PDF-XChange Viewer SDK
Simple DLL

Version 2.0.4x

Head Quarters:

Tracker Software Products (Canada) Ltd.,

#3 - 466 Trans Canada Highway.

Duncan. V9L 3R6

British Columbia. Canada.

Sales
Tel: Canada (+00) 1-250-597-1621
Fax: Canada (+00) 1-250-597-1623

In Europe:

Unit 17, Raleigh Court. Priestly Way.
Crawley. Sussex. RH10 9PD.
England.

Sales
Tel: +44 (0) 20 8555 1122
Fax: +44 (0) 844 800 4521

http://www.docu-track.com

Sales@docu-track.com

Table of Contents

1. Overview

PDF-XChange & Viewer SDK – Version 1-2.x

1.1. Introduction

The PDF-XChange Viewer SDK is available as a stand alone Developer kit and also included with our PDF-XChange and PDF-Tools SDK's. The PDF-XChange Viewer offers both a Direct DLL method to create simple PDF viewing within your application or a full ActiveX which allows the use of the full PDF-XChange Viewer within your product. The PDF-XChange Viewer is not a Royalty Free tool kit and a limited number of Licenses are included for end user distribution with your application with the SDK product you purchase, with additional bulk license packs available for a modest cost - see our web site/price lists for more detailed information.

This toolkit as with all our developer kits may not be used to develop Toolkits or Components of any type for use by other non-licensed developers or for use to assist in the creation of Printer driver under the usual license conditions provided. For more information on licensing please read the license agreement and if any doubt as to whether your intended use would be in breach of the license terms please contact us to discuss your needs in more depth as we do offer alternate licensing and will tailor our agreement to meet your needs in most circumstances under different terms of supply.

1.2. Support

Support is available direct from our user forums http://www.docu-track.com/forum/index.php.

We recommend that developers use the evaluation download as extensively as possible prior to purchase. The evaluation versions are fully functional with no time out or other crippling mechanism – save that a watermark is stamped on any PDF page generated by the Driver/Library tools – only on purchase will you be provided with the serial number and unlock string required by each component to be passed within your application code (see the demo applications provided) to enable PDF generation without this demo watermark stamp.

By virtually creating your application to the point where you are ready for distribution before you purchase, you are guaranteed satisfaction and we do not disappoint developers asking for refunds once purchased – we do not offer any money back options – so please ensure you are 100% satisfied before you purchase.

Developer's may also find it useful when developing applications for the purpose of creating and manipulating Adobe PDF Formats - to download the documentation relevant to this format from the Adobe web site - at the time of writing this is currently free and may prove useful in explaining in more detail the functionality available. http://www.adobe.com/.

Tracker Software Products Ltd also provide End User and Developer Tool Kits for the creation and manipulation of PDF and Raster Image files and Virtual Printer Drivers. For more information please visit http://www.docu-track.com.

1.3. License Agreement

License Agreement PDF-XChange Viewer (SDK) from Tracker Software Products Ltd 2006-2009

PRINTED BELOW IN ITS ENTIRETY IS THE LICENSE AGREEMENT GOVERNING YOUR USE OF THE SOFTWARE. PLEASE READ THE LICENSE AGREEMENT.

IMPORTANT

TRACKER SOFTWARE PRODUCTS LTD. IS WILLING TO LICENSE THE ENCLOSED SOFTWARE TO YOU ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THE LICENSE AGREEMENT PRINTED BELOW. PLEASE READ THE TERMS CAREFULLY BEFORE OPENING THE PACKAGE CONTAINING THE DISKETTE(S)/CD-R(S), Electronic File OR CLICKING THE ACCEPT BUTTON DURING INSTALLATION, AS SUCH CONDUCT INDICATES YOUR ACCEPTANCE TO ALL OF THE TERMS OF THIS LICENSE AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS, TRACKER SOFTWARE PRODUCTS LTD IS UNWILLING TO LICENSE THE SOFTWARE TO YOU, IN WHICH CASE YOU MUST IMMEDIATELY RETURN THE PACKAGE AND ALL ACCOMPANYING MATERIAL TO TRACKER SOFTWARE PRODUCTS LTD. OR YOUR AUTHORIZED DEALER FOR A FULL REFUND.

This License Agreement ("Agreement") is a legal agreement between Tracker Software Products Ltd, (Tracker), a Company registered in England, principally located in Turners Hill, West Sussex, England, and you, the user ("Licensee"), and is effective the date Licensee opens the package containing the diskette(s)/CD-R(s) or otherwise uses the enclosed software product.

This Agreement covers all materials associated with Tracker's PDF-XChange Viewer SDK developer's toolkit product, both Dll and ActiveX based options - including the enclosed software product ("Software")

1. GRANT OF DEVELOPMENT LICENSE

TRACKER grants Licensee a non-exclusive, non-transferable, worldwide license for one (1) programmer to install the Software on a single personal computer and use the Software and one copy of the associated user documentation contained in the accompanying user manual, "online" help and Acrobat files ("Documentation") in the development of End User software application's as contemplated in section 2 below (herein, the "Application Software"). If additional programming seats are needed, Licensee should contact TRACKER for discounted license pricing. The license granted hereunder applies only to the designated version of the enclosed Software. If the Software is an upgrade or cross grade, it, and the product that was upgraded/cross graded constitute a single copy of the Software for purposes hereof and the new version and product that was upgraded/cross graded cannot be used by two people at the same time.

2. END USER APPLICATION

The Application Software developed by Licensee must be an "End User Application." An "end user application" is a specific application program that is licensed to a person or firm for business or personal use and not with a view toward redistributing the application or any part of the application, and may be either an application that is used by Licensee internally, or an application that is commercially distributed to end users for their use. A user of an end user application may not modify or redistribute the application and may not copy it (other than for archival purposes). Licensee's license agreement covering the Application Software must contain restrictions prohibiting redistribution, modification and copying of the Application Software.

The license rights hereunder do not apply to development and deployment of software products such as Printer Drivers, ActiveX controls, plug-ins, authoring tools, development toolkits, compilers, operating systems and also software products where a significant function is to generate 'PDF' format files (as defined by Adobe Systems Inc').

Further, with specific regards to any "Royalty Free SDK" Licensing options available - should such be purchased be made by Licensee, Licensee may not, under any circumstances, create a competing software application to Trackers own "PDF-XChange Viewer" for End users, or for which a significant intended purpose is the viewing or manipulation of PDF format files. If Licensee wishes to develop a product outside the scope of this license, Licensee should contact TRACKER'S OEM Sales department to see if a special license is available.

If Licensee wishes to develop a product outside the scope of this license, Licensee should contact TRACKER'S OEM Sales department to see if a special license is available.

3. GRANT OF DUPLICATION AND DISTRIBUTION LICENSE

The Software includes certain runtime libraries and files intended for duplication and distribution by Licensee within the Application Software to the user of Application Software ("Redistributables"). The Redistributable components of the Software are those files specifically designated as being distributable in the "Files to be Included with Your Application" section of the Online Help file, the terms of which are hereby incorporated herein by reference. Licensee should refer to the Documentation and specifically the "Online Help" file for additional information regarding the Redistributables. Under TRACKER'S copyright, and subject to all the restrictions and conditions set forth in this Agreement and the Documentation, TRACKER hereby grants Licensee (and only Licensee) a non-exclusive, non-transferable, worldwide license to reproduce exact copies of the Redistributables and include such files in the Application Software, and to deploy the Application Software internally and/or distribute the Application Software, directly or through customary distribution channels, to end users to the limits prescribed below in Section 4, "Duplication and Distribution of Royalty Bearing Versions " below.) If Licensee wishes to use an OEM who will modify the Application Software and copy it, Licensee must first obtain an OEM distribution license from TRACKER or must require the OEM to obtain a license from TRACKER. Duplication or Redistribution of the Application Software, or any portion thereof, by the users of the Application Software, without a separate written redistribution license from TRACKER is prohibited. If the enclosed Software is packaged "For Evaluation Only," no right to copy and/or distribute the Redistributables is granted. No rights to copy or redistribute the Application Software are granted until such time as Licensee has properly licensed and registered the Software with TRACKER and otherwise complied with this Agreement. Unless otherwise agreed in writing by Tracker.

