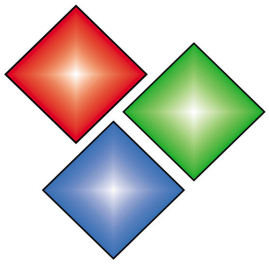


Inforce® XE

Programmierschnittstelle

Version: 2.5.2003

Einführung

Die Programmierschnittstelle von Inforce besteht aus 2 Teilen.

Visual Basic Schnittstelle

Erstens können Sie mit Hilfe von Visual Basic (VB) oder Visual Basic for Applications (VBA) Inforce ansprechen, um beispielsweise neue Datensätze in Inforce einzufügen. Diese Schnittstelle wird beispielsweise benutzt, um von MS Word, MS Excel, den Internet-Explorer oder anderen Office-Applikationen aus über Inforce-Buttons in diesen Applikationen Dokumente, Internet-Adressen, e-Mails etc. in Inforce einzufügen.

Mithilfe dieser Schnittstelle kann Inforce praktisch in beliebige Applikationen integriert werden, so wie Inforce bereits in MS Office Applikationen, den Adobe Acrobat Reader und den Internet Explorer integriert ist: Über einen Inforce-Button in den Symbolleisten dieser Applikationen kann das aktuell in diesen Applikationen geladene Dokument, die Internet-Seite, e-Mail, Kontaktadresse etc. in Inforce eingefügt werden. Es ist lediglich erforderlich, dass eine Applikation über eine VBA-kompatible Makrosprache verfügt oder es über Add-Ins oder Plug-Ins ermöglicht, auf die von Inforce bereitgestellt COM-Schnittstelle zuzugreifen.

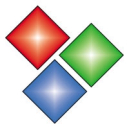
Wie Sie diese Schnittstelle nutzen, sehen Sie am einfachsten am Beispiel der Integration in MS Word und den Internet Explorer. Laden Sie hierzu in MS Word die Dateien **Inforce.dot, Inforce2. dot, InforceIE.htm, InforceIE2.htm** und schauen Sie sich das Makro **DateiIn InforceEinfuegen** an. Über diese Dateien können Sie die Integration in MS Word bzw. Den Internet Explorer auch an Ihre persönlichen Wünsche anpassen, beispielsweise wenn Sie mehr als die standardmäßig übernommenen Daten in Inforce übernehmen möchten.

Inforce Plug-Ins

Zweitens können Sie Plug-Ins in Inforce einbinden. Mit Hilfe von Plug-Ins können zusätzliche Leistungsmerkmale in Inforce integriert werden und Inforce kann an die spezifischen Anforderungen und Wünsche von verschiedenen Anwendern angepasst werden.

Mit Hilfe von Plug-Ins können beispielsweise

- an beliebiger Stelle im Menü neue Menüpunkte sowie Buttons in der Symbolleiste hinzugefügt werden, womit Plug-Ins sich nahtlos in Inforce integrieren,
- wiederkehrende oder umfangreiche Arbeitsabläufe in Inforce automatisisiert werden
- Datensätze, Dokumente, Bereiche oder ganze Bereichsbäume angelegt werden
- externe Daten oder Dokumente automatisch importiert bzw. übernommen werden
- Es kann auf beliebige externe Anwendungen über COM-Schnittstellen zugegriffen werden
- und es kann eigener Code inkl. eigener Dialoge in Inforce integriert werden, um völlig neue Features in Inforce zu integrieren
    - wie beispielsweise das Verschicken von Nachrichten
    - das Erstellen von Dokumenten
    - oder das Exportieren oder Zusammenstellen von Daten, Dokumenten und Bereichsbäumen in eigenen Formaten
- Inforce kann für verschiedene Anwender und Anwendungsgebiete maßgeschneidert werden
    - beispielsweise, indem nur bestimmten Anwendern spezielle Funktionen zur Verfügung gestellt werden
    - oder Menüpunkte für andere Anwender ganz entfernt werden



- oder Menüpunkte umbenannt werden, beispielsweise auch für die Schaffung einer Bedienoberfläche in einer anderen Sprache

Dabei steht Plug-Ins die gesamte Funktionalität von Inforce zur Verfügung. Auch haben Plug-Ins Zugriff auf alle Bereiche und Datensätze einer Wissensbank. Die Möglichkeiten, Inforce mit Hilfe von Plug-Ins zu erweitern und an die persönlichen Wünsche anzupassen, sind damit fast unbegrenzt.

