

Whitepaper

## soft Xpansion Print to Document 7

Die Technologie für virtuelles Drucken und  
ein universelles Format elektronischer Dokumente

Copyright 2002-2009 soft Xpansion GmbH & Co. KG

## Inhaltsverzeichnis

Digitale Dokumente .....	2
Ein universelles Format.....	2
Print to Document .....	3
Version 7 .....	3
Virtueller Drucker.....	4
Lösungsarchitektur.....	4
Technologische Komponente .....	5
Ordner und Dateien im SDK.....	7
Systemvoraussetzungen.....	8
Lizenzen und Preise .....	8
Komponenten- und Drucker-Setup.....	9
Interface IP2DSetup70 .....	10
Grundlegende Regeln und Informationen .....	15
Konfiguration und Drucker-Ereignisse .....	16
Konfiguration des Druckers in der Windows-Registry .....	18
Struktur und Werte des Registry-Schlüssels.....	19
Die Ereignisse im Treiber und Interface IP2DEvents70.....	23
Die Ereignisse im P2D-Drucker .....	24
Interface I2PEvents70.....	25
Interfaces I2PDialog70 .....	26
Druckerkonfiguration in einer Applikation (Besonderheiten bei P2D-Druckern).....	27
Escape-Codes - Erweiterung der Print to Document-Technologie .....	28
Verarbeitung von Druckaufträge .....	28
Print Processor und Konnektor.....	28
Konnektor-Typen .....	29
Interface IP2DHSJobProcessor70 .....	30
Interface IP2DJob70 .....	31
Interface IP2DConnector70.....	31
Interface IP2DHSJob70.....	32
Job Info.....	33
Job Info - Erweiterung der Print to Document-Technologie .....	34

## **DIGITALE DOKUMENTE**

Im Zuge der zweifellos in vielen Bereichen zunehmenden Digitalisierung des Dokumentverkehrs steigt die Verbreitung von elektronischen Dokumenten. Der Umstieg auf digitale Lösungen ermöglicht Kosteneinsparungen, eine Effizienzsteigerung durch Automatisierung und den Verzicht auf eine Vielzahl überflüssiger Bearbeitungsschritte.

Die Verdrängung der traditionellen Papierform geht mit einer Zunahme an unterschiedlichen digitalen Formaten einher. Und hier liegt ein Nachteil von papierlosen Dokumenten: Es existiert kein universelles, einheitliches Format für digitale Daten. Vielmehr stehen sehr viele Formate zur Verfügung, die wiederum eine entsprechende Software benötigen, um die Daten weiter zu verarbeiten, auszudrucken und zu archivieren.

Um die Kosten- und Effizienzvorteile von digitalen Dokumenten in vollem Umfang nutzen zu können, kommt es somit darauf an, ein solches universelles Format zu verwenden. In dieses Format können dann alle anderen Formate, zum Beispiel die weit verbreiteten MS Office-Dokumente, PDF-Dateien, Raster- und Vektorbilder, konvertiert werden.

## **EIN UNIVERSELLES FORMAT**

Nahezu jede Software verfügt über eine Druckfunktion, unabhängig von den Datenformaten, die sie unterstützt und verwendet. In manchen Fällen fehlt zwar die Möglichkeit, bestimmte Daten zu speichern; ein Ausdrucken ist aber möglich. Ein Beispiel ist Software, die Berechnungen dynamisch durchführt und die Ergebnisse ausdruckt, ohne sie vollständig speichern zu können.

Über das Drucken von (internen) Daten aus beliebigen Formaten bietet jede druckfähige Software die Möglichkeit, diese Daten in ein universelles Format umzuwandeln: Der Druckvorgang wird vom Betriebssystem gesteuert - um Ausdrücke zu generieren, werden die Daten aus den verwendeten Formaten in ein Standardformat umgewandelt und dem Druckersystem übergeben.

Dieses Standardformat hat allerdings zwei Nachteile:

- 1) Es können nur sichtbare Informationen umgewandelt und exportiert werden. Alle "im Hintergrund" liegenden Daten werden nicht exportiert, auch dann nicht, wenn sie sich im Dokument befinden. So werden zum Beispiel Hyperlinks und andere Strukturelemente nicht

übernommen. Auch Video- oder Audiodateien werden nicht beziehungsweise nicht vollständig exportiert.

2) Die umzuwandelnden Daten werden immer auf die Seiten verteilt, die Inhaltselemente der Seiten haben immer bestimmte Koordinaten auf einer Seite. Zum Beispiel wird der Inhalt einer umfangreichen Webseite immer auf die einzelnen Seiten verteilt. Die Inhaltselemente haben fixierte Positionen, die von den Seitenabmessungen abhängig sind.

Dieses universelle Format wird in Windows als Enhanced Metafile Format (EMF) bezeichnet.

Trotz der erwähnten Nachteile gibt es gegenwärtig in Windows keine wirkliche Alternative zum EMF-Format. Aus diesem Grund ist EMF als universelles Format erste Wahl.

Angenommen, Ihre Aufgabe besteht darin, Daten aus einem oder mehreren Programm(en) in ein gemeinsames Format zu übertragen. Wie kann diese Aufgabe gelöst werden?

## PRINT TO DOCUMENT

### Version 7

Die Entwicklung der Print to Document-Technologie hat im Jahre 2002 begonnen. Als erstes Ergebnis wurde das Produkt "PDFs leicht gemacht" veröffentlicht. In dieser Software wurde die Technologie für die PDF-Erstellung mittels Drucken auf einem [virtuellen Drucker](#) eingesetzt.

Seitdem wird diese Technologie permanent weiter entwickelt, sodass nahezu jedes Jahr eine neue Version der **Print to Document**-Software veröffentlicht werden kann. Die auf Basis der Technologie erstellten virtuellen Drucker kommen heute in mehreren **soft Xpansion**-Produkten zum Einsatz. Seit der Version 5 ist die Technologie auch für externe Entwicklerteams als SDK verfügbar und wird als separates Produkt angeboten.

Insgesamt sind seit der ersten Version bis heute über **100.000** Software-Installationen in den Bereichen Drucken, Publishing und PDF erfolgt. Gleichzeitig ist dank der hohen Qualität und Zuverlässigkeit der Technologie die Zahl der Supportanfragen sehr gering.

Die **Version 7** wartet mit einer Reihe von Änderungen auf, auch grundsätzlicher Art. Die wichtigsten Ursachen für die Änderungen sind die Weiterentwicklung der gesamten Windows-Umgebung und die Anforderungen anderer Software. Da die aktuelle Version **Print to**

**Document 7** nicht mehr die alten Betriebssysteme Windows 95, Windows 98, Windows ME und Windows NT unterstützt, ist sie flexibler und produktiver geworden.

Die neuen Betriebssysteme **Windows Vista** und Server 2008 bedeuten, unter anderem wegen der Varianten mit **64-Bit**-Architektur, neue Anforderungen für die Softwareentwicklung. Ihre Verbreitung sowohl im geschäftlichen Bereich als auch bei Privatanwendern nimmt zu. Die Version 7 von Print to Document ist bereits an diese Architektur angepasst und darüber hinaus in Netzwerkumgebungen auf den verschiedenen Betriebssystemen im Einsatz.

Daneben wurden die eigentlichen Funktionen und Möglichkeiten der Technologie erweitert. Auch die Produktivität und Effizienz sind im Vergleich mit den vorherigen Versionen gestiegen. Die konsequent erweiterte Funktionalität geht dabei nicht zulasten der Zuverlässigkeit.

### **Virtueller Drucker**

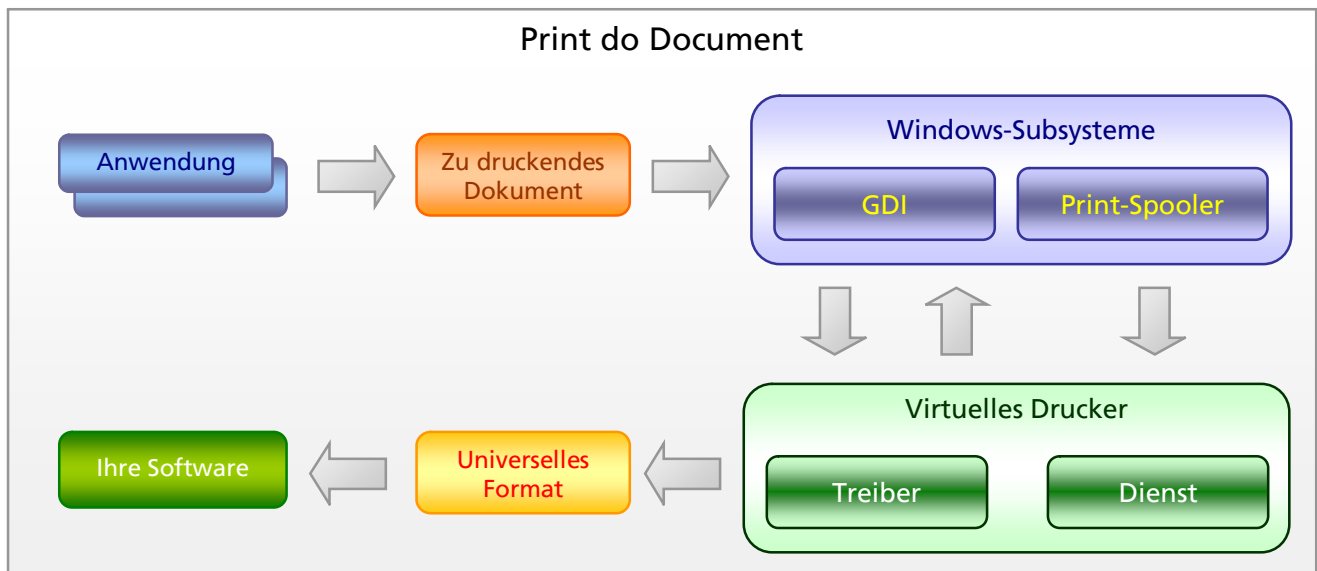
Die **Print to Document**-Technologie basiert auf einem so genannten virtuellen Drucker. Dieser virtuelle Drucker (eine Software) wird auf einem Windows-Computer wie ein "normaler" physikalischer Drucker (Hardware) installiert. Die Anwender verwenden ihn wie alle anderen Drucker und es ist keine zusätzliche Anpassung notwendig, um ihn einsetzen zu können.