4. DUPLICATION AND DISTRIBUTION OF ROYALTY BEARING VERSIONS OF THE SOFTWARE

The enclosed software is a Royalty Bearing software kit and may not be distributed Free of Royalties - your initial purchase of one of the software products detailed below includes the right (subject to your acceptance of the terms and conditions of this agreement) to embed within your software application the PDF-XChange Viewer SDK by accessing either the DLL or ActiveX based versions of the PDF-XChange Viewer SDK, subject to an appropriate purchase and distribute a specified number of user licenses to your end user software application clients. When this limit is reached you must purchase additional distribution licenses prior to any further distribution or remove the software from your application prior to further distribution of your application. When calculating your distribution of licenses, each user having access to use of your application incorporating this software must be accounted for individually, Server, Concurrent and Site licensing models are not acceptable or applicable for this purpose.

The following Tracker developer kits include the right to distribute the software for the initial specified number of end user desktop licenses as detailed below, this is an indication only and the actual specified and agreed number of licenses may be different and is detailed and accepted by the parties above the signatures provided on the final page of this document.

PDF-XChange Viewer Basic SDK– Initial max distribution included: 25,000 Single user Licenses

(Available in a variety of predetermined or negotiated license packs)

PDF-XChange PRO SDK – max distribution included: 7500 Single user Licenses

PDF-XChange Drivers API SDK – max distribution included: 2500 Single user Licenses

PDF-XChange Tools SDK – max distribution included: 2500 Single user Licenses

PDF-XChange Viewer SDK extended License packs are available for a variety of volume requirements and on a Royalty Free basis – please contact sales@docu-track.com for more detailed information.

No duplication or distribution rights are granted hereunder with respect to the Royalty Bearing Versions to enable live use and distribution of your application(s) using this software until Tracker have received from you, a copy of this agreement with each page initialed and the last page signed acknowledging your understanding and acceptance of all the terms of this agreement, only then will you receive your license unlock codes enabling use other than for evaluation purposes.

Licensee agrees to account on request by TRACKER for all applications sold or distributed by Licensee or its subsidiaries incorporating the software since its first inclusion in the products of the Licensee - within 28 days of such request having been received from Tracker to the Licensee's contact information as provided (either by post or email). In the event the licensee has exceeded the limits detailed within this agreement or any dispute regarding license volumes & payment Tracker Software Products Ltd may appoint a qualified Auditor to authenticate the records of the Licensee to establish the validity and the Licensee agrees to make all records available pertaining to the Licensee's accounting and other related information, without exception, on written request within 24 hours of receipt of such a request during normal working hours. In the event that such an audit reveals any material inaccuracy in the reporting of the licensee sales and royalty liabilities to TRACKER - Licensee shall:

• Make full payment to TRACKER of all outstanding royalty liabilities within 7 days of demand

• Pay in full all fees and associated costs of the audit howsoever arising

• Immediately cease all sales of all products containing Tracker Software Products Ltd's Toolkits or intellectual property until guarantees of the future reliability of the Licensee's reporting to the satisfaction of Tracker Software Products Ltd are provided.

5. OTHER RESTRICTIONS

The licenses granted under this Agreement are expressly conditioned upon Licensee's compliance with all the terms and conditions of this Agreement. Licensee may not use, copy, rent, lease, sell, sublicense, assign or otherwise transfer the Software except as expressly provided for in this Agreement. Licensee may make a reasonable number of archival copies of the Software. Except for the Redistributables, Licensee shall not distribute any files contained in the Software, including without limitation, .CLW, .INC, .TPL, .CHM, .DRV, .LIB, .H, .MAK, .DEF, .TXT, .PDF or .HLP files. Licensee shall not reproduce, copy or transfer any Documentation, except Licensee may use the sample source code examples contained in the Documentation for the purpose of developing the Application Software. Upon TRACKER'S request, Licensee agrees to send TRACKER one demonstration copy of the Application Software. Any distributor or reseller of Application Software appointed by Licensee must be subject to a binding agreement that includes provisions no less protective of TRACKER'S intellectual property rights in the Software as it is protective of Licensee's rights in its own software. Licensee acknowledges that the Software, in source code form, remains a confidential trade secret of TRACKER and/or its suppliers and therefore Licensee agrees that it shall not modify, decompile, disassemble or reverse engineer the Software or attempt to do so except as permitted by applicable legislation. Licensee agrees to refrain from disclosing the Software (and to take reasonable measures with its employees to ensure they do not disclose the Software) to any person, firm or entity except as expressly permitted herein. Specifically, Licensee will not disclose or publish any license or unlock codes or instruction sets provided by TRACKER relating to the Software. If Licensee wishes to use the Software in a manner specifically or generally prohibited by this Agreement, Licensee should contact TRACKER'S OEM department to determine whether a special license may be required/obtained.

6. PROPRIETARY RIGHTS; COPYRIGHT NOTICES

Except for the limited license granted herein, TRACKER, and its suppliers, retains exclusive ownership of all proprietary rights (including all ownership rights, title, and interest) in and to the Software. Licensee agrees not to represent that TRACKER is affiliated with or approves of Licensee's Application Software in any way. Except as required hereby, Licensee shall not use TRACKER'S name, trademarks, or any TRACKER designation in association with Licensee's Application Software. The Application Software may contain the following copyright notice in the "About box": "Portions of this product were created using PDF-XChange & Image-XChange SDK's From Tracker Software Products Ltd ©2001-6, ALL RIGHTS RESERVED."

7. EXPORT LAW

Licensee acknowledges and agrees that the Software and Application Software may be subject to restrictions and controls imposed by the United States Export Administration Act, as amended (the "ACT"), and the regulations there under. Licensee agrees and certifies that neither the Software nor any direct product thereof (e.g. the Application Software) is being or will be acquired, shipped, transferred or re-exported, directly or indirectly, into any country prohibited by the ACT and the regulations there under or will be used for any purpose prohibited by the same. Licensee acknowledges that the Software may include "technical data" subject to export and re-export restrictions imposed by U.S. law. Licensee bears all responsibility for export law compliance and will indemnify TRACKER against all claims based on Licensee's exporting of the Application Software.

8. U.S. GOVERNMENT RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19, as applicable. Manufacturer/Contractor is TRACKER SOFTWARE PRODUCTS LTD, Units 1-3, Burleigh Oaks, East Street, Turners Hill, West Sussex. England.RH10 4PZ

9. TERM

The license granted hereby is effective until terminated. Licensee may terminate the license by returning the Software and Documentation to TRACKER, without refund, and destroying all copies thereof in any form. TRACKER may terminate the licenses if Licensee fails to comply with any term or condition of this Agreement or any corresponding duplication and distribution agreement for Printer Driver Products. Upon such termination, Licensee shall cease using the Software and cease using or distributing the Application Software containing the Redistributables. All restrictions prohibiting Licensee's use of the Software and intellectual property provisions relating to Software running to the benefit of TRACKER will survive termination of the license pursuant hereto. Termination will not affect properly granted end user licenses of the Application Software distributed by Licensee prior to termination.

