Телефон в Ижевске: +7 (3412) 72-11-00
Телефон в Москве: +7 (499) 277-1560
Сайт: www.directum.ru
E-mail: support@directum.ru

Инструкция по использованию единой функциональной библиотеки интеграции

Назначение документа

Документ описывает библиотеку в целом, вызов функций библиотеки, каждую функцию библиотеки и типы используемых данных.

Содержание

Назначение библиотеки

Единая функциональная библиотека интеграции с DIRECTUM предназначена для облегчения создания собственных надстроек над приложениями для интеграции с системой DIRECTUM. Библиотека содержит функции для реализации основных действий интеграции. Эти функции можно использовать при разработке собственных надстроек. Физически библиотека представляет собой файл UnifiedISBuilderIntegrationLibrary.exe. Подробнее см. документацию системы DIRECTUM, руководство администратора.

После настройки интеграции в приложении на ленте появится группа **DIRECTUM**. В этом пункте доступны все действия интеграции, реализованные в надстройке. Подробнее см. документацию системы DIRECTUM, руководство администратора.

История работы единой функциональной библиотеки интеграции ведется в журнале событий. По умолчанию название журнала состоит из имени компьютера, на котором совершались действия, и имени библиотеки интеграции. Например, «W32_UnifiedLibrary.log».

Включить ведение журнала можно с помощью файла конфигурации NPOComputer.MSOfficeAddin.config. Подробнее см. в руководстве администратора системы DIRECTUM, в главе «Настройка модулей системы DIRECTUM», раздел «Файл NPOComputer.MSOfficeAddin.config».

Рекомендуемый состав действий в надстройке

В надстройках над приложениями-редакторами документов системы DIRECTUM рекомендуется реализовывать действия интеграции:

- создание нового документа из шаблона;
- сохранение документа в системе DIRECTUM;
- подписание документа;
- отображение информации о подписях документа в виде отчета и стандартного диалога IS-Builder;
- копирование ссылки на документ в буфер обмена;
- отправка документа вложением в задачу;
- отправка ссылки на документ в письме;
- отображение истории работы с документом.

В надстройках над почтовыми клиентами рекомендуется реализовывать действия интеграции:

- поиск контактных лиц организаций;
- вставка документа во вложение текста письма;
- выбор получателей из справочника контактных лиц организаций;
- сохранение текста письма в системе DIRECTUM;
- сохранение вложения в системе DIRECTUM;
- сохранение контакта с клиентом. DeleteSpecialSymbolsFromString

В зависимости от возможностей приложения, с которым осуществляется интеграция, в надстройке могут быть реализованы:

- вставка реквизита в текст документа;
- вставка штрихкода в текст документа;
- обновление полей в тексте документа;
- сравнение версий документа.

Вызов функций библиотеки

Перед использованием библиотеки следует убедиться, что файл UnifiedISBuilderIntegrationLibrary.exe зарегистрирован. Подробнее см. документацию системы DIRECTUM, руководство администратора.

Перед вызовом функций необходимо выполнить код:

```
Set IntLib = CreateObject("UnifiedISBuilderIntegrationLibrary.Connect")
```

Далее может следовать один или несколько вызовов функций. Например:

```
Call IntLib.ShowCreateEDocumentFromTemplateDialog
```

Примечания

- функции библиотеки всегда работают с той системой DIRECTUM, код которой указан в ключе реестра HKCR*\shellex\ContextMenuHandlers\IS_Builder\SysCode;

- если в функцию библиотеки передано имя документа, который не является документом системы DIRECTUM, то будет появится сообщение о том, что выполняемое действие доступно только для документов DIRECTUM.

Функции

CopyHyperlinkToClipboard – скопировать гиперссылку в буфер

Синтаксис:

```
procedure CopyHyperlinkToClipboard(
  DocumentName: WideString);
```

Параметры:

DocumentName – имя файла активного документа приложения без указания пути.

Описание:

Функция копирует гиперссылку на документ системы DIRECTUM в буфер обмена.

Пример:

```
' Вставить ссылку на документ в текущую позицию курсора.
Call IntLib.CopyHyperlinkToClipboard(ActiveDocument.Name)
```

DeleteSpecialSymbolsFromString – удалить недопустимые символы в названиях документов DIRECTUM

Синтаксис:

```
function DeleteSpecialSymbolsFromString (
  InitialString: WideString): WideString;
```

Параметры:

InitialString – строка, которую нужно проверить на содержание недопустимых символов. Например, функция может проверить имя файла на локальном диске.

Возвращаемое значение:

Строка, в которой все недопустимые символы, а именно «/», «:», «|», «*», «?», «"», «<» и «>», заменены на символ «_».

Описание:

Функция заменяет все символы, недопустимые для использования в названиях документов DIRECTUM, на символ «_». Рекомендуется использовать в случае создания документов системы DIRECTUM из файлов на локальном диске, например при сохранении вложения в электронное письмо как документа DIRECTUM.

```
' Получить вложение в электронное письмо как объект Outlook.
Dim Attach As Outlook.Attachment
Attach = Mail.Attachments.Item( 1)
' Вырезать недопустимые символы в имени вложения.
Path = IntLib.DeleteSpecialSymbolsFromString(Attach.DisplayName)
' Сохранить вложение на локальный диск.
Call Attach.SaveAsFile(Path)
```

Пример:

```
' Получить вложение в электронное письмо как объект Outlook.
Dim Attach As Outlook.Attachment
Attach = Mail.Attachments.Item( 1)
' Вырезать недопустимые символы в имени вложения.
Path = IntLib.DeleteSpecialSymbolsFromString(Attach.DisplayName)
' Сохранить вложение на локальный диск.
Call Attach.SaveAsFile(Path)
```

DirectumFieldCode – проверить, является ли поле полем системы DIRECTUM

Синтаксис:

```
function DirectumFieldCode(
  const FieldCode: WideString): WordBool;
```

Параметры:

FieldCode – код поля.

Возвращаемое значение:

True, если указанное поле является полем системы DIRECTUM, иначе **False**.

Описание:

Функция определяет по указанному коду поля, является ли оно полем системы DIRECTUM, вставленным из реквизита типа «Справочник».

Пример:

```
' Проверить, является ли поле под курсором полем из DIRECTUM.
DirectumReferenceField = IntLib.DirectumFieldCode( _
  Application.Selection.Fields(1).Code.Text)
```

DirectumHyperlink – проверить, является ли гиперссылка гиперссылкой на объект DIRECTUM

Синтаксис:

```
function DirectumHyperlink(
  const HyperlinkAddress: WideString): WordBool;
```

Параметры:

HyperlinkAddress – адрес гиперссылки.

Возвращаемое значение:

True, если гиперссылка указывает на объект текущей системы DIRECTUM, иначе **False**.

Описание:

Функция проверяет, указывает ли заданная гиперссылка на объект текущей системы DIRECTUM.

Пример:

```
' Проверить, является ли гиперссылка под курсором гиперссылкой DIRECTUM.
HyperlinkAddress = Application.Selection.Hyperlinks.Item(1).Address
DirectumHyperlink = IntLib.DirectumHyperlink(HyperlinkAddress)
```

DocumentCardLocked – проверить, заблокирована ли карточка документа

Синтаксис:

```
function DocumentCardLocked(
  const DocumentName: WideString;
  out UserName: WideString): WordBool;
```

Параметры:

- **DocumentName** – имя файла активного документа приложения без указания пути;
- **UserName** – имя пользователя, который заблокировал карточку документа.

Возвращаемое значение:

True, если карточка документа заблокирована, иначе **False**.

Описание:

Функция проверяет, заблокирована ли карточка документа. Если карточка заблокирована, то в параметре **UserName** возвращается имя пользователя, который заблокировал карточку документа. Если карточка не заблокирована, то параметр **UserName** содержит пустую строку.

Если документ не найден в системе DIRECTUM, то функция выдает сообщение об ошибке и возвращает значение **True**.

Пример:

```
' Проверить, заблокирована ли карточка документа.
' DocumentName - имя документа.
Dim UserName As String
If IntLib.DocumentCardLocked(DocumentName, UserName) Then
  MsgBox "Документ заблокирован пользователем" & UserName
Else
  MsgBox "Документ никем не заблокирован"
End If
```

DocumentOpenedFromDIRECTUM – проверить, открыт ли документ из системы DIRECTUM

Синтаксис:

```
function DocumentOpenedFromDIRECTUM(
  FullDocumentName: WideString): WordBool;
```

Параметры:

FullDocumentName – имя файла активного документа приложения с указанием полного пути.

Возвращаемое значение:

True, если документ открыт из системы DIRECTUM, иначе **False**.

Описание:

Функция проверяет, открыт ли активный документ приложения из системы DIRECTUM или из папки на локальном компьютере.

Пример:

```
' Проверить, открыт ли документ из системы DIRECTUM.
If IntLib.DocumentOpenedFromDIRECTUM(ActiveDocument.FullName) Then
  …
End If
```

DocumentVersionLocked – проверить, заблокирована ли версия документа другим пользователем (открытие версии документа на редактирование)

Синтаксис:

```
function DocumentVersionLocked(
  VersionID: Integer;
  out UserName: WideString): WordBool;
```

Параметры:

- **VersionID** – идентификатор открытой версии документа. В качестве значения параметра следует передавать значение поля **XRecID** таблицы **SBEDocVer**;
- **UserName** – имя пользователя, который заблокировал документ.

