sceye-Sphere Storage-API

Basics:

Component	C / C++
Component name	StorageAPI
Visual Studio:	Version 2010 or higher (only Version 2010 tested)

Description:

The Storage-API is the core component of SceyeSphere to store and manage the document structure, document metadata and image data.
The default Storage-API is implemented as a DLL with methods exported in C-style.

Features:

- stores Archive, Document and Page information (both metadata and images)
- provides Clipboard and Trashbox functionality
- provides GlobalSettings and ArchiveSettings storage
- provides update mechanism for document structure (for example for split and merge functionality, deletion of documents and pages, etc.)

Features of the default Storage-API:

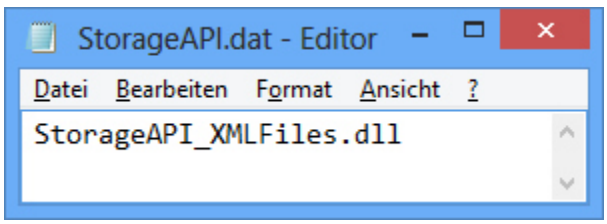
- uses xml files to store Archive, Document and Page information
- xml files and image files are stored in local filesystem folder %appdata%\SceyeSphere
- Global settings and Archive Settings are stored as xml files

Implementation:

You will find the detailled interface description in the file
sceye-Sphere Storage-API Interface.pdf

You will find the framework project as a Visual Studio 2010 solution in the zip file
sceye-Sphere Storage-API project.zip

To implement your own Storage-API please use the Storage-API framework project provided by Silvercreations. The project contains all method stubs exported by the DLL. Be sure to use a different filename for the Storage-API DLL. To use the new Storage-API DLL you have to copy your new DLL in the SceyeSphere folder and create a text file in the same folder called ***StorageAPI.dat*** containing your Storage-API DLL filename. From now on SceyeSphere will use your DLL instead of the default StorageAPI_XMLFiles.dll.



Annotations:

Basic structure of the archive

SceyeSphere consists of one or more archives, a global clipboard workspace, a global trashbox workspace and global settings.

A SceyeSphere archive consists of

- one workspace (element which holds the documents and additional workspace metadata)
- one or more documents (element which contains pages and additional document metadata)
- one or more pages (element which represents the image data and additional metadata)
- workspace settings

Terms and definitions

- All indices are 0-based
- The default storage api stores all files in the folder %appdata%\SceyeSphere
- The default storage api stores all metadata information in xml format
- Used image file formats in the archive are jpeg (and alternatively tif and png but currently not used)
- Each page is stored in four different sizes
    - small thumbnail (filename end with _8)
    - two medium sized images (filename end with _4 and _2)
    - the full-sized image (filename has no additional suffix)

- Identifiers
    - a document is identified by a GUID string (GUID = Globally Unique Identifier)
    - a page is identified by a GUID string
    - the identifier looks like `d9227263-8545-443f-b921-cb9929644891`

- Timestamps
  Timestamps are stored like yyyymmdd_hhmmss
  for example: `20131218_154543`

- Colormodes
  There are three colormodes supported. Internally all images are processed as color images. The system stores all black&white images as greyscale images for better display quality. The black&white images are converted during export.
  Colormode values:
    - 1 = color (8 bit per channel, 3 channels)
    - 2 = greyscale (8 bit per channel, 1 channel, indexed color)
    - 3 = black&white (1 bit per channel, 1 channel)

- Strings
  All strings passed to the storage api and returned from the storage api are wchar_t strings. If a method returns a string the caller tells you via the MaxBufferSize variable how much space is reserved for string.

Attention: You have to clear the string buffer yourself.You should always do something like this:

```
(see ApiGetLicense for details)
memset(strLicenseName,0,iMaxBufferSize_LicName * sizeof(wchar_t));
memset(strLicenseNumber,0,iMaxBufferSize_LicNumber * sizeof(wchar_t));
```

- The pApiData pointer
  The parameter list of each method call starts with a pApiData pointer. This pointer holds a reference to an object (for example a struct) which lives as long as the program runs. This object is created during ApiInit and released during ApiRelease. The reason for this object is that you should not use global variables in the DLL and you have no possibility to pass data between two dll method calls. You can define and use this struct as you like. The program just stores the pointer and passes it around and is not accessing any data of this object.

- Clipboard and Trashbox
  The clipboard workspace and the trashbox workspace are very similar to the normal workspace but with special meaning and features.
  If a page is copied to the clipboard or trashbox a new document is created in the clipboard/trashbox to hold the page.

- The ArchiveUpdate procedure
  The ArchiveUpdate commands are used to insert, update or delete a document
  Update types are:
  - 0 = insert document
    with -1 as update index the document will be appended to the document list, else it will be inserted at the specified position

  - 1 = update document
    re-creates the document structure with the page info provided by ArchiveUpdateAddPage command and document metadata provided by ArchiveUpdateSetDocInfo

  - 2 = delete document
    if a document identifier is specified via ArchiveUpdateSetDocInfo then this information will be used for deletion, else the index will be used

  - 3 = insert document with existing doc file
    insert a document in the workspace structure (see update type 0) which already exists

  - 4 = create document from existing page(s)
    page information provided via ArchiveUpdateAddPage

  - 5 = delete document from base file but leave document file intact
    see update type 2

- 6 = Insert Document without adding to base file creates an “invisible” document