10. EXCLUSION OF WARRANTIES

TRACKER and its suppliers offer and Licensee accepts the Software "AS IS." TRACKER and its suppliers do not warrant the Software will meet Licensee's requirements or will operate uninterrupted or error-free. ALL WARRANTIES, EXPRESS OR IMPLIED, ARE EXCLUDED FROM THIS AGREEMENT AND SHALL NOT APPLY TO ANY SOFTWARE LICENSED UNDER THIS AGREEMENT, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.

11. LICENSEE'S REMEDIES: LIMITATIONS

LICENSEE'S SOLE AND EXCLUSIVE REMEDIES AGAINST TRACKER ON ANY AND ALL LEGAL OR EQUITABLE THEORIES OF RECOVERY SHALL BE, AT TRACKER'S SOLE DISCRETION, (A) REPAIR OR REPLACEMENT OF DEFECTIVE SOFTWARE; OR (B) REFUND OF THE LICENSE FEE PAID BY LICENSEE.

12. NO LIABILITY FOR CONSEQUENTIAL DAMAGES

In no event shall TRACKER, or its suppliers, be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information or other pecuniary loss) arising out of use of or inability to use the Software, even if TRACKER or its dealer have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of certain implied warranties or the exclusion or limitation of incidental or consequential damages, in which case and to the extent such exclusion or limitation is not allowed, some of the foregoing limitations and exclusions may not apply to Licensee..

13. GENERAL

This Agreement shall be interpreted, construed, and enforced according to the laws of England. In the event of any action under this Agreement, the parties agree that courts located in England will have exclusive jurisdiction and that a suit may only be brought in England, and Licensee submits itself for the jurisdiction and venue of the courts located in England. This Agreement constitutes the entire agreement and understanding of the parties and may be modified only in writing signed by both parties. No officer, salesman, or agent has any authority to obligate TRACKER by any terms, stipulations or conditions not expressed in the Agreement. All previous representations and agreements, if any, either verbal or written, referring to the subject matter of this Agreement are void. If any portion of this Agreement is determined to be legally invalid or unenforceable, such portion will be severed from this Agreement and the remainder of the Agreement will continue to be fully enforceable and valid. This Agreement, and the rights hereunder, may not be assigned by Licensee, whether by oral or written assignment, sale of assets, merger, consolidation or otherwise, without the express written consent of TRACKER. Licensee agrees to be responsible for any and all losses or damages arising out of or incurred in connection with the Application Software. Licensee agrees to defend, indemnify and hold TRACKER harmless from any such loss or damage, including attorney's fees, arising from the use, operation or performance of the Application Software or Licensee's breach of any terms of this Agreement. Licensee shall be responsible for paying all state and federal use, sales or value added taxes, duties or governmental charges, whether presently in force or which come into force in the future,

related to the distribution and sale of the Application Software and will indemnify TRACKER against any claim made against TRACKER relating to any such taxes or assessments.

PDF-XChange Templates & Classes for Clarion for Windows (PDF-XChange-API/SDK customers only)

PDF-XChange API/SDK (PDF-XChange-API/SDK customers only)

PDF-XChange SDK Printer Driver (PDF-XChange-Print Driver customers only)

PDF-Tools SDK Templates & Classes for Clarion for Windows (PDF-Tools-API/SDK customers only)

Delphi Components for PDF-XChange and/or PDF-Tools SDK products.

All Demo/Evaluation components and examples for PDF-XChange and/or PDF-Tools SDK products.

Image-XChange SDK

PDF-XChange Viewer SDK

1.4. Redistribution of the PDF-XChange Viewer

Redistribution of PDF-XChange Viewer SDK components

The PDF-XChange Viewer SDK depends only on the pxcview.dll, and is not reliant on any other PDF-XChange/Tools Image-XChange SDK components. However, please note, The PDF-XChange Viewer takes advantage of the Microsoft© GDI+ for vector printing and it is required to have installed it on the OS where it is not installed by default (all Windows prior to Windows XP). The PDF-XChange Viewer is available for Windows 2000 and later only - earlier versions of Windows are not supported.

2. Functions

Simple DLL SDK Functions and Structures

2.1. PXCV_CheckPassword

PXCV_CheckPassword validates the supplied password against the current document.

This function should only be called after PXCV_ReadDocumentW returns PS_ERR_DocEncrypted.

```
HRESULT PXCV_CheckPassword(
    PXVDocument Doc,
    BYTE* pPassword,
    DWORD PassLen
);
```

Parameters

Doc specifies the PDF object previously created by the function PXCV_Init.

pPassword specifies a pointer to a buffer which contains password data (buffer may contain zero '\0' symbol(s)).

PassLen *PassLen* specifies the length of the buffer.

Return Values

If the function succeeds, the return value is one of the following:

Value	Meaning
1	User password
2	Owner password

If the function fails, the return value is error code.

2.2. PXCV_Delete

PXCV_Delete releases the PDF object, created previously using the PXCV_Init function.

You must call this function once the PDF object is no longer required or all updates are complete.

```
HRESULT PXCV_Delete(
    PXVDocument Doc
);
```

Parameters

Doc specifies the PDF object previously created by the function PXCV_Init.

Return Values

If the function succeeds, the return value is DS_OK.

If the function fails, the return value is error code.

2.3. PXCV_DrawPageToDC

PXCV_DrawPageToDC draws the specified page (or part thereof) of the document to a specified device context (DC).

```
HRESULT  PXCV_DrawPageToDC(
    PXVDocument Doc,
    DWORD page_num,
    HDC hDC,
    LPPXV_CommonRenderParameters pParams
);
```

Parameters

Doc specifies the document previously created by the PXCV_Init function.

page_num specifies the zero-based page number to be drawn.

hDC specifies handle of the device context onto which the page information should be drawn.

pParams Pointer to the PXV_CommonRenderParameters structure, which defines drawing parameters.

Return Values

If the function fails, the return value is error code.

If the function succeeds, the return value is DS_OK, or other value which isn't an error code.

Example (C++).

```
HRESULT DrawPageThumbnail(PXVDocument pDoc, DWORD page_num, LPCRECT thumb_bound_rect,HDC dc){
    HRESULT hr;
    double pw, ph;
    hr = PXCV_GetPageDimensions(pDoc, page_num, &pw, &ph);
    if (IS_DS_FAILED(hr))
        return hr;
    // calculation rect of thumbnail in pixels
    // (fitting page proportional into thumb_bound_rect)
    LONG tbw = thumb_bound_rect->right - thumb_bound_rect->left;
    LONG tbh = thumb_bound_rect->bottom - thumb_bound_rect->top;
    LONG tw = tbw;
    LONG th = tbh;
    double z1 = (double)tw / pw;
    double z2 = (double)th / ph;
    if (z1 >= z2)
    {
        tw = (LONG)(z2 * pw + 0.5);
     }
    else
    {
        th = (LONG)(z1 * ph + 0.5);
     }
    RECT thumb_rect;
```

```
        thumb_rect.left = thumb_bound_rect->left + (tbw - tw) / 2;
        thumb_rect.top = thumb_bound_rect->top + (tbh - th) / 2;
        thumb_rect.right = thumb_rect.left + tw;
        thumb_rect.bottom = thumb_rect.top + th;
        // now filling PXV_CommonRenderParameters structure
        PXV_CommonRenderParameters crp;
        crp.WholePageRect = &thumb_rect;
        crp.DrawRect = NULL; // because we will draw whole page. It is equal to: crp.DrawRect =
    &thumb_rect;
        crp.Flags = 0;       // should be zero as specified
        crp.RenderTarget = pxvrm_Viewing;
        hr = PXCV_DrawPageToDC(pDoc, page_num, dc, &crp);
        return hr;
    }
```

2.4. PXCV_DrawPageToDIBSection

PXCV_DrawPageToDIBSection creates a Microsoft® Windows® Graphics Device Interface (GDI) DIB section from the specified page (or its rectangular area).

```
HRESULT  PXCV_DrawPageToDIBSection(
    PXVDocument Doc,
    DWORD page_num,
    LPPXV_CommonRederParameters pParams,
    HDC hBaseDC,
    COLORREF backcolor,
    HBITMAP* pResDIBSection,
    HANDLE hSection,
    DWORD dwOffset
);
```

Parameters

Doc specifies a document previously created by the PXCV_Init function.

page_num specifies the zero-based page number to be drawn.

pParams Pointer to the PXV_CommonRenderParameters structure, which defines drawing parameters.