Возвращаемое значение:

True, если версия документа заблокирована, иначе **False**.

Описание:

Функция проверяет, заблокирована ли указанная версия документа. Если версия заблокирована, то в параметре **UserName** возвращается имя пользователя, который заблокировал версию документа. Если версия не заблокирована, то параметр **UserName** содержит пустую строку.

Если документ не найден в системе DIRECTUM, то функция выдает сообщение об ошибке и возвращает значение **True**.

Пример:

```
' Проверить, заблокирована ли карточка документа.
Dim UserName As String
DocInfoXML = IntLib.GetEDocumentInfo(ActiveDocument.FullName)
Set XMLFile = CreateObject("MSXML.DOMDocument")
Call XMLFile.loadXML(DocInfoXML)
VersionID = XMLFile.selectSingleNode("CurrentVersionID").Text
If IntLib.DocumentVersionLocked(VersionID, UserName) Then
  MsgBox "Версия документа заблокирована пользователем" & UserName
Else
  MsgBox "Версия документа никем не заблокирована"
End If
```

DocumentVersionLockedManually – проверить, была ли версия заблокирована через проводник DIRECTUM

Синтаксис:

```
function DocumentVersionLockedManually(
  VersionIndex: Integer;
  DocumentName: WideString;
  out VersionExtractor: WideString): WordBool;
```

Параметры:

- **VersionIndex** – индекс версии, которую нужно проверить на факт блокировки через проводник одним из пользователей системы. При этом индекс версии не соответствует номеру версии документа.

Возвращаемое значение:

True, если версия документа была заблокирована через проводник системы, иначе **False**.

Описание:

Функция проверяет, была ли версия документа заблокирована через проводник одним из пользователей системы. Если версия была заблокирована, то в параметре **VersionExtractor** возвращается имя пользователя, осуществившего блокировку. Рекомендуется использовать в

случае создания документов системы DIRECTUM из файлов на локальном диске, например при сохранении вложения в электронное письмо как документа DIRECTUM.

Пример:

```
' Проверить, была ли версия документа заблокирована вручную.
Dim DocumentVersionExtractor As String
DocumentLockedManually = IntLib.DocumentVersionLockedManually( _
  DocumentVersion, DocumentName, DocumentVersionExtractor)
' Если версия была заблокирована, оповестить об этом.
If DocumentLockedManually Then
  Call MsgBox("Версия документа была заблокирована пользователем" _
    & DocumentVersionExtractor)
End If
```

ExecuteComponentToken – выполнить вариант запуска компоненты

Синтаксис:

```
procedure ExecuteComponentToken(
  TokenID: Integer);
```

Параметры:

TokenID – ИД варианта запуска.

Описание:

Функция запускает компоненту, на которую указывает заданный вариант запуска.

Пример:

```
' Выполнить вариант запуска компоненты.
Call IntLib.ExecuteComponentToken(12345)
```

ExecuteComponentTokenInNewProcess – выполнить вариант запуска компоненты в новом процессе

Синтаксис:

```
procedure ExecuteComponentTokenInNewProcess(
  TokenID: Integer);
```

Параметры:

TokenID – ИД варианта запуска.

Описание:

Функция запускает компоненту, на которую указывает заданный вариант запуска, в отдельном процессе.

Пример:

```
' Выполнить вариант запуска компоненты в отдельном процессе.
Call IntLib. ExecuteComponentTokenInNewProcess(12345)
```

ExportEDocumentVersion – экспортировать версию документа

Синтаксис:

```
Function ExportEDocumentVersion(
  const DocumentName: WideString;
  DocumentVersionNumber: Integer): WideString;
```

Параметры:

- **DocumentName** – имя файла активного документа приложения без указания пути;
- **DocumentVersionNumber** – номер экспортируемой версии документа.

Возвращаемое значение:

Строка, содержащая полный путь к файлу версии документа.

Описание:

Функция экспортирует версию документа, указанную в параметре **DocumentVersionNumber**, во временную папку текущего пользователя. Если не удается экспортировать документ, то функция возвращает пустую строку.

Пример:

```
' Экспортировать и открыть документ.
' DocumentName - имя документа.
' DocumentVersionNumber - номер версии документа.
ExportedFileName = IntLib.ExportEDocumentVersion(DocumentName, _
  DocumentVersionNumber)
If ExportedFileName <> "" Then
  Set Word = CreateObject("Word.Application")
  Word.Open(ExportedFileName)
End If
```

ExportEDocumentVersionToCompare – экспортировать версию документ для сравнения

Синтаксис:

```
procedure ExportEDocumentVersionToCompare(
  DocumentName: WideString;
  var DestinationPath: WideString);
```

Параметры:

- **DocumentName** – имя файла активного документа приложения без указания пути;
- **DestinationPath** – полный путь к файлу версии документа.

В качестве входного значения параметра можно передавать пустую строку или полный путь к файлу без указания расширения. Если в параметре передается пустая строка, то документ будет сохранен во временной папке пользователя, иначе – в указанном в файле.

После выполнения функции параметр содержит полный путь к файлу, в который экспортирована версия документа.

Описание:

Функция отображает диалог выбора версии документа, переданного в параметре **DocumentName**. Если выбранная версия совпадает с открытой в активном документе, то выдается предупреждение, и диалог выбора версии отображается повторно. Если выбранная версия отличается от открытой в активном документе, то функция экспортирует данную версию в файл, указанный в параметре **DestinationPath**, либо во временную папку пользователя.

Если активный документ не является документом системы DIRECTUM или не удается выполнить экспорт версии, то параметр **DestinationPath** содержит пустую строку.

Рекомендуется использовать данную функцию для сравнения версий документа средствами приложения-редактора.

Пример:

```
' Сравнить версии документа.
VersionPath = ""
Call IntLib.ExportEDocumentVersionToCompare(ActiveDocument.Name, Path)
If Path <> "" Then
  Call ActiveDocument.Merge(Path)
End If
```

GetBarCodeImage – сгенерировать изображение штрихкода

Синтаксис:

```
procedure GetBarCodeImage(
  DocumentName: WideString;
  var DestinationPath: WideString;
  out Width: Double;
  out Height: Double;
  out XBarCodeHash: String);
```

Параметры:

- **DocumentName** – имя файла активного документа приложения без указания пути;
- **DestinationPath** – полный путь к файлу с изображением штрихкода.

В качестве входного значения параметра можно передавать пустую строку либо полный путь к файлу. Если в параметре передается пустая строка, то документ будет сохранен во временной папке пользователя, иначе – в указанном в файле.

После выполнения функции параметр содержит:

- полный путь к файлу, в котором находится изображение штрихкода;
- **Width** – ширина изображения в миллиметрах;
- **Height** –высота изображения в миллиметрах.
- **XBarCodeHash** – строка, содержащая захешированную метаинформацию штрихкода.

Описание:

Функция генерирует изображение одномерного или двумерного штрихкода для указанного в параметре **DocumentName** документа и сохраняет это изображение в файл, указанный в параметре **DestinationPath**. Независимо от указанного в параметре расширения файла создается изображение в формате WMF. Если параметр **DestinationPath** не указан, то имя файла

будет сгенерировано автоматически. Формат штрихкода зависит от настроек, заданных в справочнике **Форматы штрихкодов документов** для соответствующего вида документа. Подробнее см. документацию DIRECTUM, руководство по модулю «Канцелярия», в главе «Быстрая идентификация бумажных документов», описание справочника **Форматы штрихкодов документов**. Если настройки формата штрихкода для данного вида документа не заданы или запись справочника закрыта, то тип штрихкода берется одномерный, а штрихкод формируется в формате <Префикс>-<ИД инсталляции>-<ИД документа>.

В параметрах **Width** и **Height** возвращается реальная ширина и высота полученного изображения в миллиметрах. Эти данные необходимы для вставки изображения в документ с последующей печатью, чтобы сканер в дальнейшем смог корректно распознать штрихкод.

В параметре **XBarCodeHash** возвращается строка, содержащая захешированную метаинформацию штрихкода. Параметр используется для определения необходимости замены штрихкода в тексте документа.

Если при генерации штрихкода произошла ошибка, то параметр **DestinationPath** содержит пустую строку.

Рекомендуется использовать данную функцию для вставки в документ штрихкода непосредственно из приложения-редактора.

Пример:

```
' Сгенерировать изображение штрихкода и вставить его в текущую позицию.
Dim Width, Height As Double
Dim BarCodeHash As String = ""
Path = ""
Call IntLib.GetBarCodeImage(ActiveDocument.Name, Path, Width, Height, BarCodeHash)
Set BarCodePicture = Selection.InlineShapes.AddPicture(Path)
BarCodePicture.Width = MillimetersToPoints(Width)
BarCodePicture.Height = MillimetersToPoints(Height)
' Вставить в документ свойство XBarCodeHash (Хеш-сумма типа и строки штрих-кода на момент его вставки в документ)
Call ActiveDocument.CustomDocumentProperties.Add("XBarCodeHash", False,
MsoDocProperties.msoPropertyTypeString, BarCodeHash)
```

GetConstantValue – получить значение константы DIRECTUM

Синтаксис:

```
function GetConstantValue(
  const ConstantName: WideString;
  const OrganizationCode: WideString): WideString;
```

Параметры:

- **ConstantName** – имя константы в компоненте **Константы**;
- **OrganizationCode** – код нашей организации из справочника **Наши организации**.

