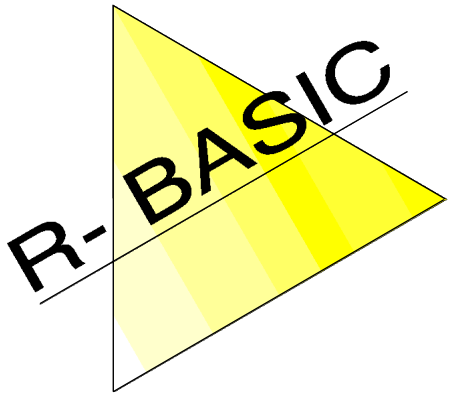
R-BASIC Extensions



DocumentTools Library Handbuch

DocumentTools Library Manual

Version 1.0
Initial Release

Inhaltsverzeichnis

Table of Contents

1 Konzeptionelles

Die DocumentTools Library stellt eine Reihe von Funktionen bereit, die Sie bei der Arbeit mit Dokumenten unterstützen. Dazu gehören zum Beispiel die typischen Dialogboxen zum Öffnen oder Speichern einer Datei. Um die DocumentTools Library nutzen zu können müssen Sie sie ihn Ihr Programm einbinden:

```
Include "DocumentTools"
```

Die Routinen der DocumentTools Library arbeiten eng mit einem Document-Guardian-Objekt zusammen. Damit diese Zusammenarbeit funktioniert müssen die Instancevariablen **ConfigData** und **ButtonHandler** des DocumentGuardian Objekts korrekt belegt sein. ConfigData speichert allgemeine Informationen über die vom DocumentGuardian verwalteten Dokumente, z.B. den Dateityp und das Token. Zum Beispiel kommuniziert die Libraryroutine DTOpenDialog direkt mit dem DocumentGuardian-Objekt, so dass sie den FileSelector des "Öffnen"-Dialogs auf den korrekten Pfad, den Dateityp und gegebenenfalls auf das Token der zu öffnenden Datei einstellen kann. Der ButtonHandler wird von der Routine DTShowNewOpenDialog benötigt, damit das BASIC-Programm auf eine Klick auf einen der Buttons in "Neu/Öffnen" Dialog reagieren kann.

Mithilfe der von der DocumentTools Library bereitgestellten Routinen ist es sehr einfach, ein Dokument-Interface mit den üblichen Funktionen zu programmieren. Wie man das macht ist ausführlich im Handbuch "Spezielle Themen", Volume 3, Kapitel 15, beschrieben. Die dort beschrieben Routinen sind dem Beispiel "Dokument Interface" entnommen, das sich im Ordner "R-BASIC\Beispiel\Objekte\ Dateiarbeit" befindet.
Zur Vereinfachung kann der gesamte Document-Interface Code über ein R-BASIC Menu in ihr Programm eingebaut werden. Dazu verwenden Sie das Menü "Extras" -> "Code Bausteine" -> "Dokument-Interface". Dort können Sie auch festlegen, welche Teile des Dokumentinterfaces Sie unterstützen wollen, z.B. ob Sie Muster-dokumente unterstützen wollen oder nicht.

Die von der Library bereitgestellten Routinen kann man grob in drei Gruppen unterteilen:
1. Bereitstellung von Dialogboxen, die Dateien bzw. Ordner auswählen, aber selbst keine Operationen mit den Dateien vornehmen.
2. Routinen, die selbst Dateien verändern, anlegen oder löschen können.
3. Tool-Routinen, die Aufgaben übernehmen, die nicht in die beiden anderen Gruppen fallen.

Alle Routinen handeln eventuell auftretende Fehler oder Probleme intelligent und selbständig. Die folgenden Seiten beschreiben die Funktion der einzelnen Routinen. Wie man sie im Zusammenspiel mit einem DocumentGuardian-Objekt einsetzt, finden Sie im oben genannten Handbuch.

<u>Routinen, die Dialogboxen bereitstellen</u>

Die SUBs **DTShowNewOpenDialog** und **DTCustomNewOpenDialog** stellen den typischen "Neu/Öffnen"-Dialog bereit. Mit den Funktionen **DTOpenDialog** bzw.