Auch der virtuelle Drucker wird im "Drucken"-Dialog aller druckfähigen Anwendungen und im Startmenüeintrag "Einstellungen"-"Drucker und Faxgeräte" auf dem Computer eingerichtet, mit ihm ist aber kein angeschlossenes Hardware-Gerät assoziiert. Er entspricht somit aus der Sicht des Betriebssystems - und damit auch für die Anwendungssoftware auf dem Computer - einem "echten" Drucker, der Informationen allerdings nicht auf Papier ausgibt. Der virtuelle Drucker erhält von der Anwendung, die ihn zum "Drucken" verwendet, die notwendigen Daten in einem [universellen digitalen Format](#).

Nachdem die Daten zum virtuellen Drucker gesandt und in ein universelles Format konvertiert wurden, ist die Hauptaufgabe des virtuellen Druckers bereits erledigt. Die Daten müssen lediglich noch der Software übergeben werden, die sie angefordert hat.

### **Lösungsarchitektur**

Das folgende Schema zeigt, wie eine Software unter Verwendung der **Print to Document**-Technologie Daten aus einer anderen, beliebigen Software in ein [universelles Format](#) umwandelt.



Der Druckvorgang in Windows wird immer indirekt, durch spezielle Subsysteme, ausgeführt und zwar durch das grafische Subsystem (GDI) und durch den Print-Spooler.

Das GDI stellt den Anwendungen die Standardschnittstelle für das Drucken (Print API) bereit. Diese Schnittstelle standardisiert den Datenausdruck aus beliebigen Anwendungen und übernimmt die Besonderheiten verschiedener Drucker und Druckanlagen.

Das GDI kommuniziert mit einem [Druckertreiber](#) und einer Anwendung über einen speziellen Befehlssatz. So wird ein sogenannter Druckauftrag erstellt - die Daten im ausgewählten Format. Diese Daten werden als Spool-Datei dem Print-Spooler des Systems übergeben. Dieser überträgt sie an den speziellen Druckertreiber, den [Print Processor](#). Falls das Drucken auf einem Netzwerkdrucker erfolgt, liefert der Print-Spooler die Spool-Datei über das Netzwerk an den Computer, auf dem der Netzwerkdrucker unmittelbar installiert ist. Somit werden die Daten nicht nur in ein [universelles Format](#) konvertiert, sondern auch an ein bestimmtes Ziel geliefert.

### Technologische Komponente

Die **Print to Document**-Software enthält drei Treiber:

- Konfigurationstreiber
- Render-Treiber
- Print Processor

Konfigurationstreiber und Render-Treiber sind in einem Modul zusammengefasst. Außerdem wird eine Bibliothek für die Installation der Treiber und Drucker mitgeliefert.

Als optionale Komponente ist ein spezielles Hostprogramm enthalten. Es gewährleistet während der Bearbeitung für jeden Druckauftrag den Anwender-Kontext anstelle des System-Kontexts.

Der **Konfigurationstreiber** definiert das Programm-Interface als Benutzerinterface-Komponente in einer Anwendung. In der Komponente werden für die Benutzer die Druckereinstellungen angezeigt und können durch diese verändert werden. Die Komponente reagiert auch auf Ereignisse, die beim Drucken auf dem [virtuellen Drucker](#) eintreten (zum Beispiel "CreateDC", "StartDoc").

Der **Render-Treiber** wandelt die grafischen Befehle (gedruckte Daten) aus der Anwendung in das [universelle Datenformat](#) (EMF-Datei) um. Diese Datei wird als Druckjob an den Print-Spooler gesandt.

Der **Print Processor** wird durch den Print-Spooler aufgerufen. Er empfängt die Daten des Druckauftrags und zusätzliche Informationen über den Druckauftrag. Der Treiber definiert das Programm-Interface für die Verarbeitungskomponente ([Konnektor](#)) in Ihrer Anwendung. Hier wird die Logik zur Umwandlung der empfangenen Daten in das universelle Format (EMF-Dateien) implementiert.

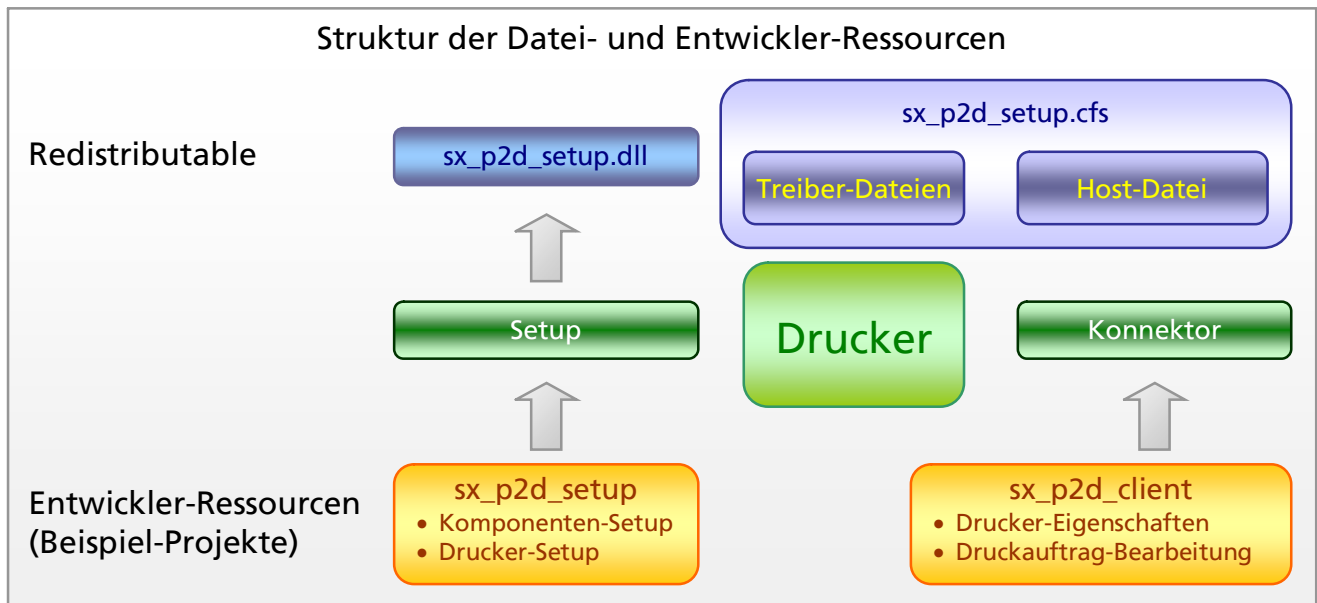
Die **Installationsbibliothek** enthält die Methoden für die [Installation und Deinstallation](#) aller Komponenten der Print to Document-Technologie, sowie für die Installation, Deinstallation und Konfiguration von einem oder mehreren [virtuellen Druckern](#).

Das Hostprogramm wird bei Bedarf verwendet und erfüllt dabei drei Aufgaben:

- 1) Der Print Processor wird immer von dem Print-Spooler aufgerufen, der im Kontext des Systemkontos gestartet wird. Dementsprechend wird der Aufruf des [Konnektors](#) in Ihrer Anwendung vom Print Processor ebenfalls im Kontext des Systemkontos ausgeführt. Dies ist allerdings nicht immer passend, da eine Kommunikation mit dem Endanwender erforderlich ist oder die Daten in demselben Benutzer-Kontext der druckenden Anwendung bearbeitet werden. Der Print Processor startet den speziellen Prozess (Host) im Kontext des erforderlichen Benutzers, und dieser Prozess ruft den [Konnektor](#) für die Bearbeitung des Druckauftrags auf. In einigen Fällen ist es zweckmäßig, den Host auch für die Bearbeitung des Druckauftrags unter dem Systemkonto zu verwenden. So wird der Print-Spooler nicht gesperrt, da er die Druckaufträge nacheinander an den Print Processor übergibt.
- 2) Die zweite Aufgabe ist die parallele Multithread-Bearbeitung der Druckaufträge. Da der Print-Spooler die Druckaufträge immer

nacheinander bearbeitet, und manche Aufträge viel Zeit erfordern, steigert der Host die Produktivität der Bearbeitung bei Computern mit mehreren Prozessoren deutlich.

3) Die dritte Aufgabe, die vom Hostprogramm erfüllt wird, ist die temporäre Sicherung von Druckaufträgen. Diese tritt an die Stelle der Benachrichtigung des [Konnektors](#) zur sofortigen Bearbeitung. In diesem Fall kann die Anwendung später mittels Host-Prozess über eine spezielle Schnittstelle abfragen, ob nicht bearbeitete Druckaufträge vorhanden sind und die Bearbeitung ausführen.



Das **Print to Document**-SDK enthält zwei [Beispiel-Projekte](#). Die beiden Beispiele sind in C++ programmiert. Allerdings kann die Print to Document-Technologie mit beliebigen Programmiersprachen verwendet werden, die das COM-Interface unterstützen.

Das Projekt **sx\_p2d\_setup** zeigt, wie man die Technologie-Komponente und den Drucker mithilfe der Setup-Komponente installiert.

Das Projekt **sx\_p2d\_client** zeigt, wie man den [Konnektor](#) zwischen dem Drucker und Ihrer Software implementiert sowie die Organisation der Bearbeitung der Druckaufträge durch den Drucker. Zudem enthält das Projekt Informationen zur Erstellung eines Dialogs für die Druckereigenschaften und für die weiteren Druckeinstellungen.

### Ordner und Dateien im SDK

**DevRes** - der Ordner mit den Header- und IDL-Dateien, die der Entwickler für die Integration von Print to Document in die eigene Software benötigt, sowie die Beispiel-Projekte.

**Docs** - die Dokumentation von Print to Document.



**Redist** - die Dateien, die in die ausgelieferte Anwendung implementiert werden müssen und auf den Endanwender-Computer installiert werden.

**Trial** - die Test-Version für den Drucker, um Print to Document auf einem Entwicklersystem auszuprobieren.

### Systemvoraussetzungen

Folgende Windows-Plattformen: Windows 2000, Windows XP, Windows Server 2003, **Windows Vista**, Windows Server 2008, alle Betriebssysteme mit 32 und **64 Bit**.

VMWare wird ebenfalls unterstützt.