Возвращаемое значение:

Строка, содержащая значение константы.

Описание:

Функция возвращает значение константы **ConstantName** для организации **OrganizationCode**. Для общих констант параметр **OrganizationCode** указывать не нужно.

Если константы **ConstantName** не существует или константа индивидуальная, а значение для организации **OrganizationCode** не задано, то функция возвращает пустую строку.

Пример:

```
' Получить значении константы RegistrateActionNames.
RegistrateActionNames = GetConstantValue('RegistrateActionNames', '')
```

GetCurrentLanguage – получить текущий язык системы DIRECTUM

Синтаксис:

```
function GetCurrentLanguage: WideString;
```

Возвращаемое значение:

Строка, содержащая код языка.

Описание:

Функция возвращает двухбуквенный код языка текущей системы DIRECTUM.

Пример:

```
' Получить код языка системы.
LanguageCode = GetCurrentLanguage()
```

GetEDocumentInfo – получить информацию о документе

Синтаксис:

```
function GetEDocumentInfo(
  FullDocumentName: WideString): WideString;
```

Параметры:

FullDocumentName – имя файла активного документа, с указанием полного пути.

Возвращаемое значение:

Строка, содержащая информацию о документе.

Описание:

Функция возвращает строку в формате XML со значениями некоторых реквизитов документа. Если указанный документ не является документом DIRECTUM, то функция возвращает пустую строку.

Функция возвращает значения реквизитов самого документа и открытой версии. Узлы XML-документа:

- **CurrentVersionSigned** – признак того, что версия документа подписана;
- **DateModified** – дата последней модификации файла документа;

- **EncodeType** – тип шифрования документа;
- **Extension** – расширение файла документа;
- **ExportedWithLock** – признак того, что документ был экспортирован с блокировкой;
- **HasBoundDocuments** – наличие связанных документов;
- **ID** – ИД документа;
- **OpenMode** – режим открытия документа: просмотр или редактирование;
- **ShowNamesOfEmptyRequisitesInEDocsText** – признак отображения в полях документа названий реквизитов вместо их значений для незаполненных реквизитов;
- **Version** – номер версии документа;
- **VersionsCount** – количество версий документа;
- **UserCanModify** – признак того, что текущий пользователь имеет права на изменение документа;
- **UserCanManage** – признак того, что текущий пользователь может изменять права доступа на документ.

Функция возвращает значения реквизитов в формате:

```
<DocumentInfo>
  <!-- Реквизиты документа. -->
  <Name1>Value1</Name1>
  …
</DocumentInfo>
```

Рекомендуется использовать данную функцию, чтобы отобразить пользователю список подписей документа и отчет по подписям.

Пример:

```
' Отобразить список подписей документа, если документ подписан.
Set XML = CreateObject("MSXML.DOMDocument")
Call XML.loadXML(IntLib.GetEDocumentInfo(ActiveDocument.Name))
DocSigned = XML.getElementsByTagName( _
  "CurrentVersionSigned").nextNode.nodeValue
If DocSigned = "Yes" Then
  Call MsgBox(IntLib.GetEDocumentInfo(ActiveDocument.Name))
End If
```

GetEDocumentPredefinedRequisitesList – получить значения предопределенных реквизитов документа

Синтаксис:

```
function GetEDocumentPredefinedRequisitesList(
  const FullDocumentName: WideString): WideString;
```

Параметры:

FullDocumentName – имя файла активного документа приложения с указанием полного пути.

Возвращаемое значение:

Строка, содержащая значения некоторых предопределенных реквизитов документа.

Описание:

Функция возвращает строку в формате XML со значениями некоторых реквизитов документа. Если указанный документ не является документом системы DIRECTUM, то функция возвращает пустую строку.

Функция возвращает значения реквизитов документа и его открытой версии:

- **Наименование**;
- **Дата документа**;
- **№ документа**;
- **ИД**;
- **Номер версии**.

Функция возвращает значения реквизитов в формате:

```
<Requisites>
  <!-- Реквизит1. -->
  <Requisite
    Name="{Название реквизита1}"
    LocalizedName="{ИД строки локализации реквизита1}">
    <![CDATA[{Значение реквизита1}]
  </Requisite>
  …
</Requisites>
```

Пример:

```
' Отобразить информацию о документе.
Set XMLDoc = CreateObject("MSXML.DOMDocument")
XMLData = IntLib.GetEDocumentPredefinedRequisitesList( _
  AtiveDocument.FullName)
Call XMLDoc.loadXML(XMLData)
Set NodeList = XMLDoc.selectNodes("//Requisite")
ReDim ReqList(NodeList.length - 1, 1)
NodeIndex = 0
For Each Node In NodeList
  MsgBox Node.attributes.getNamedItem("Name").nodeValue _
    & " = " & Node.Text
  NodeIndex = NodeIndex + 1
Next
```

GetEDocumentRequisitesList – получить значения всех реквизитов документа

Синтаксис:

```
function GetEDocumentRequisitesList(
  DocumentName: WideString): WideString;
```

Параметры:

DocumentName – имя файла активного документа приложения без указания пути.

Возвращаемое значение:

Строка, содержащая значения реквизитов документа.

Описание:

Функция возвращает строку в формате XML со значениями всех реквизитов главного раздела документа. Если указанный документ не является документом системы DIRECTUM, то функция возвращает пустую строку.

Функция возвращает значения реквизитов в формате:

```
<Requisites>
  <!-- Реквизит1. -->
  <Requisite
    Name="{Имя реквизита1}"
    LocalizedName="{ИД локализации реквизита1}">
      {Значение реквизита1}
  </Requisite>
  …
</Requisites>
```

Рекомендуется использовать функцию:

- для заполнения полей документа, содержащих значения реквизитов документа;
- для передачи списка реквизитов в функцию **ShowEDocumentRequisitesListDialog()**.

Пример:

```
' Отобразить список реквизитов.
Dim Requisite, Reference As String
ReqList = IntLib.GetEDocumentRequisitesList(ActiveDocument.Name)
Call IntLib.ShowEDocumentRequisitesListDialog( _
  ReqList, Requisite, Reference)
```

GetFilePathInTempFolder – получить полный путь к файлу во временной папке IS-Builder

Синтаксис:

```
function GetFilePathInTempFolder(
  DocumentName: WideString): WideString;
```

Параметры:

DocumentName – имя файла активного документа приложения без указания пути.

Возвращаемое значение:

Путь к временному файлу, создаваемому одной из библиотек интеграции и сохраняемому во временной папке IS-Builder. Например, для Windows Server 2012 путь будет иметь формат: C:\Users\<Имя пользователя Windows>\AppData\Local\Temp\<Имя сервера DIRECTUM>\<Название базы DIRECTUM>.

Если операция получения пути к временному файлу в папке IS-Bulder прошла с ошибкой, то функция возвращает путь к файлу во временной папке текущего пользователя. Например, для Windows Server 2012: C:\Users\<Имя пользователя Windows>\AppData\Local\Temp. К имени файла добавляется префикс «Temp_». Если файл с таким именем уже существует в указанной папке, то к имени файла перед расширением добавляются символы «_I», где I – номер дубля временного файла минус один.

Описание:

Функция используется при необходимости получения пути к файлу во временной папке IS-Builder.

Рекомендуется использовать при необходимости сохранения временных файлов офисных приложений на локальный диск. Например, импорт или экспорт документов, переоткрытие.

Пример:

```
' Получить имя файла во временной папке IS-Builder.
SavedDocumentName = IntLib.GetFilePathInTempFolder(Document.Name)
'Удалить файл, если он уже есть.
System.IO.File.Delete(SavedDocumentName)
' Сохранить документ во временную папку.
Document.SaveAs(FileName:=SavedDocumentName)
```

GetHyperlinkTitle – получить заголовок гиперссылки на объект DIRECTUM

Синтаксис:

```
function GetHyperlinkTitle(
  const HyperlinkAddress: WideString): WideString;
```

Параметры:

HyperlinkAddress – адрес гиперссылки.

Возвращаемое значение:

Заголовок гиперссылки на объект системы DIRECTUM.

Описание:

Функция возвращает заголовок заданной ссылки на объект системы DIRECTUM.

Пример:

```
Dim HyperlinkTitle As String
' Получить эталонный заголовок гиперссылки.
HyperlinkTitle = IntLib.GetHyperlinkTitle( _
  Application.Selection.Hyperlinks.Item(1).Address)
' Вернуть эталонный заголовок гиперссылке, если он изменился.
If (HyperlinkTitle <> "") And _
  (Application.Selection.Hyperlinks.Item(1).TextToDisplay <> _
  HyperlinkTitle) Then
  Application.Selection.Hyperlinks.Item(1).TextToDisplay = HyperlinkTitle
End If
```

GetLocalizedString – получить значение строки локализации

Синтаксис:

```
function GetLocalizedString(
  Code: WideString;
  GroupCode: WideString): WideString;
```

Параметры:

- **Code** – код строки в компоненте **Словарь локализации**;
- **GroupCode** – код группы локализации в компоненте **Словарь локализации**.