> Please Note that the flag pxvrpf_UseVectorRenderer within the Flags field of the PXV_CommonRenderParameters structure is ignored by this function.

hBaseDC Handle of a device context which may be used for creation of the DIB section. This parameter may be NULL.

backcolor Specifies created bitmap's background color.

> Please Note that the most significant byte is used as transparency value. 0 means fully transparent; 255 means no transparency.

pResDIBSection Pointer to the HBITMAP variable that receives the handle to the GDI DIB section.

hSection Handle of a file-mapping object that the function will use to create the DIB. This parameter may be NULL.

> Note: For more information regarding this parameter, please see the documentation detailing the Windows® GDI function CreateDIBSection.

dwOffset Specifies the offset from the beginning of the file-mapping object referenced by *hSection* where storage for the bitmap bit values begin. This value is ignored if *hSection* is NULL.\

> Note: that the bitmap bit values are aligned on doubleword boundaries, so the offset must be a multiple of the size of a DWORD.

Return Values

If the function fails, the return value is error code.

If the function succeeds, the return value is DS_OK, or other value which isn't an error code.

2.5. PXCV_FinishReadDocument

PXCV_FinishReadDocument Completes the reading of an encrypted document after PXCV_ReadDocumentW returns PS_ERR_DocEncrypted and the correct password was supplied using the PXCV_CheckPassword function.

```
HRESULT PXCV_FinishReadDocument(
    PXVDocument Doc,
    DWORD Flags
);
```

Parameters

Doc Specifies the PDF object previously created by the function PXCV_Init.

Flags This argument is reserved for future use and should be set to 0.

Return Values

If the function succeeds, the return value is DS_OK.

If the function fails, the return value is error code.

Remarks

This function should be called only after PXCV_ReadDocumentW has returned PS_ERR_DocEncrypted and a correct password was supplied using PXCV_CheckPassword. In the case of a successful call to the function PXCp_ReadDocumentW there is no need to call the function.

2.6. PXCV_GetDocumentInfoW

PXCV_GetDocumentInfoW retrieves information from the Info dictionary for the current PDF document (e.g. this same information would be displayed when using a mouse in Windows Explorer and selecting a file - this information becomes viewable when you 'right click' and select the 'Properties' option for the selected file).

```
HRESULT  PXCV_GetDocumentInfoW(
    PXVDocument Doc,
    LPCSTR name,
    LPWSTR value,
    DWORD* valuebuflen
);
```

Parameters

Doc specifies the document that was previously created by the PXCV_Init function.

name Pointer to an ASCII string which defines the information key (e.g. Title, or Author) for the value to be retrieved.

> Please note that *name* is case-sensitive and it is an ASCII string, not UNICODE.

value specifies a pointer to a buffer where the information will be placed. If the value of *value* is NULL, the required buffer size (in chars) will be placed in *valuebuflen*.

Valuebuflen specifies an available buffer size in characters (including a null-terminating character). If *value* is NULL, the function will insert a DWORD variable as pointed to by *valuebuflen* the required buffer size (in chars). If *value* is not NULL, the function will write the actual character count and place into a buffer (including a null-terminating character).

Return Values

If the function fails, the return value is error code.

If the function succeeds, the return value is DS_OK, or another value that is not an error code.

Example (C++).

```
LPCWSTR GetAuthor(PXVDocument Doc)
{
    LPWSTR res = NULL;
    DWORD sz = 0;
    HRESULT hr = PXCV_GetDocumentInfoW(Doc, "Author", res, &sz);
    if (IS_DS_FAILED(hr) || (sz == 0))
        return res;
    res = new WCHAR[sz + 1];
    PXCV_GetDocumentInfoW(Doc, "Author", res, &sz);
    return res;
}
```

2.7. PXCV_GetPageDimensions

PXCV_GetPageDimensions retrieves the dimensions (width and height) of the specified page of the document. This function utilizes the page's rotation and crop box when calculating its dimensions. Returned values are in points (1 point is 1/72 inch).

```
HRESULT  PXCV_GetPageDimensions(
    PXVDocument Doc,
    DWORD page_num,
    double* width,
    double* height
);
```

Parameters

Doc specifies the document previously created by the PXCV_Init function.

page_num specifies the zero-based page number for which dimensions should be retrieved.

width pointer to a double variable, which receives the width of the page (width of crop box of the page), in points.

height pointer to a double variable, which receives the height of the page (height of crop box of the page), in points.

Return Values

If the function fails, the return value is error code.

If the function succeeds, the return value is DS_OK, or other value which isn't an error code.

Example (C++).

```
HRESULT GetPageDimInPixels(PXVDocument pDoc, DWORD page_num, DWORD dpi, SIZE* dims){
    double pw, ph;
    HRESULT hr = PXCV_GetPageDimensions(pDoc, page_num, &pw, &ph);
    if (IS_DS_SUCCESSFUL(hr))
    {
        dims.cx = (LONG)(pw * dpi / 72.0 + 0.5);
        dims.cy = (LONG)(ph * dpi / 72.0 + 0.5);
    }
    return hr;}
```

2.8. PXCV_GetPageRotation

PXCV_GetPageRotation retrieves the specified rotation angle for the specified page. The angle is always a multiple of 90 degrees.

```
HRESULT  PXCV_GetPageRotation(
    PXVDocument Doc,
    DWORD page_num,
    LONG* angle
);
```

Parameters

Doc specifies the document as previously created by the PXCV_Init function.

page_num specifies the zero-based page number whose rotation will be retrieved.

angle pointer to the LONG variable which receives the rotation angle of the page.

Value 0 - means that page isn't rotated; 90 - page is rotated clockwise a quarter turn;

180 - page is rotated by 180 degrees (upside-down); 270 - page is rotated counter-clockwise a quarter turn.

Return Values

If the function fails, the return value is error code.

If the function succeeds, the return value is DS_OK, or other value which isn't an error code.

2.9. PXCV_GetPagesCount

PXCV_GetPagesCount retrieves the page count of the specified document.

```
HRESULT  PXCV_GetPagesCount(
    PXVDocument Doc,
    DWORD* count
);
```

Parameters

Doc specifies the document which was previously created by the PXCV_Init function.

count pointer to a DWORD variable into which to return the page count.

Return Values

If the function fails, the return value is error code.

If the function succeeds, the return value is DS_OK, or other value which isn't an error code.

2.10. PXCV_GetPermissions

The PXCV_GetPermissions function extracts the existing encryption level and user's permissions set for the document.

```
HRESULT  PXCV_GetPermissions(
    PXVDocument Doc,
    DWORD* enclevel,
    DWORD* permFlags
);
```

Parameters

Doc specifies the document which was previously created by the PXCV_Init function.

enclevel specifies a pointer to a variable of the DWORD type, which receives encryption level information for the document. Possible values are 40 and 128.

permFlags specifies a pointer to the variable of the DWORD type which receives permission flag information for the document.