### Lizenzen und Preise

Mit einer Lizenz wird immer eine einzelne Druckerinstanz autorisiert. Eine Druckerinstanz ist ein [virtueller Drucker](#) als separates Produkt, das wiederum auf beliebig vielen Computern installiert werden darf. Auf einem Computer darf dabei immer nur eine Druckerinstanz installiert werden. Der Druckername wird vom Softwareentwickler vergeben und kann für verschiedene Computer und Anwendungen unterschiedlich sein.

Die **Professional-Lizenz** erlaubt die Verwendung des Druckers für Ausdrücke aus beliebigen Anwendungen auf beliebig vielen Computern. Nicht erlaubt sind Massen- oder Serienausdrücke, wenn der Drucker zum Beispiel als Teil eines Dokumentenmanagementsystems eingesetzt wird. Massenverarbeitung liegt bei einer Überschreitung von 30 Dokumenten in einer (jeder) Stunde vor.

Die **Business-Lizenz** erlaubt auch eine Massenverarbeitung, und zwar ohne weitere Beschränkungen hinsichtlich der Anzahl an Dokumenten.

Die Preise entnehmen Sie bitte der folgenden Tabelle. Die Basispreise verstehen sich für ein Land oder einen Sprachraum, zzgl. MwSt.

Lizenz oder Bedingung	Preis (€) oder Aufpreis (%)	Bemerkungen
Professional-Lizenz, eine Instanz	4900,-	Pauschaler Preis
Business-Lizenz, eine Instanz	4900,-	Preis pro Jahr
Zweite Instanz	50%	
Dritte und jede weitere Instanz	25%	Pro Instanz
Netzwerkdrucker für Professional-Lizenz	25%	
Netzwerkdrucker für Business-Lizenz	Im Basispreis inklusive	
Weltweite Nutzungsrechte	25%	
Wartung für Professional-Lizenz	25%	Preis pro Jahr
Wartung für Business-Lizenz	Im Basispreis inklusive	

## KOMPONENTEN- UND DRUCKER-SETUP

Das SDK enthält die Komponente **sx\_p2d\_setup.dll**, die dem Entwickler die Methoden für die Installation, Konfiguration und Deinstallation anderer Komponenten und [virtueller Drucker](#) zur Verfügung stellt. Mithilfe dieser Komponente erfordert die Installationsprozedur lediglich einige Methoden-Aufrufe mit den entsprechenden Konfigurationsparametern. Die Komponente kann man über das COM-Interface [IP2DSetup70](#), oder durch exportierte Funktionen im C-Standard verwenden. Somit kann der Entwickler diese Komponente auch in Applikationen einsetzen, die in anderen Programmiersprachen geschrieben wurden (inklusive .NET), und in Scripten mit unterschiedlicher Installationssoftware wie zum Beispiel MSI (Windows Installer) oder InstallShield verwenden.

Das SDK enthält auch ein Beispiel, das in C++ geschrieben wurde und die Setup-Komponente verwendet. Es zeigt eine typische Installation, Konfiguration und Deinstallation des virtuellen Druckers **Print to Document 7**. Dabei wird eine [Testlizenz](#) verwendet. Man kann den Testdrucker installieren und das Drucken aus jedem beliebigen Programm testen. Die Bearbeitung der ausgedruckten Daten ist mit der Testlizenz nicht möglich, die gedruckten Seiten werden lediglich im Drucker-Dialog angezeigt. Nach dem Schließen des Dialogs werden die gedruckten Daten gelöscht.

Die Setup-Komponente benötigt keine weitere Software und verwendet ausschließlich die Systemfunktionen von Windows. Die Komponente erfordert Administratorrechte der entsprechenden Anwender. Das heißt, dass für den Anwender der Benutzerkontotyp "Computeradministrator" eingerichtet sein muss. Für Vista und neuere Windows-Versionen mit aktivierter Benutzerkontensteuerung (UAC) gilt eine zusätzliche Bedingung: die Applikation, die die Setup-Komponente verwendet, muss über ein Manifest verfügen, in dem definiert ist, dass Administratorrechte für die Applikation erforderlich sind. In dem Beispiel im SDK ist das erforderliche Manifest bereits vorhanden.

Bevor die Komponente verwendet werden kann, müssen sie und die Datei [sx\\_p2d\\_setup.cfs](#) auf die Festplatte kopiert werden. Zudem muss die Komponente standardmäßig als COM-Komponente registriert werden.

## Interface IP2DSetup70

Das Interface stellt dem Entwickler die Methoden für die Installation, Konfiguration und Deinstallation von SDK-Komponenten und [virtuellen Druckern](#) zur Verfügung.

```
OpenLibrary([in] BSTR sResPath);
```

Initialisierung der [Setup-Komponente](#) vor der Verwendung.

*sResPath* - absoluter Pfad zur Datei **sx\_p2d\_setup.cfs**, die zusammen mit der Setup-Komponente geliefert wird.

```
CloseLibrary();
```

Deinitialisierung der Setup-Komponente.

```
BSTR GetVersion();
```

Die Versionsnummer des **Print to Document-Treibers** in einer CFS-Datei.

Sie wird als String zurückgegeben, zum Beispiel "7.1.3.0".

```
BSTR GetDriverVersion([in] BSTR Name, [in] boolean b64);
```

Die Versionsnummer des **Print to Document-Treibers**, der auf dem Computer installiert ist, falls ein Treiber vorhanden ist.

Sie wird als String zurückgegeben, zum Beispiel "7.1.0.0", oder als "0", wenn der Treiber nicht installiert ist.

*Name* - der Name des **Print to Document-Treibers**.

*b64* - falls TRUE, wird die Versionsnummer des 64-Bit-Treibers zurückgegeben, ansonsten die des 32-Bit-Treibers.

```
BSTR GetDriverProcVersion([in] BSTR Name);
```

Die Versionsnummer des [Print Processors](#), der gegebenenfalls auf dem Computer installiert ist.

Sie wird als String zurückgegeben, zum Beispiel "7.1.0.0", oder als "0", wenn der Print Processor nicht installiert ist.

*Name* - der Name des Print Processors.

```
InstallDriver([in] BSTR Filename, [in] BSTR Name, [in] boolean b32, [in] boolean b64);
```

Die Methode zur Installation des **Print to Document-Treibers**. Wenn der Treiber bereits in einer älteren Version installiert ist, wird er aktualisiert. Nach der Aktualisierung muss Windows neu gestartet werden.

*Filename* - Name der Treiber-Datei. Mit *Name* wird der Treiber selbst installiert.

*Name* - Name des Treibers, mit dem er installiert wird.

*b32* - falls TRUE, wird der 32-Bit-Treiber installiert.

*b64* - falls TRUE, wird der 64-Bit-Treiber installiert.

```
UninstallDriver([in] BSTR Name);
```

Die Methode zur Deinstallation des Print to Document-Treibers. Nach der Deinstallation muss Windows neu gestartet werden.

*Name* - der Name des Treibers.

```
InstallDriverProc([in] BSTR Filename, [in] BSTR Name);
```

Die Methode zur Installation des **Print Processors**. Wenn der Print Processor bereits in einer älteren Version installiert ist, wird er aktualisiert. Nach der Aktualisierung muss Windows neu gestartet werden.

*Filename* - Name der Print Processor-Datei. Mit *Name* wird der Print Processor selbst installiert.

*Name* - der Name des Print Processors, mit dem er installiert wird.

```
UninstallDriverProc([in] BSTR Name);
```

Die Methode zur Deinstallation des Print Processors. Nach der Deinstallation muss Windows neu gestartet werden.

*Name* - der Name des Print Processors.

```
InstallPrinter([in] BSTR Name, [in] BSTR PrinterID, [in] BSTR DriverName, [in] BSTR PrintProcessorName, [in] BSTR PortName, [in] BSTR ShareName, [in] BSTR LicensePath);
```

Die Methode zur Installation des [virtuellen Druckers](#).

*Name* - der Druckername.

*PrinterID* - die Drucker-ID. Sie muss mit der Drucker-ID in der Lizenz-Datei identisch sein.

*DriverName* - Name des **Print to Document-Treibers**, der mittels der Methode "InstallDriver" installiert wird.

*PrintProcessorName* - der Name des Print Processor-Treibers, der mittels der Methode "InstallDriverProc" installiert wird.

*PortName* - der Name des Ports, der vom virtuellen Drucker verwendet wird. In der Regel ist das "LPT1:".

*ShareName* - der Freigabename des Druckers, falls der Name hingewiesen, wird der Drucker bei der Installation sofort freigegeben und ist im Netzwerk verfügbar. Falls der Wert gleich NULL ist, wird der Drucker nicht freigegeben. Um den Drucker im Netzwerk verwenden zu können, müssen die entsprechenden Rechte in der Lizenzdatei hinterlegt sein.

*LicensePath* - absoluter Pfad zur Lizenzdatei.

```
UninstallPrinter([in] BSTR Name);
```

Die Methode zur Deinstallation des virtuellen Druckers.

*Name* - der Druckername.

```
InstallPrinterDependence([in] BSTR PrinterName, [in] BSTR SrcPath, [in] BSTR DstFileName);
```

So werden die Dateien registriert, die für die Arbeit des Druckers als Netzwerk- Drucker erforderlich sind. Wenn der Netzwerkdrucker hinzugefügt wird, überträgt Windows die mit dieser Methode registrierten Dateien von dem Computer mit dem freigegebenen Drucker auf die Computer, auf denen der Netzwerk-Drucker hinzugefügt werden soll. Mindestens eine der Dateien ist das Modul, das das Interface [IP2DEvents70](#) implementiert. Bei Bedarf kann man so einige

Dateien registrieren, die zum Beispiel die Einstellungen oder die Ressourcen für die Implementierung des Interface enthalten. Diese Dateien werden in einen speziellen Systemordner kopiert.

*PrinterName* - der Druckername.

*SrcPath* - absoluter Pfad zu der zu registrierenden Datei.

*DstFileName* - Dateiname, der für das Kopieren in das Systemverzeichnis und für das Übertragen an den anderen Computer verwendet wird.

```
UninstallPrinterDependences([in] BSTR PrinterName);
```

Löscht alle registrierten Dateien.

*PrinterName* - der Druckername.

```
BSTR GetPrinterRegKey([in] BSTR PrinterName);
```

Ruft den Namen des [Registrierungsschlüssels](#) in HKEY\_LOCAL\_MACHINE ab. Der Schlüssel enthält die Einstellungen und Eigenschaften des Druckers.