Jeder, der im Besitz entsprechender Entwicklungswerkzeuge ist, kann mit Hilfe des Inforce SDKs eigene Plug-Ins entwickeln. Ein Plug-In ist eine Windows-DLL, die bestimmten Regeln genügen muss. Sobald eine solche DLL in das Unterverzeichnis „*plugins*“ des Inforce-Installationsverzeichnisses kopiert wird, wird dieses Plug-In beim Start von Inforce geladen und die Funktionalität dieses Plug-Ins steht dem Anwender zur Verfügung.

Installation des SDKs

Das Inforce-SDK wird in Form einer ZIP-Datei ausgeliefert. Kopieren Sie die Datei an eine beliebige Stelle auf Ihrem Rechner und entpacken Sie die Datei. Es wird ein Verzeichnis *InforceSDK* angelegt, das das Inforce SDK beinhaltet.

Das Sample Plug-In

Das Inforce-SDK enthält ein Beispiel für ein Inforce Plug-In, das mit einigen einfachen Funktionen demonstriert, was mit Plug-Ins alles möglich ist. Dieses Sample Plug-In führt folgende Aktionen aus:

1. Es entfernt den Menüpunkt **Extras/Programme**
2. Es benennt den Menüpunkt **Extras/Verzeichnisse** in Extras/Ordner um
3. Es fügt die folgenden Menüpunkte zum Inforce-Menü und jeweils einen Button in der Inforce-Symbolleiste hinzu

    a. **Extras/Bereiche und Datensätze zählen**: Es werden die Bereiche und Datensätze der gesamten Wissensbank sowie des ausgewählten Bereichs gezählt. Diese Funktion demonstriert, wie einfach es ist, für alle Datensätze und Bereiche einer Wissensbank eine Aufgabe zu erledigen.

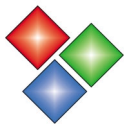
    b. **Extras/Dateien auflisten**: Es werden alle Dateien aufgelistet, auf die in dem selektierten Bereich sowie seinen Teilbereichen verwiesen wird. Diese Funktion demonstriert, dass Plug-Ins Zugriff auf alle Datensätze und deren Daten haben.

    c. **Bearbeiten/Neues Info „Aufgabe“ anlegen**: Es wird ein neuer Datensatz vom Typ „Aufgabe“ angelegt, ohne dass der Typ vorher beim Anwender erfragt wird, wie dies bei dem Standard-Menüpunkt „Neues Info“ der Fall ist. Der neue Datensatz wird mit einigen Daten initialisiert.

    d. **Struktur/Bereichsbaum ABCD anlegen**: Diese Funktion legt auf einen Schlag 4 Bereiche an, im Gegensatz zum Standard-Menübefehl „Neuer Bereich“, der nur einen Bereich anlegt. Diese Funktion demonstriert damit, dass Plug-Ins auch automatisch Daten aus beliebigen Formaten einlesen und automatisch eine entsprechende Bereichsstruktur aufbauen können

    e. **Struktur/Bereich bearbeiten**: Diese Funktion schaltet die Dialogbox zum Bearbeiten des aktuell ausgewählten Bereichs auf. Sie demonstriert damit, dass Plug-Ins Zugriff auf das gesamte Inforce-Menü haben.

    In der Gesamtheit demonstrieren diese Punkte, dass Plug-Ins Menüpunkte an beliebiger Stelle im Inforce-Menü einfügen können, sich dadurch nahtlos in Inforce einfügen und die Funktionalität von Inforce praktisch beliebig erweitern können.



Sample Plug-In installieren

Um das Sample-Plug-In zu installieren, genügt es, die Datei *SampleIfcPlugIn.dll* aus dem Verzeichnis *InforceSDK\samples\SampleIfcPlugIn\Release* in das Verzeichnis *c:\Programme\Inforce\plugins* zu kopieren (oder das entsprechende Inforce-Installationsverzeichnis auf Ihrem Rechner).