For more information about the value of 'bits' used for permissions, please read the Adobe PDF Specification (section "Standard Encryption Dictionary", table "User access permissions").

Return Values

If the function fails, the return value is error code.

If the function succeeds, the return value is DS_OK, or any other value which is not an error code.

2.11. PXCV_Init

PXCV_Init Creates a PDF object, usually required by the majority of functions in the Simple Dll library .

```
HRESULT  PXCV_Init(
    PXVDocument* pDoc,
    LPCSTR Key,
    LPCSTR DevCode
);
```

Parameters

pDoc pointer to a variable of the type PXVDocument that will receive the created PDF object.

Key pointer to a null-terminated string which contains your license key for use with . This parameter may be NULL; if so, the library will operate in 'evaluation' mode and a demo stamp/watermark will be printed on all output and cannot be removed subsequently.

DevCode pointer to a null-terminated string which contains your individual developer code for use with This parameter may be NULL; if so, the library will operate in 'evaluation' mode and a demo stamp/watermark will be printed on all output and cannot be removed subsequently.

Return Values

If the function succeeds, the return value is DS_OK, and a variable pointer to *pDoc* will contain the valid PDF object.

If the function fails, the return value is error code.

Example (C++).

```
PXVDocument hDocument = NULL;
// Please note - RegCode and DevCode are case sensitive
LPCSTR regcode = "<Your serial/keycode code here>";
LPCSTR devcode = "<Your developers' code here>";
HRESULT res = PXCV_Init(&hDocument, regcode, devcode);
if (IS_DS_FAILED(res))
    return res;
...
PXCV_Delete(hDocument);
```

2.12. PXCV_ReadDocumentFromIStream

PXCV_ReadDocumentFromIStream reads the PDF document using an IStream interface.

```
HRESULT  PXCV_ReadDocumentFromIStream(
    PXVDocument Doc,
    IStream* stream,
    DWORD Flags
);
```

Parameters

Doc specifies the PDF object previously created by the function PXCV_Init.

stream specifies a pointer to the IStream interface for the stream from which the PDF document is to be loaded.

Flags this argument is reserved for further usage and should be set to 0.

Return Values

If the function succeeds, the return value is DS_OK.

If the function return value is equal to PS_ERR_DocEncrypted, then a password must be provided using PXCV_CheckPassword and PXCV_FinishReadDocument must be called to complete reading and parsing the document.

If the function fails, the return value is error code.

2.13. PXCV_ReadDocumentFromMemory

PXCV_ReadDocumentFromMemory reads the document from the specified memory buffer.

```
HRESULT  PXCV_ReadDocumentFromMemory(
    PXVDocument Doc,
    const BYTE* mem,
    UINT size,
    DWORD Flags
);
```

Parameters

Doc specifies the PDF object previously created by the function PXCV_Init.

mem specifies a pointer to a memory buffer containing PDF document to be opened.

size specifies the size in bytes of the buffer pointed to by *mem*.

Flags this argument is reserved for further usage and should be set to 0.

Return Values

If the function succeeds, the return value is DS_OK.

If the function return value is equal to PS_ERR_DocEncrypted, then a password must be provided using PXCV_CheckPassword and PXCV_FinishReadDocument must be called to complete reading and parsing the document.

If the function fails, the return value is error code.

Comments

Memory block passed to the function and should not be released until the function PXCV_Delete has been called.

2.14. PXCV_ReadDocumentW

PXCV_ReadDocumentW reads the document from the specified PDF file.

```
HRESULT PXCV_ReadDocumentW(
    PXVDocument Doc,
    LPCWSTR pwFileName,
    DWORD Flags
);
```

Parameters

Doc specifies the PDF object previously created by the function PXCV_Init.

pwFileName specifies a pointer to a null-terminated UNICODE string that contains the fully qualified path to the file.

Flags this argument is reserved for further usage and should be set to 0.

Return Values

If the function succeeds, the return value is DS_OK.

If the function return value is equal to PS_ERR_DocEncrypted, then a password must be provided using PXCV_CheckPassword and PXCV_FinishReadDocument must be called to complete reading and parsing the document.

If the function fails, the return value is error code.

Example (C++).

```
// Generic example on how to read the document
 PXVDocument hDocument = NULL;
 // Please note - RegCode and DevCode are case sensitive
 LPCSTR regcode = "<Your serial/keycode code here>";
 LPCSTR devcode = "<Your developers' code here>";
 HRESULT res = PXCV_Init(&hDocument, regcode, devcode);
 if (IS_DS_FAILED(res))
     return res;
 hr = PXCV_ReadDocumentW(hDocument, FileName, 0);
if (IS_DS_FAILED(hr))
{
 if (hr == PS_ERR_DocEncrypted)
 {
  while (IS_DS_FAILED(hr))
  {
   BYTE* Password;
   DWORD PassLen;
   // Obtain password (i.e. showing some dialog)
   // ...
   // Check password
   hr = PXCV_CheckPassword(hDocument, Password, PassLen);
  }
  // Finish read document
  hr = PXCV_FinishReadDocument(hDocument, 0);
  if (IS_DS_FAILED(hr))
```

```
    {
      PXCV_Delete(hDocument);
     // In this case document seems to be corrupted
     // ...
    }
   }
   else
   {
    PXCV_Delete(hDocument);
    // In this case document seems to be corrupted
    // ...
   }
  }
  // In this place the document is completely read.
```

2.15. PXCV_ReleaseCachedData

PXCV_ReleaseCachedData releases all cached data from specified document. See Remarks below.

```
HRESULT  PXCV_ReleaseCachedData(
    PXVDocument Doc,
    DWORD dwFlags
);
```

Parameters

Doc specifies the PDF object previously created by the function PXCV_Init.

dwFlags specifies which cached content should be freed:

Name	Value	Meaning
pxvrcd_ReleaseDocumentFonts	0x0002	Release embedded fonts - this is the only flag that may be used.

Note: See PXCV_ReleasePageCachedData for more information on this parameter.

Return Values

If the function succeeds, the return value is DS_OK.

If the function fails, the return value is error code.

Comments

This function clears all currently cached data for the document, so the next rendering operations will require re-reading and converting of some data. However it may free a significant quantity of used memory, so calling this function is recommended after rendering several pages, and especially in the case where already rendered pages are not expected to be reused..

Remarks

The PDF rasterizer requires significant amounts of memory for such things as: sequences of content rendering operators; fonts used for text rendering that are almost always shared between pages; non-embedded fonts may that be shared between documents; and images which may be shared between pages et al. All of these objects must be converted into internal rasterized representations before being used and this may well be a time-consuming operation.

To accelerate page rendering, especially in the case when several parts of same page are rendered sequentially, the rasterizer keeps all objects as internal representations, so that in subsequent rendering operations some objects will not require repeated conversion. But some objects require a lot of memory; for example, a "simple" text page may contain several thousand rendering operators, so it may become necessary to free some or all cached objects to free used memory. To do this the library provides two functions: PXCV_ReleaseCachedData and PXCV_ReleasePageCachedData.

2.16. PXCV_ReleasePageCachedData

The Function PXCV_ReleasePageCachedData releases page-specific cached data for one page in a document and global/shared resources used there-on (optional). See Remarks below for details.

```
HRESULT  PXCV_ReleasePageCachedData(
    PXVDocument Doc,
    DWORD page_num,
    DWORD dwFlags
);
```

Parameters

Doc specifies the PDF object previously created by the function PXCV_Init.

page_num specifies the zero-based page number for which cached data should be released.

dwFlags specifies which cached content should be freed. May be any combination of following flags.