*PrinterName* - der Druckername.

```
BSTR GetPrinterRegKeyByID([in] BSTR PrinterID);
```

Ruft den Namen des [Registrierungsschlüssels](#) in HKEY\_LOCAL\_MACHINE ab. Der Schlüssel enthält die Einstellungen und Eigenschaften des Druckers.

*PrinterID* - Drucker-ID.

```
BSTR GetPrinterRegUserKey([in] BSTR PrinterID);
```

Ruft den Namen des [Registrierungsschlüssels](#) in HKEY\_CURRENT\_USER ab. Der Schlüssel enthält die Einstellungen und Eigenschaften des Druckers.

*PrinterID* - Drucker-ID.

```
SetupPrinterEventHandler([in] BSTR PrinterName, [in] BSTR CoClassGUID32,  
[in] BSTR FileName32, [in] BSTR CoClassGUID64, [in] BSTR FileName64, [in]  
unsigned long EventMask);
```

Konfiguriert den [virtuellen Drucker](#) für die Verwendung der Komponente, die das Interface [IP2DEvents70](#) implementiert. Die Hauptaufgabe des Interface ist die Konfiguration der produktorientierten Druckereigenschaften.

*PrinterName* - der Druckername.

*CoClassGUID32* - GUID aus der Klasse, die das Interface [IP2DEvents70](#) implementiert. Das Modul, das diese Klasse enthält, muss auf der 32-Bit-Plattform kompiliert werden.

*FileName32* - der Name der Datei, wenn für die 32-Bit-Plattform kompiliert wird. Der Wert für den Namen kann für Drucker, die nicht in einem Netzwerk eingesetzt werden, gleich NULL sein. Für Netzwerk-Drucker muss die Datei mit diesem Namen mit der Methode "InstallPrinterDependence" registriert werden.

*CoClassGUID64* - GUID derselben Klasse für die 64-Bit-Plattform. Sie kann dieselbe sein wie die für GUID32.

*FileName64* - der Name der Datei, wenn für die 64-Bit-Plattform kompiliert wird. Der Wert für den Namen kann für Drucker, die nicht in einem Netzwerk eingesetzt werden, gleich NULL sein. Für Netzwerk-Drucker muss die Datei mit diesem Namen mit der Methode "InstallPrinterDependence" registriert werden.

*EventMask* - die Maske der bearbeiteten Ereignisse. Mehr Informationen finden sich in der Beschreibung des Interface [IP2DEvents70](#).

```
InstallConnector([in] BSTR PrinterName, [in] unsigned long ConnType, [in] BSTR CoClassGUID, [in] BSTR HostPath, [in] boolean Host64);
```

Mit dieser Methode wird die [Konnektor](#)-Komponente der [virtuellen Drucker](#) für die Bearbeitung der vom Drucker erstellten Druckaufträgen registriert. Bei Bedarf wird außerdem die Host-Applikation installiert.

*PrinterName* - der Druckername.

*ConnType* - für die Kommunikation von [Print Processor](#) und [Konnektor](#)-Komponente. Mehr Informationen finden sich in der Beschreibung des Interface [IP2DConnector70](#).

*CoClassGUID* - GUID der Klasse die das Interface [IP2DHSJobProcessor70](#) implementiert, wenn das Attribut ConnType = **P2D\_CONN\_DIRECT**. Anderenfalls wird das Interface [IP2DConnector70](#) implementiert.

*HostPath* - absoluter Pfad der Host-Applikation. Im Fall von `ConnType = P2D_CONN_DIRECT` muss dieser Wert gleich NULL sein.

*Host64* - falls der Wert TRUE ist, wird die Host-Applikation für die 64-Bit-Plattform installiert. Anderenfalls wird die 32-Bit-Version, die auf der 32-Bit- und 64-Bit-Plattform verwendet werden kann.

```
UninstallConnector([in] BSTR PrinterName);
```

Deinstalliert die [Konnektor](#)-Komponente.

*PrinterName* - Druckername.

### Grundlegende Regeln und Informationen

Nachfolgend werden einige wichtige Regeln für die Arbeit mit dem **Print to Document**-SDK und mit der [Setup-Komponente](#) erläutert:

1. Die Trial-Lizenz, die im SDK enthalten ist, ermöglicht es [virtuelle Drucker](#) zu installieren und mit einer Client-Software zu Testzwecken auf Entwickler-Computern zu integrieren. Es ist nicht erlaubt, die Trial-Lizenz für andere Zwecke zu verwenden, auch nicht für die Präsentation des virtuellen Druckers in einer Client-Software.
2. Die **Print to Document**-Technologie wird von Softwareentwicklern in verschiedenen Produkten eingesetzt. Um Konflikte zwischen diesen Produkten zu vermeiden, dürfen die Werte der Konstanten (die GUIDs und die Namen), die in der Datei "trial.h" enthalten sind und in der Installationsprozedur des Trial-Druckers verwendet werden, nicht in echten Projekten verwendet werden. Bei der Integration mit der eigenen Software müssen diese Werte durch andere, eigene ersetzt werden, damit sie eindeutig bleiben.

Die GUIDs der Klassen, die die Interfaces [IP2DEvents70](#) und [IP2DConnector70](#) implementieren, müssen ebenfalls durch eindeutige GUIDs ersetzt werden. Die zu ersetzende Werte aus dem Beispiel in "sx\_p2d\_client" sind:

```
{7194F58E-699F-4C84-9018-8BDB245C2A30}  
{8D6C6CDF-5E47-42DE-9F6B-5B288AD87534}
```

3. Die Trial Lizenz erlaubt die Integration des Druckers und des [Konnektors](#) in den Modi **P2D\_CONN\_DIRECT** und **P2D\_CONN\_PASSIVE**.



4. Bei der Installation des Druckers auf einem Computer mit Windows 2000 oder XP (bis einschließlich SP1) dürfen nur die 32-Bit-[Treiber](#) installiert werden. Ab Windows XP SP2 kann man auch die 64-Bit-[Treiber](#) installieren. Dementsprechend darf der [virtuelle Drucker](#) als Netzwerkdrucker auf einem 64-Bit-Windows nur dann verwendet werden, wenn der freigegebene virtuelle lokale Drucker sich auf einem Computer mit Windows XP SP2 oder einer späteren Version befindet.

## KONFIGURATION UND DRUCKER-EREIGNISSE

Nach der Installation der **Print to Document**-Komponenten und des [virtuellen Druckers](#) ist dieser sofort einsatzbereit. Seine diverse Einstellungen sind optional, das heißt es ist nicht erforderlich, die Einstellungen vor dem Einsatz des virtuellen Druckers zu verändern. Der folgende Abschnitt enthält detaillierte technische Informationen über die Einzelheiten der Drucker- und Treiberkonfiguration, so dass auch der unerfahrene Anwender leicht die gewünschten Einstellungen vornehmen kann.

### Was stellt ein Drucker je nach Sichtweise dar?

Endanwender betrachten den Drucker als das Gerät (die Hardware), das am Arbeitsplatz steht und Papier bedruckt. Für jedes dieser Geräte ist in Windows ein Symbol im Ordner "Drucker und Faxgeräte" eingerichtet. Im Drucken-Dialog praktisch jeder Software kann es ausgewählt und zum Drucken verwendet werden.

Aus der Sicht von Windows ist der Drucker ein Systemobjekt, das gesondert im System registriert ist und zu dem einige [Treiber](#) gehören, die seine Arbeit gewährleisten und in das grafische System des Betriebssystems integriert sind. Die Registrierung des Druckers, seine Konfiguration, der Druckvorgang, die Formate und die Kommunikation zwischen dem grafischen Subsystem von Windows und dem [Druckertreiber](#) folgen exakten Regeln.

Für Anwendungsprogramme und deren Entwickler ist der Drucker ein spezifisches Objekt des grafischen Subsystems (GDI) von Windows. Das GDI generiert auf Basis dieses Objekts die wichtigsten Objekte des Subsystems: Device Context (DC) oder Graphics in GDI+. Das Programm beziehungsweise der Entwickler können so das Drucker-Objekt und DC-Objekte verwenden.

Die Steuerung erfolgt ausschließlich über die Programmschnittstelle des grafischen Subsystems. Diese Schnittstelle wird von Windows richtig dokumentiert. Grundsätzlich besteht in Windows die Möglichkeit, den

Druckertreiber direkt aufzurufen. Die **Print to Document**-Technologie verwendet diese Möglichkeit allerdings nicht: zum einen widerspricht sie dem Kernkonzept der Technologie, nämlich der Datenerzeugung in einem [universellen Format](#). Zweitens kann diese Methode nur dann verwendet werden, wenn das Programm den Drucker (Druckertreiber) "kennt" beziehungsweise wenn beide praktisch von demselben Entwickler stammen.

Mit Blick auf die unterschiedlichen Vorstellungen über das Drucken aus der Perspektive von Windows einerseits und einer Anwendungssoftware andererseits ergeben sich wichtige Folgen. Der Anwendung kann das Drucker-Objekt nur durch das grafische Subsystem zur Verfügung gestellt werden. Beim ersten Aufruf eines verfügbaren Druckers durch eine Applikation wird ein solches Objekt durch das GDI im Anwendungs-Kontext erstellt. Das Druckerobjekt übernimmt die Eigenschaften des Drucker-Systemobjekts. Das Drucker-Systemobjekt ist in Windows registriert (Registry, Systemteil). Die Folgen dieses Ansatzes sind:

- Einige Druckereigenschaften können nur bei Systemobjekt konfiguriert werden. Dies kann nur über die Registry erfolgen. Die Änderungen werden in den Druckereigenschaften für die Anwendungen wirksam, die nach den Änderungen gestartet wurden. Auch für die Anwendungen, die bereits gestartet wurden, aber noch nicht den Drucker verwenden, werden die Änderungen wirksam.
- Die Konfiguration des Druckers über die Programmschnittstelle des grafischen Subsystems ändert nur die Eigenschaften von dynamischen Drucker-Objekten im Kontext gegebener Anwendung. Sie hat keinen Einfluss auf die Konfiguration von Objekten denselben Drucker in anderen Prozessen.

**Print to Document** erweitert die Palette der konfigurierbaren, von Windows oder hardwaremäßig vorgegeben Eigenschaften eines Druckers. Zum Beispiel kann die Papiergröße, die hardwareabhängig in der Regel auf A4 und/oder A3 beschränkt ist, bei einem [virtuellen Drucker](#) frei definiert werden. Oder die (nicht bedruckbaren) Seitenränder, die bei einem normalen Drucker immer fest vorgegeben sind, lassen sich bei virtuellen Druckern ebenfalls frei konfigurieren.