Sobald Sie Inforce starten, wird das Plug-In automatisch geladen und es führt die obigen Änderungen am Inforce-Menü aus.

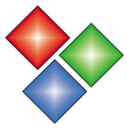
Unter dem Menüpunkt *?/Info über Plug-Ins* erhalten Sie Informationen über alle installierten Plug-Ins und welche Änderungen/Erweiterungen diese am Inforce-Menü vorgenommen haben, so auch für das Sample Plug-In.

Das Sample Plug-In kompilieren

Um zu überprüfen, ob Ihre Entwicklungsumgebung richtig installiert und konfiguriert ist, können Sie das Sample Plug-In kompilieren. Starten Sie hierzu Visual Studio C++ 6.0 und laden Sie *SampleIfcPlugIn.dsp* über den Menüpunkt **Datei/Arbeitsbereich öffnen**.

Tragen Sie über den Menüpunkt Extras/Optionen/Verzeichnisse das Include-Verzeichnis *XXX\InforceSDK\include* sowie das Bibliotheksverzeichnis *XXX\InforceSDK\lib* ein. Dabei bezeichnit XXX den Pfad, unter dem Sie das SDK installiert haben.

Nun müssten Sie das Sample-Plug-In über den Menüpunkt **Erstellen/Alles neu erstellen** neu erstellen können. Danach können Sie die SampleIfcPlug.dll aus dem Release-Verzeichnis erneut ins Inforce\plugins Verzeichnis kopieren, um sicherzustellen, dass alles richtig installiert ist.



Den Code des Sample Plug-Ins verstehen

Die Implementierung des Sample Plug-Ins befindet sich in der Datei SampleIfcPlugIn.cpp. Das Plug-In ist eine Klasse, die von **CIPlugInBase** abgeleitet ist und verschiedene Funktionen dieser Basisklasse überschreibt.

GetInforcePlugIn()

Bei Programmstart ruft Inforce die Funktion *GetInforcePlugIn()* der Plug-In-DLL auf, die eine Instanz der Plug-In-Klasse zurückliefern muss.

OnStartup(IIUserInterface* pUserInterface)

Anschließend ruft Inforce die Methode *OnStartup()* für dieses Objekt dieser Klasse auf. In dieser Methode kann das Plug-In mit Hilfe des mitgelieferten Zeigers *pUserInterface* neue Menüpunkte und Buttons anlegen, Menüpunkte entfernen oder umbenennen.

Beim Anlegen neuer Menüpunkt wird ein Objekt vom Typ *BICommand* zurückgeliefert, das das Plug-In sich merken sollte, falls mehr als ein neuer Menübefehl anmeldet wird.

OnCommand(const BICommand& Command)

Immer dann, wenn ein Anwender einen Menüpunkt oder einen Button des Plug-Ins anklickt, wird von Inforce die Methode *OnCommand* aufgerufen. Dieser wird der Parameter *Command* mitgegeben, der angibt, welcher Befehl angeklickt wurde. Das Sample-Plug-In implementiert fünf neue Befehle. Entsprechend wird in der Methode *OnCommand* je nach Befehl, den der Anwender ausgewählt hat, eine andere Methode aufgerufen.

DoCommandCount

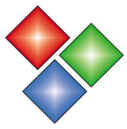
Falls der Anwender den Befehl **Bereiche und Datensätze zählen** angeklickt hat, wird diese spezielle Methode des Sample Plug-In aufgerufen. Um die Bereiche und Datensätze zu zählen, wird jeweils eine der Methoden ***DoCommandForAllRecords***, etc. aufgerufen. Diese werden von der Basisklasse *CIPlugIn* bereitgestellt und rufen der Reihe nach die Methoden *DoCommandForRecord* bzw. *DoCommandForTopic* etc. für alle Datensätze/Bereiche der Wissensbank auf. Diese Methoden müssen von dem speziellen Plug-In überschrieben werden. In ihnen muss der Code implementiert werden, der für jeden Datensatz, Bereich etc. ausgeführt werden soll. Um das Durchlaufen der gesamten Wissensbank muss sich das abgeleitete Plug-In nicht kümmern. Im Fall des Sample Plug-Ins wird in der Methode **DoCommandForRecord** nur ein Zähler hochgezählt, um die Datensätze zu zählen.