Name	Value	Meaning
pxvrcd_ReleaseDocumentImages	0x0001	Release used images
pxvrcd_ReleaseDocumentFonts	0x0002	Release used embedded fonts
pxvrcd_ReleaseGlobalFonts	0x0004	Release used global (not embedded) fonts

Return Values

If the function succeeds, the return value is DS_OK.

If the function fails, the return value is error code.

Comments

This function does not release all cached data in a document, but only data used to render a specified page. If *dwFlags* is zero (0) then only the content-rendering operators for this page will be released. Using this function is recommended after rendering a page is complete and if it is unlikely that it would be necessary render this page again soon. Further we also recommended calling PXCV_ReleaseCachedData using the pxvrcd_ReleaseDocumentImages flag, as in most cases images are not shared between adjacent pages.

Remarks

The PDF rasterizer requires significant amounts of memory for such things as: sequences of content rendering operators; fonts used for text rendering that are almost always shared between pages; non-embedded fonts may that be shared between documents; and images which may be shared between pages et al. All of these objects must be converted into internal rasterized representations before being used and this may well be a time-consuming operation.

To accelerate page rendering, especially in the case when several parts of same page are rendered sequentially, the rasterizer keeps all objects as internal representations, so that in subsequent rendering operations some objects will not require repeated conversion. But some objects require a lot of memory; for example, a "simple" text page may contain several thousand rendering operators, so it may become necessary to free some or all cached objects to free used memory. To do this the library provides two functions: PXCV_ReleaseCachedData and PXCV_ReleasePageCachedData.

2.17. PXCV_SetCallBack

PXCV_SetCallBack sets the callback function to be used during the PDF rasterization process.

```
HRESULT PXCV_SetCallBack(
    PXVDocument Doc,
    PXV36_CALLBACK_FUNC pProc,
    LPARAM UserData
);
```

Parameters

Doc specifies the PDF object previously created by the function PXCV_Init.

pProc specifies the callback function, which must be defined as:

```
typedef BOOL (__stdcall *PXV36_CALLBACK_FUNC) (DWORD dwStage, DWORD dwLevel, LPARAM param);
```

The first parameter of this function indicates the callback state; the second indicates the progress level (see table below), and the third will always have the same value as passed in *UserData*.

Callback function's state constants table

Constant	Value	Meaning of level
PXCVCIb_Start	1	MaxVal - maximum value of the level which will be passed
PXCVCIb_Processing	2	Current progress level - any value from 0 to MaxVal
PXCVCIb_Finish	3	May be any value from 0 to MaxVal (MaxVal if all passed), may be ignored

Note The Callback function should return TRUE (any non-zero value) to continue processing or FALSE (zero) to abort the operation.

UserData specifies a user-defined callback parameter to be passed as a third parameter to the function specified by *pProc*.

Return Values

If the function succeeds, the return value is DS_OK.

If the function fails, the return value is error code.

2.18. PXV_CommonRenderParameters

The PXV_CommonRenderParameters structure defines drawing parameters for the functions PXCV_DrawPageToDC and PXCV_DrawPageToDIBSection.

```
typedef struct _PXV_CommonRenderParameters {
   LPRECT WholePageRect;
   LPRECT DrawRect;
   DWORD Flags;
   DWORD RenderTarget;
} PXV_CommonRenderParameters;
```

Members

WholePageRect specifies the rectangular area within the Windows® target device's Device Context (DC) coordinate system where the entire PDF page's rectangle would be drawn. See Comments for more details.

DrawRect specifies the rectangular portion of the PDF page to be drawn. If this field is NULL, then the entire PDF page will be drawn.

Flags this DWORD value is a combination of flags, which defines rendering options, such as rotation and vector rendering. May be combination of the following values.

Flags	Value	Meaning
pxvrpf_Rotate_NoRotate	0x0000	Draws the page without any additional rotation.
pxvrpf_Rotate_Rotate90CW	0x0001	Draws the page with a 90 degree clockwise rotation (i.e. landscape).
pxvrpf_Rotate_Rotate180	0x0002	Draws the page with a 180 degree rotation.
pxvrpf_Rotate_Rotate90CCW	0x0003	Draws the page with a 90 degree counter clockwise rotation.
pxvrpf_UseVectorRenderer	0x0004	When this flag is specified, vector rendering, using Microsoft© GDI+®, is used (in place of raster rendering). This feature is recommended when possible for printing as the print job and therefore resources used are considerably smaller.
pxvrpf_RenderAsGray	0x0008	When this flag is specified, rendering will be performed in grayscale mode, in place of color.

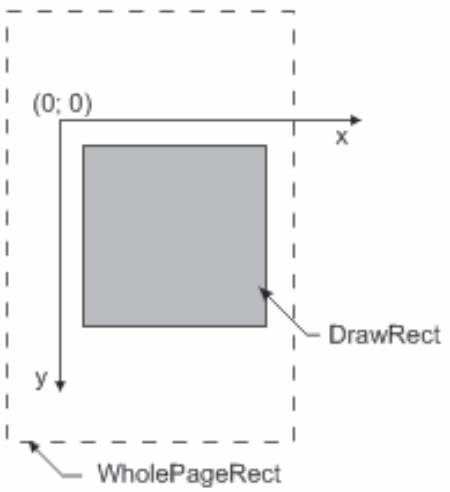
RenderTarget specifies the rendering mode to be used. This is meaningful if the optional content exists within the document which is visible only in some rendering modes, such as Push buttons within an Adobe AcroForm.

May be any one of the following values:

Constant	Value	Meaning
pxvrm_Viewing	0	Rendering target is View. For example, this mode is used for displaying a document on screen.
pxvrm_Printing	1	Rendering target is Print. This mode is used for printing a document (or for a print preview).
pxvrm_Exporting	2	Rendering target is Export. This mode is used for exporting document content to another format (e.g. any one of the supported raster image formats).

Comments

To specify only a part of the PDF page is to be drawn, it is necessary to specify the rectangular area of the target's DC as WholePageRect which the entire PDF page would occupy, and DrawRect, which defines the part of the PDF page which should be drawn within WholePageRect. If DrawRect is set to NULL, then the entire PDF page will be drawn within the target device's WholePageRect area. This simplifies scaling of the PDF page (zoom level), and helps prevent rounding errors during the "points to pixels" conversion that must take place in order to render the PDF information onto the target device.



Example 1.

We want to draw a PDF page within our application window. For example:

1. The PDF page has the dimensions 576 x 792 points (8 x 11 inches).

2. The desired "zoom level" is 400%, or a scaling factor of 4.

3. Our application window's DC has of dimensions 600 x 800 pixels, with a DPI of 96.

4. Our application window has scroll bars to control the page display, and their positions are: 120 for the vertical and 180 for the horizontal, assuming that the maximum position for the horizontal scroll bar is the page's width in pixels, less the window width.

Therefore if is first necessary to calculate the PDF page's dimensions in pixels. This is given by:

```
page_width_in_pixels = (576 / 72) * 96 * 4 = 3072 pixels;
page_height_in_pixels = (792 / 72) * 96 * 4 = 4224 pixels;
```

As long as the PDF page's dimensions and zoom level remain constant, these values will remain constant.

