When an application programmer calls a function present in the gef2.dll, the requested action usually may be performed, however in some cases the actions fails. Generally functions return the status of an action: true if the action is successful and false if the action has failed. It is the duty of the programmer to check the status to verify whether the requested was successful. If some part of the requested action has failed, the origin of the failure is written in a list of errors. There are functions made available in the gef2.dll that will allow the application programmer to query the list of what has gone wrong and to retrieve the severity of the error. Usually the origin of errors in functions is non-compliance to the rules of GEF.

The errors can be presented in a textual form, e.g. by means of the function *write_error_log();*. Additionally the severity can be queried. Errors are divided into two main classes:

1. errors against the syntax of GEF
2. errors related to a specific standard.

Examples of the first class are: a keyword is incorrectly spelled or a compulsory field of a keyword is missing. An example of the second class is: a keyword, vital for the evaluation of a specific type of measurement, is missing.

The two main classes are divided into five types, in a similar way. Errors related to the syntax of GEF:

```
#define       GEF_SEVERE    1
#define       GEF_ERROR     2
#define       GEF_MINOR     3
#define       GEF_WARNING   4
#define       GEF_INFORM    5
```

and errors related to a standard (STD):

```
#define       STD_SEVERE    11
#define       STD_ERROR     12
#define       STD_MINOR     13
#define       STD_WARNING   14
#define       STD_INFORM    15
```

The application programmer can retrieve the severity of errors by means of the function *get_error_level_all();*. The returned value of this function is a value corresponding to the errors listed above: the numbers 1-5 and 11-15.

The interpretation of these levels is:

1. The severity of a SEVERE error is such we recommend not to continue actions on these data; usually further actions are impossible. Examples are: the header of a file is not readable, the header lacks vital information or the data in the data block can not be read. The suggested action: stop.
2. An ERROR is less severe than a SEVERE error, nevertheless we can not guarantee that any further processing of the data can be done without any difficulties. Examples are: a programmer tries to retrieve the fields of a columninfo keyword for a non existing column or one of the compulsory fields of an optional keyword is missing. The suggested action: continuation is possible, but the programmer has to perform special additional checks for a reliable result.
3. The severity of a MINOR error usually allows the continuation of the program, however there is a slight possibility that something will go wrong. The error informs the programmer that some rules are not obeyed. An example of a MINOR error is an illegal format in the keyword REPORTDATAFORMAT. The header and data block can still be written to file, however the layout of the data block is not as intended. Action: continue with the processing of the data, inform the user that a minor error has occurred.
4. A WARNING tells the programmer that something is wrong, but the object of attention has been corrected by the gef2.dll. An example of a warning is the occurrence of some additional data after the last complete scan. These numbers are not added to the data block. The recommended action is: continue processing.
5. The last type is INFORM. It informs the programmer that an action has been done for him/her. Nothing has gone wrong. E.g. during processing data a temporary file is created on hard disk. Such files will be deleted after completion of the task, however sometimes these files can not be deleted. An INFORM informs the programmer about this temporary file. The recommended action is: continue processing the data.