Возвращаемое значение:

Значение строки локализации на текущем языке системы DIRECTUM.

Описание:

Функция возвращает значение строки локализации с кодом **Code** из группы строк с кодом **GroupCode** на текущем языке системы DIRECTUM. Если строка локализации не найдена, то функция возвращает строку в формате:

<Код группы локализации>.<Код строки локализации>.

Пример:

```
' Получить значение строки локализации.
LocalizedString = IntLib.GetLocalizedString("STR_ERROR_SENDING_EMAIL", _
  "DIRRES_OFFICEINTEGRATION")
```

GetSystemInfo – получить информацию о системе DIRECTUM

Синтаксис:

```
function GetSystemInfo: WideString;
```

Возвращаемое значение:

Строка, содержащая информацию о системе и о текущем пользователе.

Описание:

Примечание

Функция оставлена для совместимости с интеграцией с OpenOffice. Для получения информации о системе DIRECTUM рекомендуется использовать функцию **GetSystemInfo2()**.

Функция возвращает информацию о системе:

- **Code** – код системы;
- **InstallationID** – ИД установки системы;
- **CurrentUser** – группа свойств текущего пользователя. Включает в себя элемент **HasCertificatesForSigning** – наличие сертификатов для подписания документов.

Функция возвращает информацию о системе в формате:

```
<SystemInfo>
  <!-- Свойства системы. -->
  <Name1>Value1</Name1>
  …
  <!-- Группа свойств. -->
  <PropertyGroup1>
    <!-- Свойства системы, относящиеся к группе. -->
    <Name1>Value1<Name1>
    …
  </PropertyGroup1>
```

```
  …
</SystemInfo>
```

Рекомендуется использовать функцию для проверки наличия у текущего пользователя хотя бы одного сертификата для подписания документов.

Пример:

```
' Проверить наличие сертификата у пользователя.
Set XML = CreateObject("MSXML.DOMDocument")
Call XML.loadXML(IntLib.GetSystemInfo)
UserCanSign = XML.getElementsByTagName( _
  "HasCertificatesForSigning").nextNode.nodeValue
If UserCanSign = "Yes" Then
  …
End If
' Отобразить информацию о системе.
Call MsgBox(IntLib.GetSystemInfo)
```

GetSystemInfo2 – получить информацию о текущей системе DIRECTUM

Синтаксис:

```
function GetSystemInfo2: WideString;
```

Возвращаемое значение:

Строка, содержащая информацию о системе.

Описание:

Функция возвращает информацию о системе:

- **Code** – код системы;
- **InstallationID** – ИД установки системы;
- **Language** – код языка системы.

Функция возвращает информацию о системе в формате:

```
<SystemInfo>
  <!—- Свойства системы. -->
  <Name1>Value1</Name1>
  …
</SystemInfo>
```

Пример:

```
' Отобразить ИД инсталляции.
Set XML = CreateObject("MSXML.DOMDocument")
Call XML.loadXML(IntLib.GetSystemInfo)
InstallationID = XML.getElementsByTagName( _
  "InstallationID").nextNode.nodeValue
MsgBox(InstallationID)
```

ImportFileIntoEDocumentNewVersion – импортировать файл в новую версию документа

Синтаксис:

```
function ImportFileIntoEDocumentNewVersion(
  DocumentName: WideString,
  SavedFileName: WideString,
  out NeedToOpen: WordBool): WordBool;
```

Параметры:

- **DocumentName** – имя файла активного документа приложения без указания пути;
- **SavedFileName** – полный путь к файлу, импортируемому в новую версию;
- **NeedToOpen** – признак необходимости открытия новой версии документа. Возможные значения: **True**, если нужно открыть, иначе **False**.

Возвращаемое значение:

True, если файл был успешно импортирован в новую версию документа, **False**, если во время импорта было сгенерировано исключение.

Описание:

Функция используется для реализации импорта документа в новую версию, например при сохранении изменений документа в его новую версию.

Пример:

```
If IntLib.ImportFileIntoEDocumentNewVersion(DocumentName,
   FileName, NeedToOpenDocument) = True Then
  Call MsgBox("Файл успешно импортирован в новую версию")
End If
```

ImportFileIntoEDocumentVersion – импортировать версию документа из файла

Синтаксис:

```
function ImportFileIntoEDocumentVersion(
  DocumentName: WideString;
  DocumentVersionNumber: Integer;
  ImportedFileName: WideString): WordBool;
```

Входные параметры:

- **DocumentName** – имя файла активного документа приложения без указания пути;
- **DocumentVersionNumber** – номер импортируемой версии документа;
- **ImportedFileName** – имя файла, который будет импортирован в указанную версию.

Возвращаемое значение:

True, если файл был успешно импортирован в версию документа, **False**, если во время импорта было сгенерировано исключение.

Описание:

Функция импортирует версию документа из файла **ImportedFileName**. После импорта файл на диске удаляется.

Если не удается импортировать документ, то будет выдано сообщение об ошибке.

Пример:

```
' Импортировать документ.
' DocumentName - имя активного документа.
' DocumentVersionNumber - номер версии документа.
' ExportedFileName - имя экспортируемого документа.
If IntLib.ImportFileIntoEDocumentVersion(DocumentName,
    DocumentVersionNumber, ExportedFileName) = True Then
  MsgBox("Импорт файла прошел успешно")
End If
```

OpenEDocument – открыть документ из системы DIRECTUM

Синтаксис:

```
procedure OpenEDocument(
  const DocumentName: WideString;
  OpenForWrite: WordBool);
```

Параметры:

- **DocumentName** – строка в формате «(<ИД документа>v<Номер версии документа>)»;
- **OpenForWrite** – признак открытия документа для редактирования. Возможные значения: **True**, открыть документ для редактирования, **False**, открыть документ для просмотра.

Описание:

Функция открывает документ с именем **DocumentName**. Номер открываемой версии извлекается из имени документа.

Пример:

```
' Открыть документ с именем DocumentName.
Call IntLib.OpenEDocument(DocumentName, True)
```

OpenEDocumentByID – открыть документ по его ИД

Синтаксис:

```
procedure OpenEDocumentByID(
  DocumentID: Integer;
  OpenForWrite: WordBool);
```

Параметры:

- **DocumentID** – ИД документ;
- **OpenForWrite** – признак открытия документа для редактирования.

Описание:

Функция открывает заданный документ. Если у документа более одной версии, отображается диалог выбора открываемой версии. В зависимости от значения параметра **OpenForWrite** документ открывается для редактирования или для просмотра.

Пример:

```
' Открыть документ с ИД 12345 для редактирования.
Call IntLib.OpenEDocumentByID(12345, True)
```

OpenFolder – открыть папку

Синтаксис:

```
procedure OpenFolder(
  FolderID: Integer);
```

Параметры:

FolderID – ИД папки.

Описание:

Функция отображает содержимое указанной папки текущей системы DIRECTUM.

Пример:

```
' Открыть папку.
Call IntLib.OpenFolder(12345)
```

OpenLastEDocumentVersionByID – открыть последнюю версию документа

Синтаксис:

```
procedure OpenLastEDocumentVersionByID(
  DocumentID: Integer;
  OpenForWrite: WordBool);
```

Параметры:

- **DocumentID** – ИД документ;
- **OpenForWrite** – признак открытия документа для редактирования.

Описание:

Функция открывает последнюю версию заданного документа. В зависимости от значения параметра **OpenForWrite** документ открывается для редактирования или для просмотра.

Пример:

```
' Открыть документ с ИД 12345 для редактирования.
Call IntLib.OpenLastEDocumentVersionByID(12345, True)
```

ParseFieldCode – извлечь из кода поля данные о реквизите

Синтаксис:

```
procedure ParseFieldCode(
  const FieldCode: WideString;
  var DocumentRequisiteTitle,
  RequisiteRequisiteTitle: WideString);
```

Параметры:

- **FieldCode** – код поля;
- **DocumentRequisiteTitle** – заголовок реквизита карточки документа, который был вставлен в документ;
- **RequisiteRequisiteTitle** – заголовок реквизита справочника, с которым связан реквизит **DocumentRequisiteTitle**, если он имеет тип «Справочник».

Описание:

Функция извлекает из кода поля сведения о реквизите, значение которого является значением поля. Результат возвращается в параметрах **DocumentRequisiteTitle** и **RequisiteRequisiteTitle**.

Значение параметра **FieldCode** должно иметь формат: «<Необязательные символы>P*<Заголовок реквизита карточки>…*<Заголовок реквизита справочника><Необязательные символы>». В параметре **DocumentRequisiteTitle** возвращается «Заголовок реквизита карточки». В параметре **RequisiteRequisiteTitle** возвращается «Заголовок реквизита справочника».

Пример:

```
Call IntLib.ParseFieldCode( _
  Application.Selection.Fields(1).Code.Text, _
  DocReqTitle, ReqReqTitle)
```

ParseHyperlink – извлечь из гиперссылки данные об объекте DIRECTUM

Синтаксис:

```
procedure ParseHyperlink(
  const HyperlinkAddress: WideString;
  var ObjectID: Integer;
  var ObjectType, ObjectComponentCode: WideString);
```

Параметры:

- **HyperlinkAddress** – адрес гиперссылки;
- **ObjectID** – в этом параметре возвращается ИД объекта системы DIRECTUM;
- **ObjectType** – в этом параметре возвращается тип объекта системы DIRECTUM. Возможные значения:
  - **JOB** – задание;
  - **TASK** – задача;
  - **DOC** – документ;
  - **FOLDER** – папка;

  - **REFERENCE** – справочник;
  - **COMPONENT_TOKEN** – вариант запуска.
- **ObjectComponentCode** – в этом параметре возвращается код компоненты объекта DIRECTUM. Возвращается пустая строка, если для объекта нет компоненты.