DoCommandCreateTopicsABCD

Falls der Anwender den Befehl **Bereichsbaum ABCD anlegen** angeklickt hat, ruft OnCommand() die für das Sample Plug-In spezifische Methode *DoCommand CreateTopicsABCD* auf. Diese benutzt die Tools, die dem Plug-In von Inforce zur Verfügung gestellt wird (beispielsweise **IITopicEditor** etc. su.), um Informationen über den vom Anwender selektierten Bereich zu erhalten und neue Bereiche und Datensätze anzulegen.

DoCommandListFiles

Für den Befehl **Dateien auflisten** des Sample Plug-Ins ist die Funktion DoCommandListFiles zuständig Ähnlich wie DoCommandCount benutzt diese Funktion die Funktion **DoCommandForAllRecordsInTopicTree** der Basisklasse, um durch alle Datensätze des vom Anwender ausgewählten Bereichs und seiner Teilbereiche zu iterieren. In der Funktion DoCommandForRecord wird aber nun nicht nur ein Zähler hochgezählt, sondern auf den Datensatz zugegriffen und überprüft, ob diesem eine Datei zugeordnet ist. Falls ja, wird der Dateiname in eine Datei geschrieben, die am Ende angezeigt wird.



Die Hilfsklassen **CIPathname** und **CIFile** wurden benutzt, um auf den Dateinamen eines Datensatzes zuzugreifen bzw. um in ein File zu schreiben. Sie können statt *CIFile* aber auch Ihre eigenen Klasse benutzen, um Dateien zu öffnen etc. Auch können Sie das gesamte Windows SDK, die MFC oder andere Klassenbibliotheken benutzen.

Für den Befehl **Bereich bearbeiten (Sample PlugIn)** ist keine spezielle Methode implementiert worden. Der Code ist direkt in OnCommand eingefügt worden. Dort wird das Tool **MenuHandler** benutzt, um einen der standardmäßig vorhandenen Menübefehle von Inforce auszuführen. Eine Liste der vorhandenen Menübefehle findet sich in InforceBitclasses.h in der Klasse BICommand.

Ein Inforce Plug-In selbst erstellen

Um ein eigenes Plug-In zu erstellen, sollten Sie die Datei ***IfcPlugInWizard.awx*** aus dem InforceSDK\lib-Verzeichnis in das Template-Verzeichnis von Visual Studio kopieren, typischerweise also c:\Programme\Microsoft Visual Studio\Common\MSDEV 98\Template.

Denn wenn Sie dann im Visual Studio den Menüpunkt **Datei/Neu** wählen, sehen Sie in der Listbox **Projekte** den Eintrag **Inforce Plug-In Assistent**. Wählen Sie diesen Eintrag aus, wählen Sie ein Verzeichnis aus und einen Namen für Ihr Projekt an und klicken Sie auf OK. Der Inforce-Plug-In Assistent erstellt für Sie automtisch ein Inforce-Plug-In-Projekt mit dem angegebenen Namen. Dieses Projekt ist sofort compilierbar und sie erhalten sofort ein einfaches Plug-In, das einen Menüpunkt und einen Button anlegt.

In der Folge müssen Sie nur noch den Code in der .cpp – Datei implementieren, sowie gegebenenfalls einige Resourcen bearbeiten. Im einzelnen sollten Sie die folgenden Funktionen implementieren.

Wichtige Methoden

OnStartup(IIUserInterface* pUserInterface)

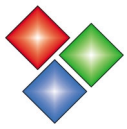
In *OnStartup()* legen Sie Menüpunkte an, können Menüpunkte entfernen und umbenennen. Sie können sich hierbei am Code des Sample Plug-Ins orientieren. Beim Anlegen neuer Befehle können Sie einen Tooltip-Text sowie einen Hilfetext für die Statuszeile und die Info-Dialogbox sowie eine Bitmap für den Toolbar-Button angeben. Ihnen stehen über den Zeiger *pUserInterface* alle Methoden der Klasse IIUserInterface zur Verfügung. Diese finden Sie in der Datei *InforceInterfaces.h* aufgelistet. Die zurückgelieferten BICommand-Objekte sollten Sie sich merken, ähnlich wie dies auch im Sample-Plug-In getan wird.

