UniChrom file converter specification

Background and termini

1. File converters are intended for translation of external file formats to UniChrom internal storage. E.g. storage items like peaks, data arrays, properties etc. if possible are stored and restored by designed plug-in module.
2. Converters are designed as DLL modules usually without graphic interface only with ability of error report. Converters can be two way in/to selected file format and single to or from selected file format. Single converters can support several file formats simultaneously.
3. Converter is executed in UniChrom process address space and invoked in context of arbitrary thread by calling entry point **cnvImport()** or **cnvExport()** for expected purposes. Note. Conversion only takes place from/to internal UniChrom data storage to/from selected data format.
4. Converter can be a LIMS client but not file conversion utility. Client converters can have user interface.

SPEC-API – functional API for getting/setting data in UniChrom internal storage.
SPEC-API version – numeric value, specifies the current API version. Converter should check the API version is not less than library version being linked with.

File converter entry points

cnvInit()

Converter initialization routine

```
function cnvInit(const intf:PUniChromEntryTable):integer;stdcall;
```

The function is intended to pass to the converter table of UniChrom entry points, which can be used for data retriveval/storage while converting particular data format.
UniChromEntryTable is a structure containing the pointers to SPECAPI interface functions.
Table contains field named **dwVersion**, which should be checked to be not less than value specified in the header file (unit file) as **SPEC_API_VERSION**.

Function should return CNV_OK=1 on success.

Failure states reported as values less than or equal to 0. Value less than 0 specifies OS error code.

cnvEnumFormats()

Converter information routine. Used to be defines the number of formats, supported by converter, direction of conversions and conversion flags.

```
function cnvEnumFormats(nFormatId:integer;FormatName:PCHAR;var bufsize:integer;var flags:integer):integer;stdcall;
```

When UniChrom passes **nFormatId**=0 the response should be integer number defines count of formats supported. Returning zero means no further enumeration.

Returning positive N cause N subsequent calls of the entry point with nFormatId from 1 to N.

On each call the FormatName points to buffer to receive null-terminated string which describe the format name to user. String can be UTF8-encoded to pass symbols representing non-ANSI characters. The **FormatName** should be of the follwing structure:

Textual Format description With Example of (*.extension)|*.extension;*.extension1

Single format may be represented by files with different extensions. Extension should be specifie after pipe sign «|» using semicolon «;» as delimiter. Wildcards «*» and «?» are allowed.

On return the **bufsize** should contain the number of bytes copied.

The **flags** specifies in bitmask the way converter handles the format.

Possible flags in bitmask are:

FLAG_READ	The specified format can be READ by converter. The format description (FormatName) would appear in file «open» dialog.
FLAG_WRITE	The specified format can be WRITTEN by converter. The format description (FormatName) would appear in file «save as» dialog.
FLAG_IMPORT	The specified format can be READ by converter, but it is not necessary file, so instead of file dialog the menu «File\Import» is used. The converter can have its own GUI to select objects for import. The extension has only identification purpose (to distinguish file filters). The description of converter represents its purpose e.g. «MNPZ Lims»
FLAG_EXPORT	The specified format can be WRITTEN by converter, but it is not necessary file, so instead of file dialog the menu «File\Export» is used. The converter can have its own GUI to select objects to export to. The extension has only identification purpose (to distinguish file filters).

Any combination of flags is allowed. Specifying both FLAG_READ+FLAG_IMPORT would cause the textual format description appear in File\Import menu and in File Open dialog. Empty flags cause the converter to be ignored.
Function should return CNV_OK=1 on success or CNV_FAIL=0 on problem.

cnvImport()

Converter entry point for importing / reading file data.

```
function cnvImport(nFormatId:integer;const FileName:PChar;hSp:HSPEC):integer;stdcall;
```

nFormatId number corresponds to the format textual description which was determined during **cnvEnumFormats()** call.

hSpec - handle of spectra-document. This handle is used in subsequent SPEC-API calls. The handle should not be hInvalid (NULL) value.

Typically import process consists of reading file, interpreting the data and passing the data to UniChrom via calls of SPEC-API entry points from UniChromEntryTable. Typical usage of UniChromEntryTable functions is presented in sample converter files. SPEC-API reference guide is in document called “SPEC-API functions manual”.

cnvExport()

Converter enty point for exporting / saving file data.

```
function cnvExport(nFormatId:integer;const FileName:PChar;hSp:HSPEC):integer;stdcall;
```

nFormatId number corresponds to the format textual description which was determined during **cnvEnumFormats()** call.

hSpec - handle of spectra-document. This handle is used in subsequent SPEC-API calls. The handle should not be **hInvalid** (NULL) value.

Typically export process consists of obtaining data from UniChrom and writing the file of corresponding structure.