Описание:

Из переданного адреса гиперссылки извлекается информация об ИД объекта системы DIRECTUM, его типе и коде компоненты. Извлеченная информация сохраняется в параметрах **ObjectID**, **ObjectType**, **ObjectComponentCode**.

Пример:

```
HyperlinkAddress = Appilication.Selection.Hyperlinks.Item(1).Address
' Если это гиперссылка DIRECTUM.
If IntLib.DirectumHyperlink(HyperlinkAddress) Then
  ' Получить ИД, тип объекта и код компоненты из гиперссылки.
  IntLib.ParseHyperlink(HyperlinkAddress, _
    HyperlinkObjectID, HyperlinkObjectType, _
    HyperlinkObjectComponentCode)
End If
```

RegisterDocument – зарегистрировать документ

Синтаксис:

```
procedure RegisterDocument(
  const DocumentName: WideString);
```

Параметры:

DocumentName – имя файла активного документа, без указания пути.

Описание:

Функция регистрирует документ в системе DIRECTUM. Вызов функции равносилен нажатию на кнопку **РКК** в карточке документа. Подробнее см. документацию системы DIRECTUM, руководство по модулю «Канцелярия».

Для работы данной функции в системе должна присутствовать константа **RegisterActionNames**, в которой указаны действия регистрации для различных типов карточек документов в формате: <Имя действия регистрации по умолчанию>|<Код типа карточки №1>|<Имя действия регистрации №1>|<Код типа карточки №2>|<Имя действия регистрации №2>|…

Если для типа карточки документа не найдено действие регистрации, то функция выдает сообщение об ошибке.

Пример:

```
' Зарегистрировать документ.
Call IntLib.RegistrateDocument(ActiveDocument.Name)
```

SaveAsEDocument – сохранить файл в системе DIRECTUM

Синтаксис:

```
procedure SaveAsEDocument(
  FullDocumentName: WideString;
  [optional] EDocumentTypeCode: WideString;
  [optional] EDocumentKindCode: WideString;
  [optional] EDocumentEditorCode: WideString;
  [optional] Requisites: WideString;
```

Параметры:

- **FullDocumentName** – полный путь к файлу, который нужно сохранить в системе DIRECTUM;
- **EDocumentTypeCode** – имя типа документа из справочника **Типы карточек документов**;
- **EDocumentKindCode** – код вида документа из справочника **Виды документов**;
- **EDocumentEditorCode** – код приложения-редактора из справочника **Приложения-редакторы**;
- **Requisites** – строка в формате XML, содержащая значения реквизитов, которыми необходимо заполнить поля документа.

Описание:

Функция создает документ в системе DIRECTUM из файла **FullDocumentName**. Если не указан один из опциональных параметров, функция отображает диалог создания документа из файла. Иначе документ создается в невизуальном режиме.

Параметр **Requisites** должен содержать значения всех обязательных реквизитов данного типа карточки документа.

На момент вызова функции файл, из которого создается документ, не должен быть занят другими приложениями.

Значение параметра **Requisites** следует передавать в формате:

```
<Dataset>
  <!-- Реквизит1 -->
  <Requisite
    Name="{Имя реквизита1}"
    [Type="{Тип реквизита1}"]>
      {Значение реквизита1}
  </Requisite>
  …
  <!-- Детальный раздел1 -->
  <DetailDataset>
    <!-- Реквизиты детального раздела -->
    <Requisite
      Name="{Имя реквизита1}"
      [Type="{Тип реквизита1}"]>
        {Значение реквизита1}
    </Requisite>
    …
  </DetailDataset>
  …
<Dataset/>
```

В качестве значения атрибута **Type** следует передавать одну из констант перечислимого типа **TReqDataType**. Подробнее см. документацию системы DIRECTUM, книга «Объектная модель IS-Builder». Атрибут **Type** можно не указывать. В этом случае проверка соответствия типов не выполняется.

Рекомендуется использовать функцию:

- для сохранения активного документа приложения в системе DIRECTUM. Предварительно документ нужно закрыть;
- для сохранения текста письма как документа системы DIRECTUM с автоматическим заполнением некоторых реквизитов.

Пример:

```
' Сохранить активный документ приложения в системе DIRECTUM.
DocumentFullName = ActiveDocument.FullName
Call Application.ActiveDocument.Close
Call IntLib.SaveAsEDocument(DocumentFullName)
```

SaveMailAsClientContact – сохранить электронное письмо как контакт с клиентом

Синтаксис:

```
procedure SaveMailAsClientContact(
  RecipientList: WideString;
  CreationDate: WideString;
  DateDespatched: WideString;
  FullDocumentName: WideString);
```

Параметры:

- **RecipientList** – список почтовых адресов, которые следует занести в DIRECTUM. Почтовые адреса в списке разделяются точкой с запятой «;»;
- **CreationDate** – дата создания письма;
- **DateDespatched** – дата отправки письма;
- **FullDocumentName** – полный путь до файла с текстом письма.

Описание:

Функция сохраняет письмо электронной почты как контакт с клиентом в системе DIRECTUM. На каждое письмо создается отдельная запись в справочнике **Контакты с клиентами**.

Если в адресной строке содержится несколько адресов, каждый адрес помещается в отдельную запись детального раздела «Контактные лица». Текст письма помещается в поле **Описание**.

Пример:

```
' Сохранить письмо как контакт с клиентом в системе DIRECTUM.
Set Mail = Application.ActiveInspector.CurrentItem
Call Mail.SaveAs(Path, olTXT)
Call IntLib.SaveMailAsClientContact(Mail.SenderEmailAddress, _
  Mail.CreationTime, Mail.SentOn, Path)
```

SelectDocumentVersionToCompare – отобразить диалог выбора версии документа, с которой будет сравниваться текущая версия

Синтаксис:

```
procedure SelectDocumentVersionToCompare(
  DocumentName: WideString;
  out DocumentVersionNumber: Integer);
```

Параметры:

- **DocumentName** – имя файла активного документа приложения без указания пути;
- **DocumentVersionNumber** – номер версии документа, с которой будет выполнено сравнение открытой версии.

Описание:

Процедура используется для выбора версии, с которой необходимо сравнить открытую версию документа. Причем если пользователь не выбрал версию для сравнения, параметр **DocumentVersionNumber** принимает значение **-1**. То же значение параметру присваивается в тех случаях, когда для сравнения пользователь выбрал версию другого типа или при выполнении процедуры было сгенерировано исключение.

Пример:

```
' Выбрать номер версии, с которой будем сравнивать текущую версию.
IntLib.SelectDocumentVersionToCompare(Doc.Name, VersionToCompareNumber)
' Если не возникло никаких ошибок при выборе версии.
If VersionToCompareNumber <> -1 Then
  ' Экспортировать версию документа для сравнения.
  ExportedVersionToCompareFilePath = IntLib.ExportEDocumentVersion(
  Doc.Name, VersionToCompareNumber)
End If
```

SendEDocumentAsAttachmentToTask – отправить документ вложением в задачу

Синтаксис:

```
procedure SendEDocumentAsAttachmentToTask(
  DocumentName: WideString);
```

Параметры:

DocumentName – имя файла активного документа приложения без указания пути.

Описание:

Функция создает задачу, вкладывает в нее ссылку на указанный документ и отображает карточку задачи.

Пример:

```
' Отправить документ вложением в задачу.
Call IntLib.SendEDocumentAsAttachmentToTask(ActiveDocument.Name)
```

SendEDocumentAsLinkInMail – отправить документ ссылкой в письме

Синтаксис:

```
procedure SendEDocumentAsLinkInMail(
  DocumentName: WideString);
```

Параметры:

DocumentName – имя файла активного документа приложения без указания пути.

Описание:

Функция создает электронное письмо, вкладывает в него ссылку на документ и отображает письмо.

Пример:

```
' Отправить документ ссылкой в письме.
Call IntLib.SendEDocumentAsLinkInMail(ActiveDocument.Name)
```

SendFolderAsAttachmentToTask – отправить папку вложением в задачу

Синтаксис:

```
procedure SendFolderAsAttachmentToTask(
  FolderID: Integer);
```

Параметры:

FolderID – ИД папки.

Описание:

Функция создает задачу, вкладывает в нее ссылку на указанную папку и отображает карточку задачи.

Пример:

```
' Отправить папку вложением в задачу.
Call IntLib.SendFolderAsAttachmentToTask(12345)
```

SendJobAsAttachmentToTask – отправить задание вложением в задачу

Синтаксис:

```
procedure SendJobAsAttachmentToTask(
  JobID: Integer);
```

Параметры:

JobID – ИД задания.

Описание:

Функция создает задачу, вкладывает в нее ссылку на указанное задание и отображает карточку задачи.