Now WholePageRect and DrawRect values for PXV_CommonRenderParameters structure are:

```
WholePageRect.left = -180; // horizontal scroll position
WholePageRect.top = -120; // vertical scroll position
WholePageRect.right = WholePageRect.left + page_width_in_pixels;
WholePageRect.bottom = WholePageRect.top + page_height_in_pixels;

DrawRect.left = 0;
DrawRect.top = 0;
DrawRect.right = 600; // our window width
DrawRect.top = 800; // our window height
```

For any constant zoom level, WholePageRect depends only on the scroll bars' positions. This minimizes calculations (and rounding errors).

Example 2.

We have a PDF page with width 576 points (8 inches) and height 792 points (11 inches). And we want to draw part of the page as defined by: left = 144pt; top = 288pt; right = 360pt; bottom = 648pt starting from point (10, 10) onto our target DC with a zoom factor of 200%, or a scaling factor of 2. Our DC has a DPI of 96, as do most screens in Windows.

In pixels, the width and height of the page will be:

```
width = (576 / 72) * 96 * (200 / 100) = 1536 pixels;
height = 2112 pixels.
```

Therefore the width of the required portion of the page is:

```
part_width = ((360 - 144) / 72) * 96 * (200 / 100) = 576 pixels;
part_height = 960 pixels.
```

Left-top point of the drawn area we need to locate as at the coordinates (10, 10) onto our DC. So, the left upper point of the complete page will have coordinates:

```
Page_Origin_X = 10 - (144 / 72) * 96 * 200 / 100 = -374;
Page_Origin_Y = 10 - (288 / 72) * 96 * 200 / 100 = -758.
```

The required values are therefore: WholePageRect = {-374, -758, -354 + 1536, -758 + 2112}; and DrawRect = {10, 10, 10 + 576, 10 + 960}.

3. Error handling

» Error codes

» PXCV_Err_FormatFacility

» PXCV_Err_FormatSeverity

» PXCV_Err_FormatErrorCode

3.1. Error codes

Most functions return an HRESULT value which provides a simple means to determine the success or otherwise of a function call.

If the most significant bit or result is set to 1 then the specified error occurred, otherwise the function was successful. Here are two simple macros for C/C++ which apply these checks:

```
#define IS_DS_SUCCESSFUL(x)  (((x) & 0x80000000) == 0)
#define IS_DS_FAILED(x)   (((x) & 0x80000000) != 0)
```

Note: It is strongly recommended to always use the specified (or equivalent macro's) to establish if the function call was successful or otherwise. A simple comparison with 0 (zero) will usually provide erroneous and unreliable results described in the following example scenario's.

A Function may return a warning with a code that is not equal to zero (and also not negative!). This usually means that the function has succeeded and is providing additional information about the call. i.e. The function returns a default value etc. For more information see the description provided for each particular function.

To determine if the return value is generating a warning we provide the IS_DS_WARNING macros.

This would be the correct syntax to check for the error status of the PXCV_CheckPassword function :

```
HRESULT hr = PXCV_CheckPassword(doc, password, len);
if (IS_DS_FAILED(hr))
{
 // An error occurred!
 // Manage the error accordingly to provide an orderly exit from the function call
 ...
}
else
{
 // 'hr' contains value which indicate which password was supplied - owner or user
 ...
}
```

Note: The example code below demonstrates how NOT to provide error checking in your code

```
HRESULT hr = PXCV_CheckPassword(doc, password, len);
if (hr == 0)
{
 // treat as success
 ...
 (this is not true as a positive return value was received!)
 ...
}
else
{
 // treat as error
 (Incorrect as the return value has not been adequately identified and this is unreliable!)
 ...
}
```

Most frequently returned error codes are listed in the table below, however functions may return additional codes which are not listed here. There are 3 further functions available for dealing with errors and may give additional information relating to a specific error: PXCV_Err_FormatSeverity, PXCV_Err_FormatFacility, PXCV_Err_FormatErrorCode. A code example of how to use this table is provided below the table itself. Please note that this function will provide information about all error codes which may be returned.

Possible values of errors of PDF parser/structure:

Constant	Value	Description
PS_ERR_NOTIMPLEMENTED	0x820f04b0	Not implemented
PS_ERR_INVALID_ARG	0x820f0001	Invalid argument
PS_ERR_MEMALLOC	0x820f03e8	Insufficient memory
PS_ERR_USER_BREAK	0x820f01f4	Operation aborted by user
PS_ERR_INTERNAL	0x820f0011	Internal error
PS_ERR_INVALID_FILE_FORMAT	0x820f0002	Invalid file format
PS_ERR_REQUIRED_PROP_NOT_SET	0x820f2716	Required property is not set
PS_ERR_INVALID_PROP_TYPE	0x820f2717	Invalid property type
PS_ERR_INVALID_PROP_VALUE	0x820f2718	Invalid property value
PS_ERR_INVALID_OBJECT_NUM	0x820f2719	Invalid object number
PS_ERR_INVALID_PS_OPERATOR	0x820f271c	Invalid PS operator
PS_ERR_UNKNOWN_OPERATOR	0x820f2787	Unknown operator
PS_ERR_INVALID_CONTENT_STATE	0x820f2788	Invalid content state
PS_ERR_NoPassword	0x820f27a8	No password
PS_ERR_UnknowCryptFlt	0x820f27a9	Unknown crypt filter
PS_ERR_WrongPassword	0x820f27aa	Wrong password
PS_ERR_InvlaidObjStruct	0x820f27ab	Invalid object structure
PS_ERR_WrongEncryptDict	0x820f27ac	Invalid encryption dictionary
PS_ERR_DocEncrypted	0x820f27ad	Document encrypted
PS_ERR_DocNOTEncrypted	0x820f27ae	Document not encrypted
PS_ERR_WrongObjStream	0x820f27af	Invalid object stream
PS_ERR_WrongTrailer	0x820f27b0	Invalid document trailer
PS_ERR_WrongXRef	0x820f27b1	Invalid xref table
PS_ERR_WrongDecodeParms	0x820f27b2	Invalid decode parameter(s)
PS_ERR_XRefNotFounded	0x820f27b3	xref table is not found
PS_ERR_DocAlreadyRead	0x820f27b4	Document is already read
PS_ERR_DocNotRead	0x820f27b5	Document is not read

Comments

There is an additional utility included with this library which provides valuable additional data regarding all known error codes - DSErrorLookUp.exe. This can be found in your PDF-XChange/Tools installation folders and is extremely useful during your application development process - we strongly recommend ALL developers utilize DSErrorLookUp.exe during the debugging of their applications and prior to support requests relating to Error Code return values and their meaning.

Example (C++).

```
// Using of PXCV_Err_FormatSeverity, PXCV_Err_FormatFacility, PXCV_Err_FormatErrorCode functions
char* err_message = NULL;
char* buf = NULL;
_PXCPage* p = NULL;
 // Code below should always return an error and never work
HRESULT dummyError = PXCV_ReadDocumentW(NULL, NULL, 0);
LONG sevLen = PXCV_Err_FormatSeverity(dummyError, NULL, 0);
LONG facLen = PXCV_Err_FormatFacility(dummyError, NULL, 0);
LONG descLen = PXCV_Err_FormatErrorCode(dummyError, NULL, 0);
if ((sevLen > 0) && (facLen > 0) && (descLen > 0))
{
 // Total length of the formatted text is the sum of the length for each description
 // plus some additional characters for formatting
 LONG total = sevLen + facLen + descLen + 128;
 // allocate buffer for message
 err_message = new char[total];
 err_message[0] = '\0';
 // allocate temporary buffer
 buf = new char[total];
 // get error severity and append to message
 if (PXCV_Err_FormatSeverity(dummyError, buf, total) > 0)
  lstrcat(err_message, buf);
 lstrcat(err_message, " [");
 // get error facility and append to message
 if (PXCV_Err_FormatFacility(dummyError, buf, total) > 0)
  lstrcat(err_message, buf);
 lstrcat(err_message, "]: ");
 // and error code description and append to message
 if (PXCV_Err_FormatErrorCode(dummyError, buf, total) > 0)
  lstrcat(err_message, buf);
 ::MessageBox(NULL, err_message, "Test error", MB_OK);
 delete[] buf;
 delete[] err_message;
}
```

3.2. PXCV_Err_FormatFacility

PXCV_Err_FormatFacility returns information of where an error occurred for the specified error code.

```
LONG  PXCV_Err_FormatFacility(
    HRESULT errorcode,
    LPSTR buf,
    LONG maxlen
);
```

Parameters

errorcode specifies the HRESULT returned by a function.

buf specifies a pointer to a buffer where the error facility information will be returned.