Weitere Vorteile von **Print to Document** bei der Verarbeitung von Druckaufträgen sind: die Möglichkeit der Bearbeitung von Schlüsselereignissen während des Druckens, die Umsetzung von spezifischen, produktorientierten Eigenschaften, die dem Endanwender zugänglich sein sollen, sowie die Bereitstellung von zusätzlichen, hilfreichen technischen Informationen. Diese Vorteile können genutzt werden, wenn in der Software, die den [virtuellen Drucker](#) enthalten soll, das COM-Interface [IP2DEvents70](#) eingesetzt wird.

## Konfiguration des Druckers in der Windows-Registry

Die Konfiguration eines Drucker-Systemobjekts wird in der Windows-Registry abgelegt, der Name des Schlüssels wird von Windows vergeben. Dieser Name ist von der Windows-Version, vom Druckertyp (lokaler oder Netzwerkdrucker) und vom Druckernamen abhängig. Aus diesem Grund wird empfohlen, den Namen des Schlüssels zu verwenden, der von den Methoden der [Setupkomponenten](#) von **Print to Document** vergeben wird.

Der Registry-Schlüssel eines Druckers befindet sich in dem Zweig HKEY\_LOCAL\_MACHINE. Aus diesem Grund ist auf diesen Schlüssel aus Anwendungen mit eingeschränkten Rechten nur ein Lese-Zugriff möglich. Die Konfiguration des Drucker-Systemobjekts ist aus solchen Anwendungen heraus nicht veränderbar.

Die **Print to Document**-Technologie vermag dieses Problem zu lösen. Bei der Erstellung der dynamischen Druckerobjekt-Exemplare in einer Anwendung (beim Aufruf durch das grafische Windows-Subsystem) versuchen die [Druckertreiber](#), die Drucker-Eigenschaften und Parameter aus dem Registry-Schlüssel in HKEY\_CURRENT\_USER zu lesen. Falls entsprechende Werte vorhanden sind, ersetzen die gelesenen Werte jene aus dem Systemobjekt. Da der Zugriff auf HKEY\_CURRENT\_USER üblicherweise nicht eingeschränkt ist, können die Anwendungen den Drucker in diesem Fall auch im Kontext von Benutzern mit eingeschränkten Rechten konfigurieren.

Um die Arbeit in diesem Modus zu ermöglichen, muss bei der Installation des Druckers eine spezielle Eigenschaft hinzugefügt werden (siehe UserKey), anderenfalls können Benutzer mit eingeschränkten Rechten die Konfiguration des Druckers nicht ändern.

Dieses Vorgehen erlaubt, unterschiedliche Konfigurationseinstellungen des Druckers für verschiedene Benutzer einzurichten. Diese betreffen Parameter, die sich in Unterschlüsseln des Hauptschlüssels befinden ("Papers", "Events", "Resolutions", "Connector").

Den vollständigen Namen des Schlüssels ruft man durch die Methoden der [Setup-Komponenten](#) auf: GetPrinterRegKey, GetPrinterRegKeyByID, GetPrinterRegUserKey, sowie die Methoden des zusätzlichen Interface [IP2DDialog70](#) bei der Darstellung des Dialogs für die Druckereigenschaften.

## Struktur und Werte des Registry-Schlüssels

### PrinterID (Wert)

Die Drucker-ID wird während der Installation vergeben. Sie darf während der Verwendung des Druckers nicht geändert werden. Wenn der gesetzte Parameter dem in der Lizenzdatei nicht entspricht, wird der Drucker blockiert.

### UserKey (Wert)

Der Parameter erlaubt oder unterbindet die Konfiguration des Druckers, unter anderem für Benutzer mit eingeschränkten Rechten. Es wird dabei der Registry-Schlüssel verwendet, der in HKEY\_CURRENT\_USER eingetragen ist.

Der Parametertyp ist DWORD, der Vorgabewert lautet "0". Ein Wert ungleich "0" erlaubt die Konfiguration über HKEY\_CURRENT\_USER.

### Shareable (Wert)

Der Parameter erlaubt oder unterbindet die Verwendung eines Druckers als Netzwerkdrucker. Windows bietet keine Möglichkeit, die Netzwerkinstallation eines Druckers nicht zu erlauben, falls der lokale Drucker freigegeben ist. Aus diesem Grund blockiert dieser Parameter nicht die eigentliche Installation, sondern den Druckvorgang durch den vorgegebenen Drucker.

Der Parametertyp ist DWORD, der Vorgabewert lautet "0". Ein Wert ungleich "0" erlaubt die Verwendung des Druckers als Netzwerkdrucker.

Wenn bei der Installation des Druckers ein Freigabename festgelegt wurde, erhält der Parameter den Wert "1". Für die Verwendung des Druckers als Netzwerkdrucker müssen spezielle Rechte in der Lizenzdatei eingetragen sein.

### Color (Wert)

Dieser Parameter definiert den Druckertyp - farbig oder schwarz-weiß. Im Falle eines Schwarz-Weiß-Druckers wird dennoch keine automatische Umwandlung durchgeführt: die gedruckten Daten bleiben in der Regel farbig. Manche Programme können dies allerdings berücksichtigen und ihre Daten entsprechend ausdrucken.

Der Parametertyp ist DWORD, der Vorgabewert lautet "0". Ein Wert ungleich "0" steht für einen Farbdrucker.

### ResetDC (Wert)

Der Parameter dient der Lösung eines Problems, das infolge der Besonderheiten des System-Spoolers (Systemdienst für das Drucken) und

manchen Programme, zum Beispiel Notepad, auftritt. Beim Drucken aus diesem Programm fordert der Spooler vom [Treiber](#) die *DevMode*-Struktur mit der aktuellen Druckerkonfiguration nicht an, und assoziiert eine Struktur mit einem nicht aktuellen Stand mit dem Druckauftrag.

Der Parametertyp ist *DWORD*, der Vorgabewert lautet "0". Ein Wert ungleich "0" erlaubt dem Treiber bei Erstellung des Drucker-DC automatisch die Funktion *ResetDC()* aufzurufen. Diese veranlasst den Spooler, den aktuellen Stand des *DevMode* anzufordern.

### **HorizontalResolution (Wert)**

Der Parameter stellt die horizontale Auflösung in DPI dar. Der Parametertyp ist *DWORD*, der Vorgabewert lautet "300".

### **VerticalResolution (Wert)**

Der Parameter stellt die vertikale Auflösung in DPI dar. Der Parametertyp ist *DWORD*, der Vorgabewert lautet "300".

### **PaperID (Wert)**

Der Parameter definiert den Papiertyp (Größe/Format eines Blatts), der von einem Drucker verwendet wird. Die Werte 0 bis 255 sind von Windows für Standardpapier reserviert. Zusätzliche Blattmaße und Papier-IDs können in einem Unterschlüssel "Papers" vorgegeben werden. Papier mit der ID "256" wird in den Druckerdialogen als "Benutzerdefiniert" dargestellt. Ein Druckprogramm muss bei der Einstellung "Benutzerdefiniert" die angegebene Papiergröße zuweisen.

Der Parametertyp ist *DWORD*, der Vorgabewert lautet "256".

### **PaperWidth (Wert)**

Die Breite eines Blatts, wenn die ID "256" ist (Benutzerdefiniert). Die Maßeinheit ist 0,1 mm.

Der Parametertyp ist *DWORD*, der Vorgabewert lautet "2100".

### **PaperHeight (Wert)**

Die Höhe eines Blatts, wenn die ID "256" ist (Benutzerdefiniert). Die Maßeinheit ist 0,1 mm.

Der Parametertyp ist *DWORD*, der Vorgabewert lautet "2970".

### **Landscape (Wert)**

Die Ausrichtung eines Blatts.

Der Parametertyp ist *DWORD*, der Vorgabewert lautet "0". Ein Wert ungleich "0" steht für Querformat.

**PaperSource (Wert)**

Der Parameter dient der Lokalisierung entsprechender Druckereigenschaft "Quelle".

Der Parametertyp ist eine Zeichenfolge, der Vorgabewert ist "Auto source".

**LeftMargin (Wert)**

Der linke Rand (nicht bedruckbarer Bereich) beim Drucken. Die Maßeinheit ist 0,1 mm.

Der Parametertyp ist DWORD, der Vorgabewert lautet "0".

**RightMargin (Wert)**

Der rechte Rand (nicht bedruckbarer Bereich) beim Drucken. Die Maßeinheit ist 0,1 mm.

Der Parametertyp ist DWORD, der Vorgabewert lautet "0".

**TopMargin (Wert)**

Der obere Rand (nicht druckbarer Bereich) beim Drucken. Die Maßeinheit ist 0,1 mm.

Der Parametertyp ist DWORD, der Vorgabewert lautet "0".

**BottomMargin (Wert)**

Der untere Rand (nicht druckbarer Bereich) beim Drucken. Die Maßeinheit ist 0,1 mm.

Der Parametertyp ist DWORD, der Vorgabewert lautet "0".

**ClientData (Wert)**

Der Parameter enthält die produktspezifischen Daten und Einstellungen. Ein Beispiel: Für einen Drucker, der PDF-Dateien erstellt, können diese Daten die erforderlichen PDF-Eigenschaften sein, die vom Endanwender im Drucker-Konfigurationsdialog vorgegeben werden. Beim Ausdrucken (Erstellen der PDF-Datei) fügt der [Treiber](#) den aktuellen Wert des Parameters jedem Druckauftrag hinzu. Die Software, die den Druckauftrag verarbeitet, verwendet dann den Wert des Parameters. Mehr dazu siehe unter [Job Info \(Client-Daten\)](#).

Der Parametertyp ist ein Binärwert, die maximale Größe beträgt 4096 Bytes.

**Papers (Unterschlüssel)**

Dieser Schlüssel kann wiederum mehrere Unterschlüssel enthalten. Jeder dieser Unterschlüssel definiert den produktspezifischen Papiertyp, den der Endanwender beim Ausdrucken auf den [virtuellen Drucker](#) definieren kann. Der Name jedes Unterschlüssels repräsentiert die Papierbezeichnung in den Druckerdialogen. Beispielweise wird mit dem Unterschlüssel "Papers\A0" der Papierliste das Format "A0" hinzugefügt.

