EpiData User Defined Functions

For EpiData version 3.1 (or later)

Version of : November 25th 2004

About EpiData User Defined Functions

The addition of user defined functions to EpiData has been programmed by Salaheddin Mahmud.

Citation: EpiData User Defined Functions. EpiData Association & Salaheddin Mahmud, 2004.
Http://www.epidata.dk/documentation.php

The principle is that the user in a chk file loads functions stored in an external libarary file (dll file), e.g.:

```
BEFORE FILE
  load soundex
END
```

and then in a particular field calls functions which are in that file:

```
NAME
  AFTER ENTRY
    NEWSOUND=newsoundex(Name)
    episound= soundex(name)
    meta = metaphone(name)
  END
END
```

The development of a ”user defined function” can be done for any function which you can programme. To make it work adapt one of the provided examples.

Examples, source code and documentation will be available from http://www.epidata.dk/documentation.php

On next pages find a copy of the documentation for provided functions.

A Check Digit Algorithm Implemented as a User Defined Function in EpiData

Version 1.0
Lars Hvidberg ,Jakob Mortensen, Axel Skytthe , Institute of Public Health, Epidemiology
University of Southern Denmark

The epiGumm.dll

The epiGumm.dll contains functionality to generate check digits and validate identifiers using the algorithm described in:

H. Peter Gumm, A New Class of Check-Digit Methods for Arbitrary Number Systems, IEEE Transactions on Information Theory, Vol IT-31, No. 1, January 1985

The major benefit of this algorithm is that is it possible to use consecutive numbering and still catch all “one digit wrong” and all “two neighbouring digits interchanged” errors.

The implementation in Delphi used here is based on the development carried out in the GENOMEUTWIN project which is supported by the European Union Contract No. QLG2-CT-2002-01254.

Three methods are supplied by the dll:

1. get_gumm takes a number as argument and returns the check digit.
2. get_id takes a number as argument and returns the argument with the check digit added.
3. verify_gumm takes a number as argument and returns a boolean indicating if the number is valid or not.

The dll should be in the zip file downloaded containing this document. If you extracted the zip file to a folder you should now have a subfolder called “dll” where the dll is located.

Using the epiGumm.dll in EpiData

- Place the epiGumm.dll file in the installation directory of EpiData.
- In the relevant check file add a load command in the before file section

```
BEFORE FILE
  LOAD epiGumm
  …
END
```

- To show how to use the methods of the dll we given an example using the verify_gumm method. The other methods are called in exactly the same way. **Important**: Any field to be validated using this dll must be a numeric field accepting only integers. Any field to be validated (verify_gumm) using this dll should have range starting at 10 or above since it doesn't make sense to validate a one digit number using a check digit. Also it doesn't make sense to ask for adding or calculating a check digit to the number 0 (get_gumm and get_id). Here the range must start at 1 or above.

```
BEFORE FILE
  LOAD epiGumm
  DEFINE digitOK <Y>
END

….

* The field testdigit must be a numeric type accepting only integers and have a
*  range starting at 10 or above
testdigit
  RANGE 10 INFINITY
  AFTER ENTRY
    digitOK = verify_gumm(testdigit)
    IF digitOK <> "Y" THEN
      * if not valid, notify the user and stay in the current field
      HELP "Not a vadlid identifier" TYPE=ERROR
      GOTO testdigit
    ENDIF
  END
END
```

An example including qes, rec and chk files should be in the zip file downloaded containing this document. If you extrated the zip file to a folder you should now have a subfolder called "gummExample" where these files are located.

Source code for the epiGumm.dll

The Delphi source code for the epiGumm.dll should be in the zip file downloaded containing this document. If you extracted the zip file to a folder you should now have a subfolder called "source" where the source code is located. The implementation was made by Lars Hvidberg.