> Note: To determine the required buffer size you should pass NULL for *buf*.

maxlen specifies the available buffer size in characters (including null-terminating character).

Return Values

If the function fails to recognize an error code the return value is negative.

If the function fails to retrieve information about an error code the return value is zero.

If the function successfully retrieves information and the parameter *buf* is NULL the return value is the number of characters required to store the description (including a null-terminating character).

If the function successfully retrieves information and the parameter *buf* is not NULL the return value is the number of characters written to the buffer (including null-terminating character).

3.3. PXCV_Err_FormatSeverity

PXCV_Err_FormatSeverity returns information regarding the error severity. See Error codes for additional information.

```
LONG  PXCV_Err_FormatSeverity(
    HRESULT errorcode,
    LPSTR buf,
    LONG maxlen
);
```

Parameters

errorcode specifies the HRESULT returned by a function.

buf specifies a pointer to a buffer where the error facility information will be returned.

> Note: To determine the required buffer size you should pass NULL for *buf*.

maxlen specifies the available buffer size in characters (including null-terminating character).

Return Values

If the function fails to recognize an error code the return value is negative.

If the function fails to retrieve information regarding an error code the return value is zero.

If the function successfully retrieves information and the parameter *buf* is NULL the return value is the number of characters required to store the description (including a null-terminating character).

If the function successfully retrieves information and the parameter *buf* is not NULL the return value is the number of characters written to the buffer (including a null-terminating character).

3.4. PXCV_Err_FormatErrorCode

PXCV_Err_FormatErrorCode provides information relating to an error code.

```
LONG  PXCV_Err_FormatErrorCode(
    HRESULT errorcode,
    LPSTR buf,
    LONG maxlen
);
```

Parameters

errorcode specifies the HRESULT returned by a function.

buf specifies a pointer to a buffer where the error facility information will be returned.

> Note: To determine the required buffer size you should pass NULL for *buf*.

maxlen specifies the available buffer size in characters (including null-terminating character).

Return Values

If the function fails to recognize an error code the return value is negative.

If the function fails to find information regarding the error code the return value is zero.

If the function successfully retrieves information and the parameter *buf* is NULL the return value is the number of characters required to store the description (including a null-terminating character).

If the function successfully retrieves information and the parameter *buf* is not NULL the return value is the number of characters written to a buffer (including a null-terminating character).

4. Tracker Software Products

Tracker Software Products

Who are we and what do we do?

We at Tracker Software Products take great pride in the software products we create and distribute. We sell our products directly, via Distributors, Resellers and OEM partners - in some cases with our products and larger partners these products are sold under different labels and names than those we sell directly, this is to allow our partners to build a following for their own brand and protect their future, our printer drivers however, whilst allowing you to rename them in the user's Printers list - do not allow 100% rebranding.

No matter how our products reach you, we want you to experience the best results possible - please do contact us if for any reason you are dissatisfied or have a suggestion how we can improve our product offerings.

You may also be interested in related products available from us - in the following brief topics you will find details of how to contact us, request support and summary details of the products available from us for both 'End Users' and Software Development tools for other Software developers to utilize in their product offerings.

Please do contact us if you cannot find the information you require within this manual/help file.

4.1. Contact Us

How to contact us

Head Office

Tracker Software Products (Canada) Ltd
#3-466 TransCanada Highway
Duncan. BC
V9L 3R6
Canada

Sales/Admin only (see below for Support) :
Tel: +1 (250) 597-1621
Fax : +1 (250) 597-1623

UK & EUROPE

Tracker Software Products Ltd
Units 17 Raleigh Court
Priestly Way
Crawley RH10 9PD
Sussex
England.

VAT No: GB702613671

Sales/Admin only (see below for Support) :
Tel: +44 (0)20 8555 1122
Fax: +44(0)844 800 4521Our Web Site: http://www.docu-track.com

We also have offices and representatives in several other locations including: United States, France, Germany and Ukraine - in some instances after an initial contact with our UK office you may be referred to one of these locations if appropriate.

We recommend you use our user forums at http://www.docu-track.com/forum/index.php and scan the existing library of questions and answers, if you don't find a suitable response then feel free to post your own - all questions receive an answer within 1 business day at worst!

If for any reason you have difficulty linking to the forum or feel it is inappropriate for your needs then please support@docu-track.com, we regret we cannot answer support requests via telephone without a valid support contract. The number above is answered by administration staff who are not trained to assist with technical problems.

To Contact us for Sales/Administration related issues:

sales@docu-track.com - End User, Developer and OEM.

admin@docu-track.com

All this information and a good deal more is available via our web site and the links provided.

Magazine reviews and press requests.

We are keen to assist in any way possible - please contact our sales department for any information or help you may require.

4.2. Products

Our Products

Products Offered By Tracker Software Products Updates Can be downloaded from our Web site.

Passwords and serial numbers are changed from time to time in an attempt to thwart unauthorized use of pirated software and passwords.

End User/Retail Products

You can Purchase Direct from our web site and be using any of our products the same day!

PDF-XChange - Create fully native Adobe compatible PDF Files from virtually any Windows 32 Bit software application.

PDF-Tools - Create and manipulate Adobe PDF Files and batch Convert Images to PDF Files and more...

PDF-XChange Viewer - Upgrade your Acrobat PDF Viewer to the Free and extended PDF viewer offered by Tracker and improve your PDF experience!

TIFF-XChange - Create industry standard TIFF files in CCIT Grp 3/4 and unpacked formats from virtually any Windows 32 Bit software application.

Raster-XChange - Create Raster Image Files from Windows application.

Software Developers SDK's and other Products - all SDK's offer Royalty Free Distribution of the 'End User' components.

You can Purchase Developer SDK's from our web site and be using any of our products the same day!

PDF-XChange - Create fully native Adobe compatible PDF Files from your application output.

PDF-XChange Viewer - Display and manipulate Adobe compatible PDF Files from your application output.

PDF-Tools - Create and manipulate Adobe PDF Files and batch Convert Images to PDF Files and more...

TIFF-XChange - Create industry standard TIFF files in CCIT Grp 3/4 and unpacked formats from your application output.

Image-XChange SDK - Print, Convert, Scan and View Imaging formats!

Raster-XChange - Create Raster Image Files from your application output.

Licensing restrictions apply to SDK's, the essential limitations being that you may not create products for Developer Tools for use by others, distribute any SDK components not essential to end user use or provide functionality that is not in keeping with our licensing terms.

Trial Versions

All of our products are available as fully functional evaluation downloads for you to try before you buy - usually printing a demo watermark/stamp to differentiate between output created with the evaluation or licensed versions. We recommend that all users test the product they wish to buy first - thus ensuring you only buy once you are 100% happy the product meets your needs.

Trial versions are available from our web site: For more details visit our web site or contact us by email.

INDEX