Jeder Unterschlüssel muss drei Parameter enthalten: PaperID, PaperWidth, PaperHeight. Die PaperID muss ungleich "0" sein. Die Werte von 1 bis 256 dürfen nur für die Lokalisierung der Standardpapiere und des benutzerdefinierten Papiers verwendet werden. Wenn die PaperID größer als 255 ist, sind die Parameter "PaperWidth" und "PaperHeight" erforderlich, andernfalls werden sie ignoriert. Maßeinheit = 0.1 mm.

### **Resolutions (Unterschlüssel)**

Dieser Schlüssel kann mehrere Werte enthalten, von denen jeder die unterstützte Druckauflösung (DPI) bestimmt. Das druckende Programm wählt aus der Liste der unterstützten Auflösungen die für eine bestimmte Aufgabe passende aus. Manche Programme erlauben auch dem Benutzer die Auswahl. Der Name der Werte ist nicht entscheidend, nur der eigentliche Wert selbst wird verwendet.

### **Events (Unterschlüssel)**

Dieser Schlüssel dient der Konfiguration des [Druckertreibers](#). Er enthält die Parameter, die für die Integration mit der Software verwendet werden können, die einen [virtuellen Drucker](#) einsetzt. Eine solche Integration ist seitens des virtuellen Druckers aber nicht erforderlich. Aus diesem Grund wird für den Schlüssel nur bei Aufruf der Methode "SetupPrinterEventHandler" von der [Setup-Komponente](#) - im Rahmen der Installation - ein Wert vergeben.

**CoClassGUID32** - GUID für die Klasse, die das Interface [IP2DEvents70](#) implementiert. Das Modul, das diese Klasse enthält, muss auf der 32-Bit-Plattform kompiliert werden.

**FileName32** - Dateiname für das 32-Bit-Modul.

**CoClassGUID64** - GUID der entsprechenden Klasse für die 64-Bit-Plattform. Die GUID kann mit der GUID für die 32-Bit-Klasse identisch sein.

**FileName64** - Dateiname für das 64-Bit-Modul.

**EventMask** - die Maske der zu überwachenden Drucker-Ereignisse. Mehr dazu siehe in der Beschreibung des Interface [IP2DEvents70](#).



**Connector (Unterschlüssel)**

Dieser Schlüssel dient der Konfiguration des [Print Processors](#) des Druckers. Er enthält die Parameter, die für die Integration mit der Software erforderlich sind, die einen [virtuellen Drucker](#) einsetzt. Eine solche Integration ist erforderlich, da die Daten, die in einem virtuellen Drucker verarbeitet werden, durch diese in einem [universellen Format](#) übergeben werden. Für den Schlüssel wird bei Aufruf der Methode "InstallPrinterConnector" von der [Setup-Komponente](#) - im Rahmen der Drucker-Installation - ein Wert vergeben.

**ConnType** - der Schlüssel für die Kommunikation von Print Processor und der [Konnektor](#)-Komponente. Mehr dazu siehe in der Beschreibung des Interface [IP2DConnector70](#).

**CoClassGUID** - GUID für die Klasse, die das Interface [IP2DHSJobProcessor70](#) implementiert, wenn **ConnType** = P2D\_CONN\_DIRECT ist. Andernfalls wird das Interface [IP2DConnector70](#) verwendet.

**HostPath** - der absolute Pfad des Host-Programms.

**Host64** - wenn der Wert TRUE ist, wird das Host-Programm für die 64-Bit-Plattform installiert, andernfalls für die 32-Bit-Version. Letztere kann sowohl auf 32-Bit als auch auf der 64-Bit-Plattform eingesetzt werden.

**Threads** - die Anzahl der Threads im Host-Programm. Wenn der Wert fehlt oder gleich "0" ist, wird eine optimale Anzahl für den konkreten Computer errechnet. Der Parameter wird nicht verwendet, wenn **ConnType** = P2D\_CONN\_DIRECT ist.

**JobPath** - der absolute Pfad zu dem Ordner, in den Druckaufträge temporär abgelegt werden, wenn **ConnType** = P2D\_CONN\_PASSIVE.

**Die Ereignisse im Treiber und Interface IP2DEvents70**

Dieses COM-Interface nutzt der Drucker-[Konfigurationstreiber](#) für die Kommunikation mit einer Software, die einen [virtuellen Drucker](#) verwendet. Die Software kann dieses Interface implementieren, muss es aber nicht. Wenn das Interface implementiert wird, ist sein Modul standardmäßig für das COM-Interface zu registrieren. Außerdem wird es beim virtuellen Drucker mit der Methode "SetupPrinterEventHandler" registriert.

Für Netzwerk-Drucker muss das Modul beim virtuellen Drucker mit der Methode "InstallPrinterDependence" registriert werden. Falls das Modul



weitere Module, Komponenten oder Ressourcendateien benötigt, ist jede dieser Dateien mit derselben Methode zu registrieren. Das Modul und eventuelle weitere Module darf keine spezielle Installation und keine spezifische Systemkomponenten erfordern. Wenn solche Erfordernisse bestehen, würden sie auf anderen Computern, auf denen der virtuelle Drucker als Netzwerkdrucker hinzugefügt wird, nicht funktionieren.

Der [Konfigurationstreiber](#) gewährleistet beim Hinzufügen eines Netzwerkdruckers die COM-Registrierung des Moduls, das das Interface [IP2DEvents70](#) auf demselben Computer implementiert.

Um den [virtuellen Drucker](#) auf Computern sowohl mit 32-Bit-Architektur als auch mit 64-Bit-Architektur einsetzen zu können, müssen zwei Module kompiliert werden – für jede Architektur ein eigenes. **CoClass**, das das Interface in die Module für die 32-Bit- und 64-Bit-Plattformen implementiert, kann unterschiedliche oder gleiche GUIDs verwenden.

Bei der Erstellung des Interface muss beachtet werden, dass das Objekt, das dieses Interface realisiert, vom Drucker-Konfigurationstreiber verwendet wird. Der [Treiber](#) wird von einer beliebigen Applikation, in der ein virtueller Drucker zum Einsatz kommt, geladen.

Dies kann der PrintSpooler sein, der im Kontext des Systembenutzers arbeitet, ein anderer Systemdienst oder eine beliebige Applikation, die in einem Kontext eines beliebigen Benutzerkontos arbeitet. Dieses Benutzerkonto kann auch ein nicht aktives Konto sein oder ein Konto mit eingeschränkten Benutzerrechten.

Außerdem ist zu beachten, dass die Lebensdauer des Objekts, das das Interface [IP2DEvents](#) implementiert, vom Treiber nur für die Bearbeitung eines einzigen Ereignisses verwendet wird. In der Regel wird das Objekt nach der Bearbeitung gelöscht. Es wird nicht empfohlen, irgendwelche Daten in diesem Objekt aufzubewahren.

Nach der Registrierung des Interface beim virtuellen Drucker kann eine Maske (Auswahl aus Ereignistypen) definiert werden. Entsprechend dieser Maske informiert der Treiber das Objekt. Die Ereignisse, die nicht in der Implementierung des Interface verwendet werden, werden so ausgeschlossen.

### Die Ereignisse im P2D-Drucker

**P2D\_EVENT\_PRN\_PROP** - Es wird der Eigenschaften-Dialog des [virtuellen Druckers](#) geöffnet. Hier können im Dialog eine oder mehrere Seiten mit speziellen Eigenschaften des virtuellen Druckers angezeigt werden.

**P2D\_EVENT\_DOC\_PROP** - Es wird der Eigenschaften-Dialog des virtuellen Druckers geöffnet. Hier können im Dialog eine oder mehrere Seiten mit speziellen Eigenschaften der auszudruckenden Dokumente (zum Beispiel die PDF-Eigenschaften für einen PDF-Drucker) angezeigt werden.

**P2D\_EVENT\_PRN\_NET\_ADD** - Ein Netzwerkdrucker wird hinzugefügt (installiert). In der Regel fehlt auf diesem Computer die Software, die den virtuellen Drucker zur Verfügung stellt. Wenn eine Initialisierung für Ihr Modul erforderlich ist, darf das Modul diese mit den Ereignis ausführen.

**P2D\_EVENT\_PRN\_NET\_DEL** - Ein Netzwerkdrucker wird entfernt (deinstalliert). Temporäre Ressourcen und Daten müssen nun ebenfalls entfernt werden.

**P2D\_EVENT\_DOC\_CREATEDCPRE** - eine Applikation hat den Drucker-Device Context vom grafischen Subsystem angefordert und zu diesem Zeitpunkt wurde das Objekt noch nicht erstellt.

**P2D\_EVENT\_DOC\_CREATEDCPOST** - eine Applikation hat den Drucker-Device Context vom grafischen Subsystem angefordert und zu diesem Zeitpunkt wurde das DC-Objekt bereits erstellt.

**P2D\_EVENT\_DOC\_STARTDOC** - eine Applikation hat die Funktion "StartDoc" aufgerufen, der PrintSpooler erstellt den Druckauftrag.

**P2D\_EVENT\_DOC\_ENDDOC** - eine Applikation hat die Funktion "EndDoc" aufgerufen, der PrintSpooler hat den Druckauftrag bereits erstellt und sendet ihn an den [Print Processor](#).

**P2D\_EVENT\_DOC\_DELETEDC** - eine Applikation löscht das DC-Objekt.

### Interface I2PEvents70

```
HRESULT PrinterProperties([in] BSTR sPrinterName, [in] IUnknown* pDialog);
```

Die Methode wird vom [Treiber](#) für die Benachrichtigung über das Ereignis P2D\_EVENT\_PRN\_PROP aufgerufen.

*sPrinterName* - Druckername.

*pDialog* - Das Objekt implementiert das Interface [IP2DDialog70](#). Es wird für das Hinzufügen eigener Eigenschaften-Seiten im Dialog für die Druckereigenschaften verwendet.

```
HRESULT DocumentProperties([in] BSTR sPrinterName, [in] IUnknown*  
pDialog);
```

Die Methode wird vom Treiber für die Benachrichtigung über das Ereignis P2D\_EVENT\_DOC\_PROP aufgerufen.

*sPrinterName* - Druckername.

