IGC Vali changes

This document proposes changes to the existing IGC Specification regarding the requirements of validation programs. The changes are required so that unattended programs (online web servers, for example) can satisfactorily perform validation checks.

The problem

The current situation is that validation programs can only return one of two states (`PASSED` or `FAILED`), with any errors being reported by visual text output. This is acceptable when there is human input to read the text, but otherwise there is no reliable way to establish if the file has failed because it has been modified or because of some other condition.

The solution

This is designed to be backward-compatible and easily incorporated into existing software. It requires that one of three states are returned (`PASSED`, `FAILED`, `ERROR`) represented by an integer value that acts as a **status code.** These are outlined below.

State	Status Code	Status Name	Description
PASSED	1	STATUS_IGC_PASSED	The IGC file passed the validation check
FAILED	2	STATUS_IGC_FAILED	The IGC file failed the validation check
ERROR	100	STATUS_ERR_FILE_OPEN	The IGC file cannot be opened
ERROR	101	STATUS_ERR_FILE_OTHER	The IGC file is not supported by vali program
ERROR	102	STATUS_ERR_VALI_ABORT	The vali program was aborted
ERROR	200	STATUS_ERR_VALI_ERROR	Unexpected error in vali program
ERROR	201-299	STATUS_ERR_VALI_XXXXX	Specific errors private to the vali program

The `ERROR` status codes are split into two groups, *user errors* (100-102) and *vali program errors* (200-299), to help identify the possible cause. More information about status codes is given below. Note that the status names are informational only.

How the status code is returned depends on whether the vali program is an executable or a DLL. For executables a result string is added to the output, while for DLLs a new function is used to return the status code.

Vali executables

The result string is included as the last line of the output, formatted as follows: the word `IGCVALI` followed by a colon, then the state (`PASSED|FAILED|ERROR`), followed by a comma, then the status code. Examples:

```
IGCVALI:PASSED,1
IGCVALI:FAILED,2
IGCVALI:ERROR,x
```
where x represents the status code

Note that there are no spaces between the values and that all characters are upper-case. Programs that output usage information (help) when called without any parameters must always include a result string in the last line of the output, reporting an error with a status code of `STATUS_ERR_FILE_OPEN`. For example:

```
Usage vali-xxx <filename>
...
IGCVALI:ERROR,100
```

The following are general rules for vali executables, which must be strictly adhered to:

- Programs must be 32-bit console applications.
- Programs must use structured exception handling to catch hardware and software exceptions.
- Programs must allow their output to be redirected: all output must be sent to `STD_OUT` or `STD_ERR` (unless otherwise directed by a private option parameter).
- Programs must not show any windows (dialogs, message boxes, progress bars etc). This is particularly important for programs running on an unattended system.
- Programs must always include a result string in the last line of the output, unless a private option parameter is being used.

Vali dlls

A new function is required to return the status code, **`ValidateLogEx`**, which is similar to the existing `ValidateLog` function except that it returns an unsigned integer, the status code.

```
DWORD ValidateLogEx(LPCTSTR fileName)
```

If `quietMode` has been set to `TRUE` in the call to `InitializeDLL`, then the program must ensure that no windows are displayed (dialogs, message boxes, progress bars etc). This is particularly important when being run on an unattended system. Also, dlls must use structured exception handling to catch hardware and software exceptions.

Vali program flow

In order to return consistent results, the vali program must perform the following steps:

1. Check that a file name has been passed in. If not, return `STATUS_ERR_FILE_OPEN`.
2. Open the IGC file for reading. If this fails, return `STATUS_ERR_FILE_OPEN`.
3. Read the first four bytes and check they match `AXXX`, where `XXX` is a three-character identifier supported by the program. If they do not match, return `STATUS_ERR_FILE_OTHER`.
4. Read and save the G Record from the end of the file. If it does not exist, or is an empty value, return `STATUS_IGC_FAIL`.
5. Read the relevant IGC data to create a new G Record, then check that it matches the G-Record from the file. Return `STATUS_IGC_PASSED` if it matches, otherwise return `STATUS_IGC_FAIL`.

If any other errors or exceptions are encountered during this process, the vali program must return `STATUS_ERR_VALI_ERROR`, or a vali-defined status code in the range 201-299. If the process is aborted by the user before it has completed, the vali program must return `STATUS_ERR_VALI_ABORT`.

Status Codes

`STATUS_IGC_PASSED`

This must only be returned if the IGC file has successfully passed the validation check, with the following conditions being satisfied:

- the IGC file is applicable (`AXXX` matches expected `XXX`).
- the file contains a G Record.
- the G Record in the file matches the G Record created from the file data.

`STATUS_IGC_FAILED`

This must only be returned if the IGC file has failed the validation check, with the following conditions being satisfied:

- the IGC file is applicable (`AXXX` matches expected `XXX`).
- there is either no G Record in the file, it has an empty value or it does not match the G Record created from the file data.

`STATUS_ERR_FILE_OPEN`

This must only be returned if the vali program cannot open the IGC file. This will happen if no file name has been passed to the program, if the file name points to an incorrect location, or because of any other program or system error.

`STATUS_ERR_FILE_OTHER`

This must only be returned if the vali program does not support the IGC file, with the

following condition being satisfied:

- the IGC file is not applicable (AXXX does not match expected XXX).

Specifically this means that the first four bytes of the file data do not contain the characters AXXX, where XXX matches a three-character identifier supported by the vali program. This covers situations where the file is empty, is not in IGC format or is not applicable to the vali program.

STATUS_ERR_VALI_ABORT
This must only be returned if the vali program has been aborted by the user before it has finished its normal processing. For vali executables this means that a CTRL event has been sent to the program, while for vali dlls that a progress indicator has been cancelled or that the dll has been unloaded unexpectedly.

STATUS_ERR_VALI_ERROR
This must only be returned when there is an error condition that does not satisfy any of the status codes defined above, for example an access violation caught by structured exception handling. For more robust error reporting, specific errors can be given their own status codes in the range 201-209 (STATUS_ERR_VALI_XXXXX).

STATUS_ERR_VALI_XXXXX
These are private status codes that are defined by the vali program. They must be in the range 201-299 and can be used to report specific errors that would otherwise be covered by the more general STATUS_ERR_VALI_ERROR category.