Пример:

```
' Отправить задание вложением в задачу.
Call IntLib.SendJobAsAttachmentToTask(12345)
```

SendReferenceRecordAsAttachmentToTask – отправить запись справочника вложением в задачу

Синтаксис:

```
procedure SendReferenceRecordAsAttachmentToTask(
  const ReferenceCode: WideString;
  RecordID: Integer);
```

Параметры:

- **ReferenceCode** – код справочника;
- **RecordID** – ИД записи справочника.

Описание:

Функция создает задачу, вкладывает в нее ссылку на указанную запись справочника и отображает карточку задачи.

Пример:

```
' Отправить запись справочника "Совещания" вложением в задачу.
Call IntLib.SendReferenceRecordAsAttachmentToTask("СВЩ", 12345)
```

SendTaskAsAttachmentToTask – отправить задачу вложением в задачу

Синтаксис:

```
procedure SendTaskAsAttachmentToTask(
  TaskID: Integer);
```

Параметры:

TaskID – ИД задачи.

Описание:

Функция создает задачу, вкладывает в нее ссылку на указанную задачу и отображает карточку задачи.

Пример:

```
' Отправить задачу вложением в задачу.
Call IntLib.SendTaskAsAttachmentToTask(12345)
```

SignEDocument – подписать документ

Синтаксис:

```
procedure SignEDocument(
  DocumentName: WideString;
  VersionID: Integer);
```

Параметры:

- **DocumentName** – имя файла активного документа приложения без указания пути;
- **VersionID** – значение поля **XRecID** таблицы **SBEDocVer** открытой версии документа.

Описание:

Функция вызывает диалог подписания документа. На момент вызова функции документ должен быть закрыт.

Пример:

```
' Вызвать диалог подписания текущего документа.
DocumentName = ActiveDocument.Name
Call ActiveDocument.Close
DocInfoXML = IntLib.GetEDocumentInfo(ActiveDocument.FullName)
Set XMLFile = CreateObject("MSXML.DOMDocument")
Call XMLFile.loadXML(DocInfoXML)
VersionID = XMLFile.selectSingleNode("//CurrentVersionID").Text
Call IntLib.SignEDocument(DocumentName, VersionID)
```

ShowBoundedEDocuments – отобразить список связанных документов

Синтаксис:

```
procedure ShowBoundedEDocuments(
  DocumentName: WideString);
```

Параметры:

DocumentName – имя файла активного документа приложения без указания пути.

Описание:

Функция отображает список документов, связанных с документом, указанным в параметре **DocumentName**.

Пример:

```
' Отобразить список связанных документов.
Call IntLib.ShowBoundedEDocuments(ActiveDocument.Name)
```

ShowBoundedTasks – отобразить список задач, в которые вложен документ

Синтаксис:

```
procedure ShowBoundedTasks(
  const DocumentName: WideString);
```

Параметры:

DocumentName – имя файла активного документа приложения без указания пути.

Описание:

Функция отображает список задач, в которые вложен документ, указанный в параметре **DocumentName**.

Пример:

```
' Отобразить список задач, в которые вложен документ.
Call IntLib.ShowBoundedTasks(ActiveDocument.Name)
```

ShowContactSearchingDialog – отобразить диалог поиска контактных лиц

Синтаксис:

```
procedure ShowContactSearchingDialog(
  SenderMail: WideString);
```

Параметры:

SenderMail – почтовый адрес.

Описание:

Функция отображает диалог поиска контактных лиц организаций. Искать контактные лица можно по организации и адресу электронной почты. После заполнений условий поиска открывается справочник **Контактные лица организаций**. При этом в справочнике показаны только те записи, которые удовлетворяют заданным условиям.

При создании новой записи в справочнике **Контактные лица организаций** поле **E-mail** будет заполняться значением, указанным в параметре **SenderMail**.

Пример:

```
' Отобразить диалог поиска контактных лиц.
Set Mail = Application.ActiveInspector.CurrentItem
Call IntLib.ShowContactSearchingDialog(Mail.To)
```

ShowCreateEDocumentFromTemplateDialog – отобразить диалог создания документа из шаблона

Синтаксис:

```
procedure ShowCreateEDocumentFromTemplateDialog;
```

Описание:

Функция отображает диалог создания документа из шаблона.

Пример:

```
' Отобразить диалог создания документа из шаблона.
Call IntLib.ShowCreateEDocumentFromTemplateDialog
```

ShowEDocumentAccessRightsDialog – показать диалог настройки прав документа

Синтаксис:

```
procedure ShowEDocumentAccessRightsDialog(
  DocumentID: Integer);
```

Параметры:

DocumentID – ИД документа.

Описание:

Функция отображает диалог настройки прав заданного документа.

Пример:

```
' Показать диалог настройки прав документа.
Call IntLib.ShowEDocumentAccessRightsDialog(12345)
```

ShowEDocumentBoundedFolders – показать список папок, в которых находится документ

Синтаксис:

```
procedure ShowEDocumentBoundedFolders(
  DocumentID: Integer);
```

Параметры:

DocumentID – ИД документа.

Описание:

Функция отображает список папок, в которых находится заданный документ.

Пример:

```
' Показать список папок, в которых находится документ.
Call IntLib.ShowEDocumentBoundedFolders(12345)
```

ShowEDocumentCard – показать карточку документа

Синтаксис:

```
procedure ShowEDocumentCard(
  DocumentID: Integer);
```

Параметры:

DocumentID – ИД документа.

Описание:

Функция отображает карточку заданного документа.

Пример:

```
' Показать карточку документа.
Call IntLib.ShowEDocumentCard(12345)
```

ShowEDocumentHistory – отобразить историю документа

Синтаксис:

```
procedure ShowEDocumentHistory(
  DocumentName: WideString);
```

Параметры:

DocumentName – имя файла активного документа приложения без указания пути.

Описание:

Функция отображает историю работы с документом.

Пример:

```
' Отобразить историю документа.
Call IntLib.ShowEDocumentHistory(Application.ActiveDocument.Name)
```

ShowEDocumentRequisitesListDialog – отобразить диалог выбора реквизита документа

Синтаксис:

```
procedure ShowEDocumentRequisitesListDialog(
  Requisites: WideString;
  out RequisiteName: WideString;
  out ReferenceName: WideString);
```

Параметры:

- **Requisites** – строка, содержащая реквизиты. В качестве значения параметра следует передавать результат выполнения функции **GetEDocumentRequisitesList()**;
- **RequisiteName** – имя выбранного реквизита. Если реквизит не был выбран, то пустая строка;
- **ReferenceName** – имя справочника реквизита типа «Справочник». Если выбранный реквизит не является реквизитом типа «Справочник» или если реквизит не был выбран, то пустая строка.

Описание:

Функция отображает диалог выбора реквизита из списка реквизитов документа, переданного в параметре **Requisites**.

Рекомендуется использовать функцию для выбора реквизита для последующей вставки его значения в текст документа.

Пример:

```
' Отобразить список реквизитов.
Dim Requisite, Reference As String
ReqList = IntLib.GetEDocumentRequisitesList(ActiveDocument.Name)
Call IntLib.ShowEDocumentRequisitesListDialog( _
  ReqList, Requisite, Reference)
```

ShowEDocumentSignaturesInfo – отобразить информацию о подписях документа

Синтаксис:

```
procedure ShowEDocumentSignaturesInfo(
  DocumentName: WideString;
  MakeReport: WordBool);
```

Параметры:

- **DocumentName** – имя файла активного документа приложения без указания пути;
- **MakeReport** – признак формирования отчета в документе. Возможные значения: **True** – в документе будет сформирован отчет по подписям текущей версии документа, **False** – будет выведен стандартный диалог со списком подписей.

Описание:

Функция отображает информацию о подписях текущей версии документа. В зависимости от значения параметра **MakeReport** информация будет отображаться в виде стандартного диалога или в виде отчета в документе.

Пример:

```
' Отобразить отчет по подписям текущей версии документа.
Call IntLib.ShowEDocumentSignaturesInfo(ActiveDocument.Name, True)
```

ShowEDocumentSignaturesInfoByID – показать список подписей документа

Синтаксис:

```
procedure ShowEDocumentSignaturesInfoByID(
  DocumentID: Integer);
```

Параметры:

DocumentID – ИД документа.

Описание:

Функция отображает список подписей документа.

Пример:

```
' Показать список подписей документа с ИД 12345.
Call IntLib.ShowEDocumentSignaturesInfoByID(12345)
```

ShowFolderAccessRightsDialog – показать окно настройки прав папки

Синтаксис:

```
procedure ShowFolderAccessRightsDialog(
  FolderID: Integer);
```

Параметры:

FolderID – ИД папки.

Описание:

Функция отображает диалог настройки прав папки.

Пример:

```
' Показать диалог настройки прав папки.
Call IntLib.ShowFolderAccessRightsDialog(12345)
```

ShowFolderBoundedFolders – показать список папок, в которых находится указанная папка

Синтаксис:

```
procedure ShowFolderBoundedFolders(
  FolderID: Integer);
```

Параметры:

FolderID – ИД папки.

Описание:

Функция отображает список папок, в которых находится заданная папка.

Пример:

```
' Показать список папок, в которых находится указанная папка.
Call IntLib.ShowFolderBoundedFolders(12345)
```

ShowFolderBoundedTasks – показать список задач, в которые вложена указанная папка

Синтаксис:

```
procedure ShowFolderBoundedTasks(
  FolderID: Integer);
```

Параметры:

FolderID – ИД папки.