*pDialog* - Das Objekt implementiert das Interface [IP2DDialog70](#). Es wird für das Hinzufügen eigener Eigenschaften-Seiten im Dialog für die Eigenschaften der auszudruckenden Dokumente verwendet.

```
HRESULT DocumentEvent([in] BSTR sPrinterName, [in] UINT_PTR hPrinter, [in]  
LONG Event);
```

Die Methode wird vom Treiber für die Benachrichtigung über die folgenden Ereignisse aufgerufen:

P2D\_EVENT\_DOC\_CREATEDCPRE, P2D\_EVENT\_DOC\_CREATEDCPOST,  
P2D\_EVENT\_DOC\_STARTDOC, P2D\_EVENT\_DOC\_ENDDOC,  
P2D\_EVENT\_DOC\_DELETEDC

*sPrinterName* - Druckername.

*hPrinter* - System-Handle eines Druckers.

*Event* - Ereignistyp.

```
HRESULT PrinterEvent([in] BSTR sPrinterName, [in] LONG Event);
```

Die Methode wird vom Treiber für die Benachrichtigung über die Ereignisse P2D\_EVENT\_PRN\_NET\_ADD, P2D\_EVENT\_PRN\_NET\_DEL aufgerufen.

*sPrinterName* - Druckername.

*Event* - Ereignistyp.

### **Interfaces I2PDiallog70**

```
BSTR GetPrinterKey();
```

Ruft den Namen des [Registry-Schlüssels](#), der die Druckereigenschaften und Einstellungen enthält, in HKEY\_LOCAL\_MACHINE ab.

```
BSTR GetPrinterUserKey();
```

Ruft den Namen des [Registry-Schlüssels](#), der die Druckereigenschaften und Einstellungen enthält, in HKEY\_CURRENT\_USER ab.

```
AddPropertyPage([in] BSTR sTitle, [in] ULONG_PTR hInstance, [in] BSTR  
sDlgTemplate, [in] ULONG_PTR pDlgProc, [in] ULONG_PTR hPageKey);
```

Diese Methode fügt eine neue Seite in den Dialog für die Druckereigenschaften ein.

*sTitle* - Titel der Eigenschaften-Seite.

*hInstance* - Handle einer Prozess-Instanz für eine Dialogbox-Vorlage, ein Symbol oder eine Titelstring-Ressource.

*sDlgTemplate* - Dialogbox-Vorlage für die Erstellung der Seite. Dieses Element kann entweder den Ressource Identifier der Vorlage angeben oder die Adresse einer Zeichenfolge, die den Namen der Vorlage enthält.

*pDlgProc* – der Pointer zur Dialogbox-Prozedur für eine Seite. Die Dialogbox-Prozedur darf nicht die EndDialog-Funktion aufrufen.

*hPageKey* - ID eines Exemplars der hinzugefügten Seite. Die ID wird wie "lParam" bei der Meldung "WM\_INITDIALOG" an die Dialogbox-Prozedur übergeben.

### **Druckerkonfiguration in einer Applikation (Besonderheiten bei P2D-Druckern)**

Die Konfiguration eines Druckers über die Programmschnittstelle von Windows ändert nur die Eigenschaften des dynamischen Druckerobjekts im Kontext einer einzelnen Applikation. Sie hat keinen Einfluss auf die Konfiguration eines Systemobjekts oder dynamischer Objekte desselben Druckers in anderen Applikationen. Für die Konfiguration eines Druckers wird die Funktion "DocumentProperties" verwendet.

[Virtuelle Drucker](#), die mit **Print to Document** generiert werden, bieten in der druckenden Applikation breitere Möglichkeiten für die Drucker-Steuerung. Außer der Funktion "DocumentProperties" kann eine [andere Funktion](#) des grafischen Subsystems von Windows verwendet werden - "ExtEscape". Diese Funktion wird mit speziellen Codes aufgerufen, die von **Print to Document** unterstützt werden.

## Escape-Codes - Erweiterung der Print to Document-Technologie

**P2D\_ESCAPE\_RESET\_DRV\_DATA** - Der Aufruf von "ExtEscape" mit diesem Code meldet dem [Treiber](#), das dynamische Druckerobjekt neu zu konfigurieren, das heißt die Eigenschaften aus dem Systemobjekt zu übernehmen. Um die Eigenschaften auch für die zuvor erstellten DC-Objekte zu aktualisieren, muss bei diesen Objekten die Funktion "ResetDC" aufgerufen werden oder die Objekte müssen gelöscht und neu erstellt werden.

**P2D\_ESCAPE\_RESET PAPERS** - gibt dem Treiber den Impuls, die zu unterstützenden Papiertypen in der [Registry](#) zu aktualisieren.

**P2D\_ESCAPE\_RESET RESOLUTIONS** - gibt dem Treiber den Impuls, die zu unterstützenden Auflösungen in der [Registry](#) zu aktualisieren.

**P2D\_ESCAPE\_SET\_JOB\_ID** - gibt dem Treiber den Impuls, die String-ID mit dem Druckauftrag zu assoziieren. Details finden sich im Abschnitt [Job Info](#). Die String-ID wird über den Parameter "lpszInData" der Funktion "ExtEscape" übergeben. Die ID muss im Unicode-Format vorliegen und kürzer als 40 Zeichen sein. Vor dem Aufruf der Funktion "StartDoc" muss "ResetDC" aufgerufen werden, damit der PrintSpooler die aktuelle Druckerkonfiguration sowie die assoziierte ID sicher aktualisieren kann. Nach dem Aufruf von "EndDoc" wird empfohlen, die ID zu "nullen", das heißt einen NULL-String als ID zuzuweisen und die Funktion "ResetDC" aufzurufen. Andernfalls könnte der PrintSpooler den nicht aktuellen Wert aus dem Cache für nachfolgende Druckaufträge verwenden.

**P2D\_ESCAPE\_SET\_CLIENT\_DATA** - gibt dem Treiber den Impuls, die Zusammenstellung binärer Client-Daten im dynamischen Druckerobjekt zu aktualisieren (aktuelle produktspezifische Druckereinstellungen). Danach werden die Daten mit den Druckaufträgen assoziiert. Details finden sich im Abschnitt [Job Info](#). Die Prozedur erfolgt ähnlich wie bei P2D\_ESCAPE\_SET\_JOB\_ID.

## VERARBEITUNG VON DRUCKAUFTRÄGE

### Print Processor und Konnektor

Gedruckte Daten werden von dem speziellen Systemdienst "Print Spooler" in einem [universellen Format](#) (Spool EMF-Format) an den [Print Processor](#) (spezieller Treiber) übergeben. Die Daten (Spool-Daten) für einen Druckauftrag werden dabei in einem Aufruf, in einem Datenblock (Pseudo-Datei) weitergeleitet. Außer diesen Daten erhält der Print Processor vom Print Spooler einen Datenblock

(Druckauftragsinformation), der mit dem Druckauftrag verknüpft ist. Ein Teil der Daten in diesem Block wird vom PrintSpooler-Dienst erzeugt, der Dateninhalt und das Datenformat entsprechen der Struktur **JOB\_INFO\_2**, die im Windows Plattform SDK definiert wird. Der andere Teil der Daten wird vom [Konfigurationstreiber](#) erzeugt. Er wird während des Druckens aus einer beliebigen Software an den PrintSpooler-Dienst übergeben.

Für Netzwerkdrucker gewährleistet Windows die Übergabe der Druckaufträge an Computer mit einem installierten, freigegebenen lokalen Drucker. Auf diesen Computern werden die Druckaufträge an den [Print Processor-Treiber](#) übergeben, die entsprechende Bearbeitung von Druckaufträgen, die von Netzwerkdruckern geliefert werden, erfolgt nur zentral auf dem Server.

Der Print Processor im [virtuellen Drucker](#) bearbeitet die Druckaufträge nicht. Er dient ausschließlich der Integration mit der Software, die den virtuellen Drucker verwendet. Die Integration wird über das COM-Interface ausgeführt, das in **Print to Document** deklariert und in diese Software implementiert ist. Das Modul, das dieses Interface implementiert, wird in Folgenden als [Konnektor](#) bezeichnet.

**Print to Document** bietet dem Entwickler des [Konnektor](#)-Moduls einige Integrationsmethoden ([Konnektor-Typen](#)). Der Konnektor-Typ wird bei der Installation des Druckers festgelegt. In die [Methode](#) "InstallPrinterConnector" wird der Konnektor-Typ und die GUID aus der Klasse übergeben, die den Konnektor implementiert.

## Konnektor-Typen

### P2D\_CONN\_DIRECT

Dieser Typ gewährleistet eine sehr enge Integration mit dem [Print Processor](#), stellt dem Entwickler des [Konnektors](#) die Druckauftrag-Daten schnell zur Verfügung und erlaubt eine effektive Weiterverarbeitung. Wenn dieser Konnektor eingesetzt wird, kann der Entwickler auf einige nützliche Möglichkeiten der anderen Konnektor-Typen nicht zurückgreifen, zum Beispiel auf das Extrahieren des EMF-Blocks für die einzelnen Seiten aus den Druckauftrag-Daten. Das Host-Programm kann ebenfalls nicht verwendet werden.

Der Konnektor muss das COM-Interface [IP2DHSJobProcessor70](#) implementieren. Das Konnektor-Modul kann entweder als DLL oder als Programm realisiert werden. Es muss während der Druckerinstallation wie jedes andere COM-Modul registriert werden.

Folgendes ist zu beachten:

- Dieser Konnektor-Typ wird immer im Kontext des Systembenutzers aufgerufen
- Druckaufträge für einen [virtuellen Drucker](#) werden an den Konnektor nacheinander übergeben, das heißt: erst nachdem die Weiterverarbeitung eines Druckauftrags beendet ist, werden die folgenden Aufträge verarbeitet
- Der Konnektor darf keine Benutzerinteraktivität (Dialoge, Meldungen) mit dem Anwender beinhalten

### **P2D\_CONN\_HOST\_SYS**

Dieser Konnektor-Typ verwendet das Host-Programm. Der Entwickler profitiert damit von dessen Vorteilen. Das Host-Programm wird hier unabhängig davon, welcher Benutzer auf dem virtuellen Drucker druckt, im Systembenutzer-Kontext aufgerufen. Aus diesem Grund darf der Konnektor keine Interaktivität (Dialoge, Meldungen) mit dem Anwender beinhalten.

Das Konnektor-Modul muss das COM-Interface [IP2DConnector70](#) und kann entweder als DLL oder als Programm realisiert werden. Es muss während der Druckerinstallation wie jedes andere COM-Modul registriert werden.

