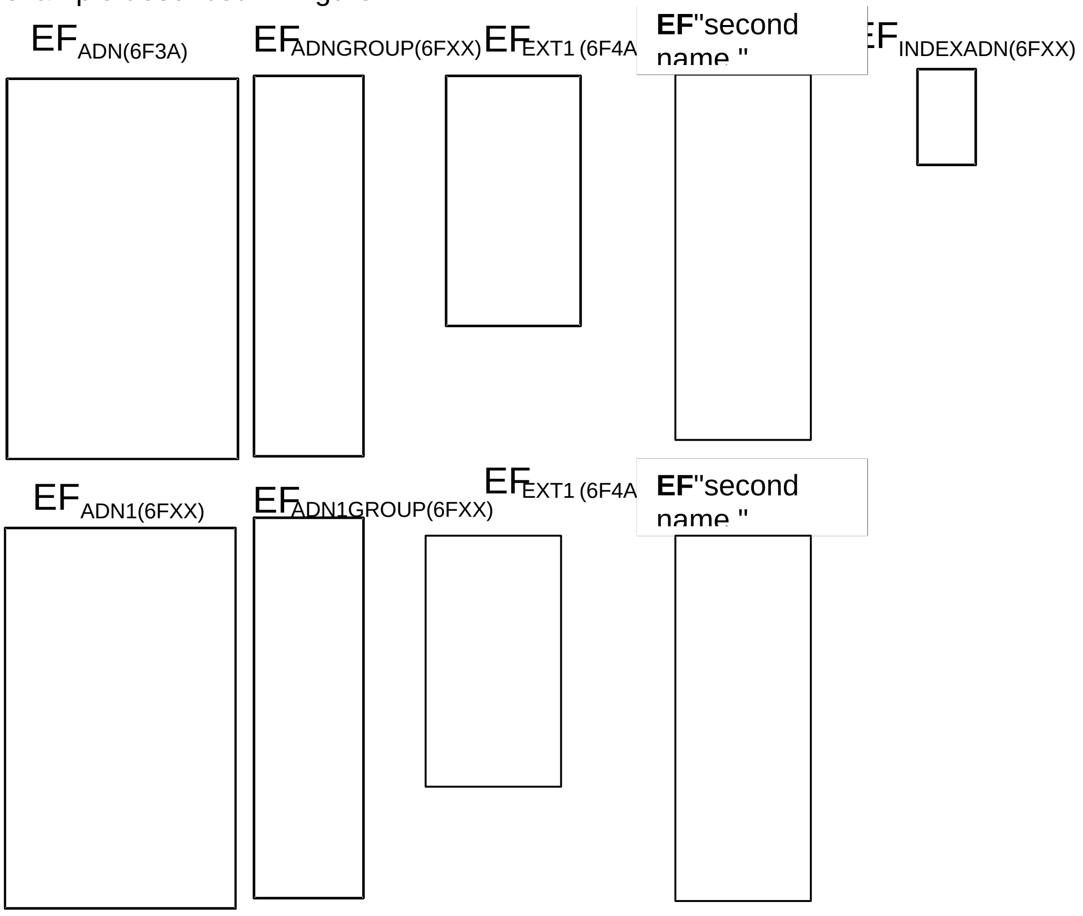

Figure 2 Example of having two names attached to one ADN Number

The implementation of this feature can basically be done in two ways. Alternative 1 is to have record 1 in the master ADN file and record 1 in the second name file to correlate. This means that if a search is performed on one of the files the record number found can be used to retrieve the number associated with the name regardless if it is search for in the master ADN file or in the "second name" file. The other possibility is to use the offset pointer provided by the administration file. In this case the administration file needs to be used to search for the record number where the "second name" entry offset matches the record number found in the search of the "second name" EF.