OnCommand(const BICommand& Command)

In dieser Funktion muss der Code implementiert werden, der ausgeführt werden soll, wenn der Anwender einen der Menüpunkte oder einen Toolbar-Button anklickt. Falls Ihr Plug-In mehrere Befehle unterstützt, sollten Sie für jeden Befehl eine DoCommandXXX-Funktion erstellen und diese aus OnCommand heraus aufrufen, ähnlich wie im Sample-Plug-In demonstriert.

In OnCommand bzw. den DoCommand-Funktionen haben Sie Zugriff auf die Tools, die Ihnen von Inforce zur Verfügung gestellt werden, um die aktuelle Wissensbank zu bearbeiten bzw. um Informationen über die Wissensbank abzufragen. Unter andem liefern die folgenden von *CIPlugIn* bereitgestellten Methoden solche Tools zurück:

GetTopicEditor(): Liefert eine Referenz auf ein **IITopicEditor**-Objekt zurück, mit dem Bereiche angelegt, editiert und gelöscht werden können.



GetInfoEditor(): Liefert eine Referenz auf ein **IIInfoEditor**-Objekt zurück, mit dem Datensätze angelegt, editiert und gelöscht werden können.

GetInfobaseView(): Liefert eine Referenz auf ein **IIInfobaseView**-Objekt zurück, das beispielsweise Auskunft gibt über die aktuell vom Anwender ausgewählten Bereiche und oder Datensätze.

GetMenuHandler(): Liefert eine Referenz auf ein **IIMenuHandler**-Objekt zurück, mit dessen Methode DoCommand() alle Menübefehle von Inforce direkt aufgerufen werden können.

GetSearcher(): Liefert eine Referenz auf ein **IISearcher**-Objekt zurück, mit dem in einer Wissensbank gesucht werden kann.

Welche Methoden die verschiedenen Tools im Detail zur Verfügung stellen, können Sie der Datei *InforceInterfaces.h* entnehmen, die sich im InforceSDK\include-Verzeichnis befindet. Mit weiteren Tools können Sie Daten importieren, exportieren, Dateien einlesen, Reports und Websites erstellen etc.

Datenklassen

IIRecord

Über das IIRecord-Interface kann auf einen Datensatz zugegriffen werden. Es können die einzelnen Felder des Datensatzes abgefragt und gesetzt werden, darunter auch auch der Name der Datei bzw. der Internet-Adresse, die dem Datensatz zugeordnet ist.

IITopic

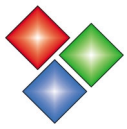
IITopic ist das Interface, über das auf Bereiche zugegriffen werden kann. Mittels IITopic kann beispielsweise der Name des Bereichs erfragt bzw. gesetzt werden und kann auf die einzelnen Datensätze, die dem Bereich zugeordnet sind, zugegriffen werden.

IIRecordType

Jeder Datensatz verfügt über einen bestimmten Typ, der beispielsweise festlegt, über welche Felder der Datensatz verfügt. So verfügt beispielsweise ein Datensatz vom Typ Buch über Felder wie Autor, Herausgeber, Verlag, während ein Datensatz vom Typ InternetLink diese Felder nicht enthält. IIRecordType ist das Interface, um Informationen zu dem Typ des Datensatzes zu erfragen, beispielsweise den Namen des Typs und die Liste der Felder.

IIRecordRep

In Inforce kann jeder Datensatz beliebig vielen Bereichen zugeordnet sein und dementsprechend in den Datensatz-Listen zu diesen Bereich (den sogenannten Bereichsfenstern) sichtbar sein. Der Datensatz selbst ist nur einmal in der Wissensbank enthalten, ist aber an verschiedenen Stellen sichtbar und über diese Stellen auch jeweils zugreifbar. Ein IIRecordRep (Representative of an IIRecord) stellt die einzelne Stelle dar, an der ein Datensatz sichtbar ist. Zu jedem IIRecordRep gehört ein IIRecord, aber zu einem IIRecord können beliebig viele IIRecordReps gehören. In einfachen Wissensbanken, in denen jeder Datensatz an genau einer Stelle eingeordnet ist, gehört jedoch zu jedem IIRecord auch nur genau ein IIRecordRep.