### **P2D\_CONN\_HOST\_IAU**

Dieser Konnektor-Typ hat Ähnlichkeit mit P2D\_CONN\_HOST\_SYS. Das Host-Programm befindet sich hier aber im Kontext der ersten Benutzersitzung (First User Session), und zwar unabhängig davon, wer gedruckt hat. Der Konnektor darf Interaktivität (Dialoge, Meldungen) für die Kommunikation mit dem Anwender beinhalten.

Falls beim Drucken keine interaktive Sitzung auf dem Computer vorhanden ist, wird der Druckauftrag ohne Benachrichtigung des Konnektors gelöscht.

### **P2D\_CONN\_HOST\_JOU**

Dieser Konnektor-Typ hat Ähnlichkeit mit P2D\_CONN\_HOST\_IAU. Das Host-Programm befindet sich hier im Kontext des Benutzers, der ausgedruckt hat, sofern seine Sitzung in dem Moment der Druckauftragsbearbeitung aktiv ist. Für Druckaufträge, die im Kontext des Systembenutzers gedruckt wurden oder vom Netzwerkdrucker kommen, wird die erste Benutzersitzung verwendet.

### **Interface IP2DHSJobProcessor70**

Dieses COM-Interface wird vom [Print Processor](#) für die Kommunikation mit dem Host-Programm und mit dem [P2D\\_CONN\\_DIRECT-Konnektor](#) verwendet.

```
HRESULT ProcessJob([in] IP2DJob70* pJob);
```

Diese Methode wird vom Print Processor für die Benachrichtigung über die Weiterverarbeitung des nachfolgenden Druckauftrags aufgerufen.

*pJob* - das Objekt, das das Interface [IP2DJob70](#) implementiert und Informationen über den Drucker und die Druckauftrag-Daten bereitstellt.

### **Interface IP2DJob70**

Dieses COM-Interface stellt dem [Konnektor P2D\\_CONN\\_DIRECT](#) Informationen über den Drucker und die Druckauftrag-Daten zur Verfügung.

```
BSTR get_PrinterName();
```

Name des Druckers, über den der Druckauftrag angekommen ist.

```
BSTR get_DocumentName();
```

Name des Dokuments (Druckauftrag), der beim Drucken angegeben wurde.

```
ULONG get_JobID();
```

Die ID des Druckauftrags, die im Print Spooler zugewiesen wurde.

```
HRESULT get_JobInfo([in] IStream* pJobInfo);
```

Der Datenblock mit Detailinformationen über den Druckauftrag (siehe [Job Info](#)).

```
HRESULT get_SpoolData([in] IStream* pSpoolData);
```

Der Datenblock im [universellen Format](#) (Spool EMF-Format) für den vorliegenden Druckauftrag (siehe Dokument "Enhanced Metafile Spool Format Specification" im MSDN von Microsoft).

### **Interface IP2DConnector70**

Dieses COM-Interface wird für die Kommunikation zwischen dem Host-Programm und allen [Konnektortypen](#) außer P2D\_CONN\_DIRECT verwendet.



```
HRESULT ProcessJob([in] IP2DHSJob70* pJob);
```

Diese Methode wird vom [Print Processor](#) für die Benachrichtigung über die Weiterverarbeitung des nachfolgenden Druckauftrags aufgerufen.

*pJob* - das Objekt, das das Interface [IP2DHSJob70](#) implementiert und Informationen über den Drucker und die Druckauftrag-Daten bereitstellt.

```
HRESULT IsAlive();
```

Die Methode wird für die Status-Überprüfung des Objekts verwendet. Es bestehen keine funktionalen Anforderungen an diese Methode, außer dass sie implementiert sein muss.

### **Interface IP2DHSJob70**

Dieses COM-Interface stellt für alle [Konnektor-Typen](#) außer für P2D\_CONN\_DIRECT Informationen über den Drucker und die Druckauftrag-Daten zur Verfügung.

```
BSTR get_PrinterName();
```

Name des Druckers, über der Druckauftrag angekommen ist.

```
BSTR get_DocumentName();
```

Name des Dokuments (Druckauftrag), der beim Drucken angegeben wurde.

```
BSTR get_JobID();
```

Die ID des Druckauftrags, die von [Konfigurationstreiber](#) in **Print to Document** (oder von der druckenden Anwendung, siehe [Escape-Code](#) P2D\_ESCAPE\_SET\_JOB\_ID) vergeben wurde.

```
HRESULT get_JobInfo([in] IStream* pJobInfo);
```

Der Datenblock mit der Detailinformationen über den Druckauftrag (siehe [Job Info](#)).

```
ULONG get_PageCount();
```

Die Anzahl der Seiten im Druckauftrag.

```
IStream* get_PageData([in] ULONG PageIndex);
```

Der Datenblock im EMF-Format für die Seite mit dem angegebenen Index. Der Stream-Pointer kann NULL sein, wenn die EMF für die angegebenen Seite im Druckauftrag fehlt. Im Test-Modus des Druckers (mit Test-Lizenz) werden bei dieser Methode für per Zufall ausgewählte Seiten die tatsächlichen EMF-Daten durch Daten mit dem Text "Print to Document" ersetzt.

### Job Info

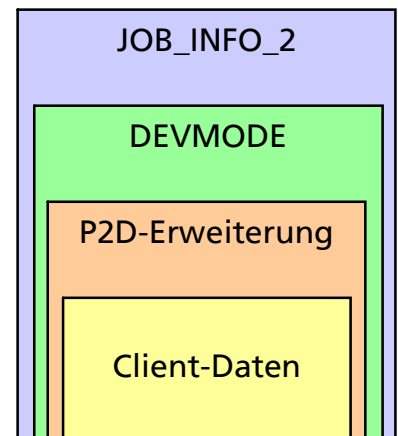
Das ist der Datenblock, der mit dem Druckauftrag assoziiert ist und detaillierte Information über diesen enthält. Ein Teil der Daten in diesem Block wird vom Print Spooler-Dienst erstellt. Das Datenformat entspricht der Struktur JOB\_INFO\_2, die vom Windows Plattform SDK definiert wird.

Da der Datenblock auch von einem Prozess zu einem anderen übergeben wird, sind alle Elemente der Datenstruktur, die als Pointer spezifiziert sind, nicht als Pointer im Speicher sondern als Verschiebungen in Bezug auf den Blockanfang anzusehen. Wenn der Wert eine NULL-Verschiebung darstellt, wird das Element ignoriert. Die Ersetzung des Pointers durch eine Verschiebung gewährleistet der **Print to Document-Treiber**. Die Strings im Block haben Unicode-Format.

Das Element "pDevMode" beinhaltet die Verschiebung vom Blockanfang auf die Struktur im Datenformat DEVMODE, die vom Windows Plattform SDK definiert wird.

Diese Struktur wird noch vor dem Ausdrucken durch den [Konfigurationstreiber](#) erstellt. Sie kann auch von der druckenden Applikation modifiziert werden.

In der Regel (wenn das DEVMODE-Feld "dmDriverExtra" ungleich Null ist) wird diese Struktur folgen Felder mit zusätzlichen Daten (P2D-Erweiterung). Diese Daten werden vom **Print to Document**-Konfigurationstreiber bereitgestellt. Für Druckaufträge, die von einem Netzwerkdrucker kommen, werden alle Daten für den Netzwerk-Computer relevant, auf dem der Netzwerkdrucker verwendet wird.



Die Verschiebung für das erste Feld der zusätzlichen Daten (P2D-Erweiterung) ist der Wert des Felds "dmSize" der Struktur DEVMODE. Diese Verschiebung ist relativ zum Anfang der Struktur DEVMODE.

### **Job Info - Erweiterung der Print to Document-Technologie**

**dwSign** - Signatur für Struktur mit zusätzlichen Daten = "-D2P", 4 Bytes.

**dwSize** - die Größe der Struktur DEVMODE mit zusätzlichen Daten, die von **Print to Document** definiert werden, 4 Bytes.

**dwFlags** - Felder, die Information über den Drucker und die Windows-Version bereitstellen, 4 Bytes.

**dwLeftMargin** - linker Rand (nicht bedruckbarer Bereich) des Druckers. Die Maßeinheit ist 0,1 mm, 4 Bytes.

**dwTopMargin** - oberer Rand (nicht bedruckbarer Bereich) des Druckers. Die Maßeinheit ist 0,1 mm, 4 Bytes.

**dwRightMargin** - rechter Rand (nicht bedruckbarer Bereich) des Druckers. Die Maßeinheit ist 0,1 mm, 4 Bytes.

**dwBottomMargin** - unterer Rand (nicht bedruckbarer Bereich) des Druckers. Die Maßeinheit ist 0,1 mm, 4 Bytes.

**dwProcessID** - die System-ID des Prozesses, der die Daten gedruckt hat, 4 Bytes.

**Time** – Start des Druckvorgangs. Größe von FILETIME.

**PrinterID** - die Drucker-ID von Print to Document, 40 Bytes.

**JobID** - die ID, die dem Druckauftrag von der druckenden Applikation oder dem [Konfigurationstreiber](#) zugewiesen wurde (siehe [Escape-Code P2D\\_ESCAPE\\_SET\\_JOB\\_ID](#)). Der Treiber verwendet die GUID als ID und stellt ihre Eindeutigkeit sicher.

**lpAppPath** - der absolute Pfad zum Modul des Prozesses, der die Daten ausgedruckt hat, MAX\_PATH Bytes.

**lpUserName** - Name des Benutzers, in dessen Kontext der Prozess verwendet wurde, der die Daten gedruckt hat, 128 Bytes.

**lpMachineName** - Netzwerkname des Computers, über den die Daten gedruckt wurden, MAX\_COMPUTERNAME\_LENGTH Bytes.

**IpHardwareCode** - reserviertes Feld, 16 Bytes.

### **Client-Daten**

Wenn der Datenblock [Job Info](#) noch weitere Daten umfasst, beinhalten die nächsten 4 Bytes die Größe der [produktbezogenen Daten](#).

Sofern die Größe ungleich Null ist, folgen die Daten unmittelbar der Feldgröße. Die maximale Größe dieser Daten ist 4 KBytes. Mit den Daten können die ausdrückende Applikation oder der [Konfigurationstreiber](#) den aktuellen Stand der Einstellungen der Applikation an den [Konnektor](#) übergeben.