Описание:

Функция отображает список задач, в которые вложена заданная папка.

Пример:

```
' Показать список задач, в которые вложена указанная папка.
Call IntLib.ShowFolderBoundedTasks(12345)
```

ShowFolderCard – показать карточку папки

Синтаксис:

```
procedure ShowFolderCard(
  FolderID: Integer);
```

Параметры:

FolderID – ИД папки.

Описание:

Функция отображает карточку указанной папки.

Пример:

```
' Показать карточку папки.
Call IntLib.ShowFolderCard(12345)
```

ShowFolderHistory – показать историю папки

Синтаксис:

```
procedure ShowFolderHistory(
  FolderID: Integer);
```

Параметры:

FolderID – ИД папки.

Описание:

Функция отображает окно истории для указанной папки.

Пример:

```
' Показать окно истории папки с ИД 12345.
Call IntLib.ShowFolderHistory(12345)
```

ShowJobCard – показать карточку задания

Синтаксис:

```
procedure ShowJobCard(
  JobID: Integer);
```

Параметры:

JobID – ИД задания.

Описание:

Функция отображает карточку задания, указанного в параметре **JobID**.

Пример:

```
' Показать карточку задания с ИД 12345.
Call IntLib.ShowJobCard(12345)
```

ShowJobHistory – показать историю задания

Синтаксис:

```
procedure ShowJobHistory(
  JobID: Integer);
```

Параметры:

JobID – ИД задания.

Описание:

Функция отображает окно истории для указанного задания.

Пример:

```
' Показать окно истории задания с ИД 12345.
Call IntLib.ShowJobHistory(12345)
```

ShowJobNotifications – показать напоминания для задания

Синтаксис:

```
procedure ShowJobNotifications(
  JobID: Integer);
```

Параметры:

JobID – ИД задания.

Описание:

Функция отображает окно напоминаний для задания, указанного в параметре **JobID**.

Пример:

```
' Показать напоминания для задания с ИД 12345.
Call IntLib.ShowJobNotifications(12345)
```

ShowTaskTreeForJob – показать дерево задач для задания

Синтаксис:

```
procedure ShowTaskTreeForJob(
  JobID: Integer);
```

Параметры:

JobID – ИД задания.

Описание:

Функция отображает дерево задач для задания, указанного в параметре **JobID**.

Пример:

```
' Показать дерево задач задания с ИД 12345.
Call IntLib.ShowTaskTreeForJob(12345)
```

ShowReferenceListForm – показать форму-список справочника

Синтаксис:

```
procedure ShowReferenceListForm(
  const ReferenceCode: WideString;
  RecordID: Integer);
```

Параметры:

- **ReferenceCode** – код справочника;
- **RecordID** – ИД записи справочника.

Описание:

Функция отображает форму-список заданного справочника, курсор при этом перемещается к записи с указанным ИД.

Пример:

```
' Показать форму-список справочника "Совещания".
Call IntLib.ShowReferenceListForm("СВЩ", 12345)
```

ShowReferenceRecordBoundedDocuments – показать список документов, связанных с указанной записью справочника

Синтаксис:

```
procedure ShowReferenceRecordBoundedDocuments(
  const ReferenceCode: WideString;
  RecordID: Integer);
```

Параметры:

- **ReferenceCode** – код справочника;
- **RecordID** – ИД записи справочника.

Описание:

Функция отображает список документов, с которыми связана запись справочника, указанная в параметре **RecordID**.

Пример:

```
' Показать список документов, связанных с указанной записью справочника.
Call IntLib.ShowReferenceRecordBoundedDocuments("СВЩ", 12345)
```

ShowReferenceByFieldCode – показать форму-список справочника, значение записи которого совпадает со значением указанного поля

Синтаксис:

```
procedure ShowReferenceByFieldCode(
  DocumentID: Integer;
  const FieldCode: WideString);
```

Параметры:

- **DocumentID** – ИД документа;
- **FieldCode** – код поля.

Описание:

Функция отображает форму-список справочника, значение записи которого совпадает со значением указанного поля.

Пример:

```
' Показать форму-список справочника, значение реквизита которого совпадает со
' значением указанного поля.
Call IntLib.ShowReferenceByFieldCode(12345, _
  Application.Selection.Fields(1).Code.Text)
```

ShowReferenceRecordByFieldCode – показать карточку записи справочника, которая имеет значение реквизита, совпадающего со значением указанного поля

Синтаксис:

```
procedure ShowReferenceRecordByFieldCode(
  DocumentID: Integer;
  const FieldCode: WideString);
```

Параметры:

- **DocumentID** – ИД документа;
- **FieldCode** – код поля справочника.

Описание:

Функция отображает карточку записи справочника, которая имеет значение реквизита, совпадающего со значением указанного поля справочника.

Пример:

```
' Показать карточку, значение реквизита которой совпадает со значением поля.
Call IntLib.ShowReferenceRecordByFieldCode(12345, _
  Application.Selection.Fields(1).Code.Text)
```

ShowReferenceRecordBoundedTasks – показать список задач, в которые вложена указанная запись справочника

Синтаксис:

```
procedure ShowReferenceRecordBoundedTasks(
  const ReferenceCode: WideString;
  RecordID: Integer);
```

Параметры:

- **ReferenceCode** – код справочника;
- **RecordID** – ИД записи справочника.

Описание:

Функция отображает список задач, в которые вложена запись справочника, указанная в параметре **RecordID**.

Пример:

```
' Показать список задач, в которые вложена указанная запись справочника.
Call IntLib.ShowReferenceRecordBoundedTasks("СВЩ", 12345)
```

ShowReferenceRecordCard – показать карточку записи справочника

Синтаксис:

```
procedure ShowReferenceRecordCard(
  const ReferenceCode: WideString;
  RecordID: Integer);
```

Параметры:

- **ReferenceCode** – код справочника;
- **RecordID** – ИД записи справочника.

Описание:

Функция отображает карточку указанной записи справочника.

Пример:

```
' Показать карточку записи справочника "Совещания".
Call IntLib.ShowReferenceRecordCard("СВЩ", 12345)
```

ShowReferenceRecordHistory – показать историю записи справочника

Синтаксис:

```
procedure ShowReferenceRecordHistory(
  const ReferenceCode: WideString;
  RecordID: Integer);
```

Параметры:

- **ReferenceCode** – код справочника;
- **RecordID** – ИД записи справочника.

Описание:

Функция отображает окно истории заданной записи справочника.

Пример:

```
' Показать историю записи справочника "Совещания".
Call IntLib.ShowReferenceRecordHistory("СВЩ", 12345)
```

ShowReferenceRecordProperties – показать свойства записи справочника

Синтаксис:

```
procedure ShowReferenceRecordProperties(
  const ReferenceCode: WideString;
  RecordID: Integer);
```

Параметры:

- **ReferenceCode** – код справочника;
- **RecordID** – ИД записи справочника.

Описание:

Функция отображает окно свойств заданной записи справочника.

Пример:

```
' Показать свойства записи справочника "Совещания".
Call IntLib.ShowReferenceRecordProperties("СВЩ", 12345)
```

ShowSelectRecipientsDialog – отобразить диалог выбора получателей письма из контактных лиц организаций

Синтаксис:

```
procedure ShowSelectRecipientsDialog(
  out RecipientsListTo: WideString;
  out RecipientsListCopy: WideString);
```

Параметры:

- **RecipientsListTo** – список почтовых адресов получателей письма;
- **RecipientsListCopy** – список почтовых адресов получателей копии письма.

Описание:

Функция вызывает диалог выбора получателей письма из справочника **Контактные лица организаций**.

Рекомендуется использовать функцию для реализации выбора адресов в почтовом клиенте при отправке письма.

Пример:

```
' Отобразить диалог выбора получаетелей письма.
Dim RecipientsListTo, RecipientsListCopy
Call IntLib.ShowSelectRecipientsDialog( _
  RecipientsListTo, RecipientsListCopy)
Set Mail = Application.ActiveInspector.CurrentItem
Mail.To = RecipientsListTo
Mail.CC = RecipientsListCopy
```

ShowTaskCard – показать карточку задачи

Синтаксис:

```
procedure ShowTaskCard(
  TaskID: Integer);
```

Параметры:

TaskID – ИД задачи.

Описание:

Функция открывает карточку указанной задачи.

Пример:

```
' Показать карточку задачи с ИД 12345.
Call IntLib.ShowTaskCard(12345)
```

ShowTaskHistory – показать историю задачи

Синтаксис:

```
Procedure ShowTaskHistory(
  TaskID: Integer);
```

Параметры:

TaskID – ИД задачи.

Описание:

Функция отображает окно истории для задачи, указанной в параметре **TaskID**.

Пример:

```
' Показать окно истории задачи с ИД 12345.
Call IntLib.ShowTaskHistory(12345)
```

ShowTaskNotifications – показать напоминания для задачи

Синтаксис:

```
procedure ShowTaskNotifications(
  TaskID: Integer);
```

Параметры:

TaskID – ИД задачи.

Описание:

Функция отображает окно напоминаний для указанной задачи.