IITopicRep

Genau wie Datensätze an beliebig vielen Stellen der Wissensbank eingeordnet sein können, können auch ganze Bereiche an mehreren Stellen der Wissensbank eingeordnet sein. Ein IIRecordRep entspricht einer sichtbaren Position eines Bereich an einer bestimmten Stelle in der Wissensbank. Dementsprechend gehört zu jedem IITopicRep ein IITopic und in einfachen Wissensbanken, die rein hierarchisch strukturiert sind, gehört auch zu jedem IITopic genau ein IITopicRep. In komplexeren Wissensbanken, in denen derselbe Bereich (wirklich derselbe Bereich und nicht nur eine kopierte Version) an mehreren Stellen erscheint, können zu einem IITopic mehrere IITopicReps gehören.

Hilfsklassen

IIPathname, CIPathname

Das Interface IIPathname definiert den Pfadnamen einer lokalen Datei oder die URL einer Internet-Seite. Wenn Sie ein Objekt von diesem Typ benötigen (beispielsweise, wenn Sie die einem Datensatz zugehörige Datei erfragen wollen), können Sie ein Objekt des Typs CIPathname benutzen. Von dieser Klasse können Sie auch lokale Objekte erstellen.

IIFile, CIFile

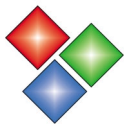
Die Klasse CIFile können Sie benutzen, wenn Sie eine Datei öffnen und in sie schreiben oder aus ihr lesen möchten. Sie können hierfür aber auch eigene Klassen oder auch das Windows SDK benutzen.

Name und About-Dialog des Plug-Ins

Um Ihr Plug-In eindeutig identifizierbar zu machen, müssen Sie ihm einen Namen geben, indem Sie in der Funktion GetName() Ihrer Plug-In-Klasse diesen Namen zurückgeben.

Desweiteren zeigt Inforce für jedes Plug-In über einen Menüpunkt im Menü **?/Info über Plug-Ins** ein Dialogfeld mit Informationen zu diesem Plug-In zu. Inforce benutzt hierzu die Dialogvorlage IDD_ABOUT, die Sie in dem Plug-In-Projekt vorfinden. Dieses Dialogfeld können Sie nach eigenen Wünschen gestalten, solange Sie einige Regeln beachten:

1. Die ID der Dialogvorlage darf nicht geändert werden. Sie muss 100 sein, so wie sie vom Assistenten erzeugt wird.
2. Das Mehrzeilen-Editfeld IDC_PLUGIN_MENU_INFO_TEXT muss in der Dialogvorlage in mindestens dieser Größe enthalten sein. In diesem Editfeld informiert Inforce darüber, welche Änderungen das Plug-In am Menü durchgeführt hat, damit der Anwender nachvollziehen kann, welches Plug-In wofür verantwortlich ist. Auch diese ID muss 103 bleiben.
3. Sie können das in der Standard-Dialogvorlage enthaltene Icon-Resource ändern. Die ID der Resource, also IDR_PLUGIN muss jedoch 101 bleiben. Dies stellen Sie am einfachsten sicher, indem Sie kein neues Icon einfügen, sondern das vom Assistenten angelegte Icon bearbeiten.
4. Die ID des Symbol auf der Dialogvorlage darf nicht geändert werden (Vom Assistenten wird IDC_PLUGIN_SYMOL). Sie muss 102 bleiben.



Weitere Informationen

Das Inforce SDK ist kostenlos. Inforce Plug-Ins können frei verteilt oder vertrieben werden.

Das Inforce SDK sowie nähere Informationen zu Inforce und zur Entwicklung von Plug-Ins für Inforce erhalten Sie bei:

SDS Software
Dirk Sandhorst
Agnesstr. 24A
22301 Hamburg
Tel: 040 / 200 99 74
e-Mail: info@inforce.de
Internet: http://www.inforce.de