DynaPDF 3.0 Beta 1
Implementation Notes

```
------------------------- Changed data types -------------------------
```

The following data types contain extensions due to new features or values which were not supported in DynaPDF 2.x. Only new constants or members are listed.

```
typedef enum
{
   ...
   atRichMediaExec // PDF 1.7 Extension Level 3
}TActionType;

typedef enum
{
   aiUserDefined   // Internal, used to store undefined values in
                   // imported annotations
}TAnnotIcon;

typedef enum
{
   ...
   atMovieAnnot,  // PDF 1.2
   atPrinterMark, // PDF 1.4
   atProjection,  // PDF 1.7 Extension Level 3
   atRichMedia,   // PDF 1.7 Extension Level 3
   atScreen,      // PDF 1.5
   atTrapNet      // PDF 1.3
}TAnnotType;

typedef enum
{
   ...
   bsUserDefined // Internal, used to store undefined values in
                 // imported annotations
}TBorderStyle;

typedef enum
{
   ...
   civCustom  // PDF 1.7 Extension Level 3, the collection view is
              // presented by a SWF file.
}TColView;
```

```
typedef enum
{
   ...
   diPDFX_Conf    = 9,  // GetInDocInfo() or GetInDocInfoEx() only. The
                        // value of the GTS_PDFXConformance key.
   diCreationDate = 10, // GetInDocInfo() or GetInDocInfoEx() or after
                        // ImportPDFFile() was called.
   diModDate      = 11  // GetInDocInfo() or GetInDocInfoEx() only
}TDocumentInfo;

typedef enum
{
   ...
   faiUserDefined // Internal, used to store undefined values in
                  // imported annotations
}TFileAttachIcon;

typedef enum
{
   ...
   naUserDefined  // Non predefined action
}TNamedAction;

typedef enum
{
   ...
   ovNotSet // Internal
}TOCVisibility;

typedef enum
{
   ...
   taNotSet = 4 // Internal -> Do not use this value!
}TTextAlign;

typedef enum
{
   ...
   toAnnots = 4, // Annotation order -> PDF 1.7 Extension Level 3
   toFields = 5  // Widget order     -> PDF 1.7 Extension Level 3
}TTabOrder;

struct TPDFAnnotationEx
{
   ...
   UI32 PageIndex; // The page index is used to sort form fields. See
                   // SortFieldsByIndex().
   /* 26 Reserved fields follow */
};
```

```
struct TPDFExtGState
{
   ...
   LBOOL SoftMaskNone; // Internal -> Disables the active soft mask
   /* 8 reserved fields follow */
};

struct TPDFFieldEx
{
   ...
   UI32            PageIndex;     // Array index to change the tab
                                  // order, see SortFieldsByIndex().
   const void*     IBarcode;      // If present, this field is a
                                  // barcode field. The field type is
                                  // set to ftText since barcode fields
                                  // are extended text fields. See
                                  // GetBarcodeDict().
   const void*     ISignature;    // Signature fields only. Present
                                  // only for imported signature fields
                                  // which have a value. That means the
                                  // file was digitally signed. See
                                  // GetSigDict().
                                  // Signed signature fields are always
                                  // marked as deleted!
};

// This structure has been changed since the old functions
// GetInNamedDest() and GetInNamedDestCount()are no longer supported.
// GetInNamedDestCount() returns always zero. See description at the
// function section.
struct TPDFNamedDest
{
   UI32            StructSize; // Must be initialized to
                               // sizeof(TPDFNamedDest)
   const char*     NameA;
   const UI16*     NameW;
   UI32            NameLen;
   const char*     DestFileA;
   const UI16*     DestFileW;
   UI32            DestFileLen;
   SI32            DestPage;
   struct TPDFRect DestPos;
   TDestType       DestType;
};

struct TPDFParseInterface
{
   ...
   TBeginLayer* BeginLayer; // Starts the definition of a layer
   TEndLayer*   EndLayer;   // Finishes the layer. Consider the new
                            // visibility state!
   /* 25 reserved fields follow */
};
```

`-------------------------- Changed functions ------------------------`

`GetInNamedDestCount(),GetInNamedDest():`

GetInNamedDestCount() returns always zero, also if the file contains named destinations. These functions were provided for a few customers who depend on named destinations. Because DynaPDF 2.x resolved all named destinations during import there was no other way to access them.

This behavior has been changed in DynaPDF 3.0. Named destinations are now imported like any other object. The new functions to access named destinations are GetNamedDest() and GetNamedDestCount(). The destinations are available after the file was imported. The old functions are no longer included in any DynaPDF interface (with exception of the abstract C++ interface), although the functions do still exist in the DLL.

Sorry, but the required overhead to get these functions working in DynaPDF 3.0 is simply too large...

`SetUseSwapFile(), SetUseSwapFileEx():`

The contents of pages, templates, and patterns are no longer written to the temp file also if SwapContents is set to true.

`------------------------- Not yet implemented ------------------------`

`FlushPages(),`
`LoadFDFData(),`
`OpenTag(),`
`CloseTag(),`
`CreateStructureTree():`

These functions do nothing in the current beta release. The implementation of these functions will be added as soon as possible.

`--------------------------- Other changes ----------------------------`

`EditPage():`

When opening an existing page for editing DynaPDF does now the following:

- When the page is opened the first time, DynaPDF parses the content stream to determine the graphics state at the end of the stream. Errors in the content stream will be repaired if possible.
    - If the content stream contains non-repairable errors then editing is still possible but the result is of course undefined. Errors are written to the error log.
    - When closing the page, the graphics state is stored in a compact format so that it is not required to parse the page again when it will be edited again. Prior versions simply enclosed the content stream into save/restore graphics state operators if the content stream did not start with a save graphics state operator. Although this produced correct results in most cases, there could be problems with malformed content streams which end with unbalanced restore graphic state operators.

OpenImportFile(), OpenImportBuffer()

When opening a PDF file the functions read the cross-reference tables and try to load the required top level objects such as the Trailers, Catalog, Page Tree, and Encryption dictionaries.

If the file contains damages that prevents loading of these objects then the functions fall back into repair mode and try to repair the file. The functions rebuild then the cross-reference table by scanning all the objects in the file. Errors and warnings are written to the new error log. If the file could successfully repaired then the functions return success and no error will be raised via the normal exception handling.

It is not always possible to identify damages when opening a PDF file, e.g. if it contains only partial damages which affect specific pages only. If it is not possible to load certain pages due to errors then it is possible to load the file explicitly in repair mode. This makes of course only sense if the file was not already loaded in repair mode. Whether this is the case can be checked with the function GetInRepairMode(). Take a look at the description of OpenImportFile() in the help file for further information.

OpenImportBuffer() supports now an additional flag to avoid copying of the input buffer.

See OpenImportBuffer() for further information.

Internal color format:

The internal color format in the graphics state was changed from 8 bits to 16 bits per channel.

Transparent images:

Images with an alpha channel are now always inserted transparent. The alpha channel is converted to a soft mask to preserve the transparency.

```
-------------------------- New Functions ----------------------------

pdfCreateBarcodeField()
pdfFlushPages()
pdfGetBarcodeDict()
pdfGetErrLogMessage()
pdfGetErrLogMessageCount()
pdfGetInFieldCount()
pdfGetInRepairMode()
pdfGetNamedDest()
pdfGetNamedDestCount()
pdfGetSigDict()
pdfHaveOpenPage()
pdfInsertImageExA() // These functions are not new. Only the Unicode
pdfInsertImageExW() // version was added.
pdfSetFontSearchOrderEx()
pdfSetMaxErrLogMsgCount()
```