Пример:

```
' Показать напоминания для задачи с ИД 12345.
Call IntLib.ShowTaskNotifications(12345)
```

ShowTaskTreeForTask – показать дерево задач для задачи

Синтаксис:

```
procedure ShowTaskTreeForTask(
  TaskID: Integer);
```

Параметры:

TaskID – ИД задачи.

Описание:

Функция отображает дерево задач для задачи, указанной в параметре **TaskID**.

Пример:

```
' Показать дерево задач задачи с ИД 12345.
Call IntLib.ShowTaskTreeForTask(12345)
```

WaitForEDocumentVersionUnlock – показать форму ожидания разблокировки версии документа

Синтаксис:

```
procedure WaitForEDocumentVersionUnlock(
  VersionID: Integer)
```

Параметры:

VersionID – идентификатор открытой версии документа. В качестве значения параметра следует передавать значение поля **XRecID** таблицы **SBEDocVer**.

Описание:

Функция проверяет, заблокирована ли открытая версия документа одним из пользователей системы, например открыта на редактирование. Также проверяет необходимость ожидания разблокировки. Если необходимо подождать разблокировку версии, то появляется соответствующее окно.

Пример:

```
' Подождать освобождения версии документа.
Call IntLib.WaitForEDocumentVersionUnlock(DocumentVersionID)
' Импортировать документ.
If IntLib.ImportFileIntoEDocumentVersion(DocumentName,
   DocumentVersionNumber, ExportedFileName) = True Then
  MsgBox("Импорт файла прошел успешно")
End If
```

IsDocumentModel – определить, является ли тип карточки макетом документов

Синтаксис:

```
function IsDocumentModel(
  const EDocCardTypeID: Integer): WordBool;
```

Параметры:

EDocCardTypeID – идентификатор типа карточки документа.

Возвращаемое значение:

True, если идентификатор переданного типа карточки является идентификатором типа карточки «Макеты документов», иначе **False**.

Описание:

Функция проверяет, является ли идентификатор переданного типа карточки документов идентификатором типа карточки «Макеты документов» или нет.

Пример:

```
' Проверить, является ли документ макетом.
DocumentIsModel = IntLib.IsDocumentModel(EDocCardTypeID)
```

GetDocumentNameByID – получить наименование документа по его идентификатору

Синтаксис:

```
function GetDocumentNameByID(
  const DocumentID: Integer): WideString;
```

Параметры:

DocumentID – идентификатор документа системы DIRECTUM.

Возвращаемое значение:

Наименование документа системы DIRECTUM.

Описание:

Функция по идентификатору находит документ в системе DIRECTUM и возвращает его наименование.

Пример:

```
' Получить наименование документа по его ИД.
DocumentName = IntLib.GetDocumentNameByID(DocumentID)
```

FindEDocumentByID – найти документ по его идентификатору

Синтаксис:

```
procedure FindEDocumentByID(
  DocumentID: Integer)
```

Параметры:

DocumentID – идентификатор документа системы DIRECTUM.

Описание:

Функция по идентификатору находит документ в системе DIRECTUM. В результате поиска открывается окно, в котором расположена ссылка на искомый документ.

Пример:

```
' Открыть документ по его ИД.
IntLib.FindEDocumentByID(DocumentID)
```

GenerateDocument – сгенерировать документ с помощью конструктора документов

Синтаксис:

```
procedure GenerateDocument(
  const InputFilePath: WideString;
  const OutputFilePath: WideString;
  const MacroSubstitutionNames: WideString;
  const MacroSubstitutionValues: WideString)
```

Параметры:

- **InputFilePath** – путь до входного файла (с неозначенными макроподстановками);
- **OutputFilePath** – путь до выходного файла (с означенными макроподстановками);
- **MacroSubstitutionNames** – строка, содержащая наименования означиваемых макроподстановок. Наименования разделяются символами «|||»;
- **MacroSubstitutionValues** – строка, содержащая значения макроподстановок. Значения разделяются символами «|||». Значения могут быть и многострочными, в этом случае отдельные строки внутри значения разделяются символом «|».

Описание:

Процедура предназначена для генерации документа на основе тестовых данных из макета документа при его подготовке с помощью конструктора документов.

Пример:

```
' Сгенерировать документ.
IntLib.GenerateDocument(InputFilePath, OutputFilePath,
  MacroSubstitutionNames, MacroSubstitutionValues)
```

FindDocumentInSystem – найти документ в системе

Синтаксис:

```
function FindDocumentInSystem(): WideString;
```

Возвращаемое значение:

Строка, содержащая полный путь к документу.

Описание:

Функция запускает поиск документов в системе DIRECTUM и экспортирует выбранный документ во временную папку %TEMP% на локальный компьютер. Если у документа несколько версий, то откроется диалоговое окно «Выбор версии документа». Если документ зашифрован паролем, откроется диалоговое окно для ввода пароля.

Пример:

```
' Запустить поиск документа в системе DIRECTUM и получить путь до
' экспортированного файла.
ExportedFileName = IntLib.FindDocumentInSystem
' Если путь до файла получен, откроется документ с помощью
' приложения Microsoft Word.
If ExportedFileName <> "" Then
  Set Word = CreateObject("Word.Application")
  Word.Open(ExportedFileName)
End If
```

ImportReportFromBusinessStudio – импортировать отчет Business Studio в DIRECTUM

Синтаксис:

```
Procedure ImportReportFromBusinessStudio (
const FileName : WideString;
const ObjectID: WideString;
const TemplateID: WideString;
const DocumentName: WideString;
const SendToAgreement: WordBool;
const CreateNewVersion: WordBool);
```

Параметры:

- **FileName** – имя файла, в который сохраняется отчет;
- **ObjectID** – ИД объекта системы Business Studio, по которому сформирован отчет;
- **TemplateID** – ИД шаблона отчета системы Business Studio;
- **DocumentName** – имя документа, который будет создан в системе DIRECTUM;
- **SendToAgreement** – признак отправки документа на согласование после создания. Возможные значения: **True** – документ будет отправлен на согласование, **False** – не будет отправлен. Значение по умолчанию **False**;
- **CreateNewVersion** – признак создания новой версии документа, если в системе DIRECTUM уже существует документ, соответствующий отчету системы Business Studio. Возможные значения: **True** – будет создана новая версия документа, **False** – новая версия не будет создана.

Описание:

Функция выполняет поиск записи в системном справочнике **Связи** по значениям параметров **ObjectID** и **TemplateID**. Реквизит **ExternalID** записи справочника должен быть равен значению в формате «<ObjectID>_<TemplateID>».

Поиск по записям справочника **Связи** может иметь результаты:

- если запись найдена, то на основании реквизита **DestID** формируется ИД документа. Если значение параметра **CreateNewVersion** равно **True**, содержимое файла, указанного в параметре **FileName**, импортируется в новую версию данного документа, если **False** – не импортируется;
- если запись не найдена, новый документ создается импортом из файла, указанного в параметре **FileName**. Новый документ будет иметь реквизиты:
    - наименование документа равно значению параметра **DocumentName**. Если параметр **DocumentName** не указан, наименование совпадает с наименованием файла, указанного в параметре **FileName**;
    - тип карточки документа равен значению константы **BSReportEDocCardType**;
    - вид документа равен значению константы **BSReportEDocKind**;
    - код приложения-редактора равен значению константы **BSReportEditorCode**.
- если одна из этих констант не заполнена, открывается диалоговое окно импорта документа из файла, в котором нужно заполнить необходимые параметры. Создается связь созданного документа с объектом системы Business Studio. Добавляется запись в справочнике **Связи**, у которой реквизиты заполняются в формате:
    - **DestID** – ИД созданного документа;
    - **ExternalID** – «<ObjectID>_<TemplateID>».

Затем функция выполняет поиск типового маршрута, у которого в свойстве задачи **Ссылка** указано значение, равное значению параметра **ObjectID**.

Поиск по типовым маршрутам может иметь результаты:

- если типовой маршрут найден, в его свойстве задачи **Инструкция** указывается ИД созданного документа;
- если значение параметра **SendToAgreement** равно **True**, создается задача по типовому маршруту. Код типового маршрута соответствует значению константы **BSReportAgreementStandardRouteCode**. В задачу вкладывается ссылка на документ, который был создан или найден в справочнике **Связи**. Откроется карточка задачи. Если значение параметра **SendToAgreement** равно **False**, появится сообщение с результатом импорта.

Пример:

```
' Импортировать отчет по оргструктуре предприятия из системы Business Studio в
' систему DIRECTUM.
IntLib.ImportReportFromBusinessStudio("C:\User\Org.xl", "c8f01bf6-e808-4521-9b72-
c011a781f773", "89c511dd-8044-4fa1-abe7-9c69e3058352");
```

Типы данных

Integer – целочисленные данные

Тип данных Integer хранит целые числа в интервале от -2147483648 до 2147483647, совместимые с OleVariant.

WordBool – логические данные

Тип WordBool хранит логические данные: **True** (истина) и **False** (ложь). Тип совместим с OleVariant.

WideString – строка в формате UNICODE

Тип WideString содержит строки в формате UNICODE. Каждый символ может занимать более одного байта (сколько именно – зависит от версии UNICODE). Строки являются динамическими, то есть память под них перераспределяется при изменении строки. Передача этих строк осуществляется по ссылке всегда, когда это возможно.