DTOpenTemplateDialog wird die Dialogbox zum Öffnen eines Dokuments bzw. eines Musterdokuments angezeigt. **DTSaveAsDialog** bzw. **DTSaveAsTemplate-Dialog** stellen die Dialogboxen "Speichern unter" bzw. "Als Muster speichern" bereit. Schließlich stellen die Funktionen **DTMoveCopyDialog** und **DTRename-Dialog** die Dialogboxen "Kopieren nach", "Verschieben nach" und "Umbenennen" bereit. Weil diese Routinen mit einem DocumentGuardian-Objekt zusammenarbeiten kennen sie selbständig den Dateityp und weitere Eigenschaften der Dokumentendatei, so dass sie z.B. bei der Eingabe eines Dateinamens sicherstellen können, dass die richtige Anzahl und nur gültige Zeichen eingegeben werden können.

Routinen, die Änderungen mit oder an Dateien vornehmen

Die SUB **DTCloneFile** legt eine Kopie der aktuell vom DocumentGuardian geöffneten Datei im aktuellen Ordner an. **DTCloneAndOpenFile** legt eine Kopie an und öffnet sie anschließend. Sie muss mit **DTCloseClone** wieder geschlossen werden. **DTConfirmAndDelete** löscht eine vorhandene Datei, fragt aber vorher beim Nutzer nach. Mit der SUB **DTChangeUserNotes** kann man die Benutzernotizen der vom DocumentGuardian geöffneten Datei ändern. Dazu wird vorher eine entsprechende Dialogbox angezeigt.

Sonstige Routinen

Die Funktion **DTFindNameForNew** findet einen Namen für eine neu anzulegende Datei. Datei greift sie auf die im DocumentGuardian abgelegte Config-Struktur zurück (z.B. Dateimaske, Dateityp). **DTCheckFileType** prüft, ob eine Datei kompatibel zu den im DocumentGuardian abgelegten Daten (ConfigStruct) ist. **DTConfirmClose** prüft den Dokumentenstatus des DocumentGuardian-Objekts, ob vor dem Schließen der aktuell geöffneten Datei weitere Aktionen sinnvoll sind (z.B. Speichern der geänderten Daten, Speichern unter neuem Namen bei unbenannten Dateien ...) und fragt beim Nutzer entsprechend nach.
Wenn Sie ein Datei-Menü implementieren müssen Sie sicherstellen, dass die Buttons (z.B. Öffnen, Speichern) in bestimmten Situationen enabled sind, in anderen nicht. Die Funktion **DTFindEnabled** liefert für jeden Button genau diese Information. Welcher Button gemeint ist, wird über eine Identifikationsnummer bestimmt. Dafür sind die folgenden Konstanten definiert:

Button	Konstante	Wert
Neu / Öffnen	ID_NEW_OPEN_DIALOG	1101
Neu	ID_NEW_DOC	1102
Öffnen	ID_OPEN_DOC	1103
Speichern	ID_SAVE_DOC	1104
Speichern unter	ID_SAVE_AS_DOC	1105
Schließen	ID_CLOSE_DOC	1106
Letzter Stand	ID_REVERT_DOC	1107
Backup anlegen	ID_QUICK_BACKUP	1108

Aus Backup wiederherstellen	ID_RESTORE_FROM_BACKUP	1109
Muster öffnen	ID_OPEN_TEMPLATE	1110
Als Muster speichern	ID_SAVE_AS_TEMPLATE	1111
Importieren	ID_IMPORT	1112
Benutzerebene ändern	ID_CONFIG	1113
Programm beenden	ID_EXIT_PROGRAM	1114
Kopieren nach	ID_COPY_TO	1115
Verschieben nach	ID_MOVE_TO	1116
Umbenennen	ID_RENAME	1117
Benutzernotizen ändern	ID_CHANGE_NOTES	1118
(reserviert)		1119 - 1130

Jede dieser Konstanten ist auch als actionData-Wert für den entsprechenden Button vorgesehen. Damit können Sie allen Buttons des Datei-Menüs den gleichen ActionHandler geben. Der Handler unterscheidet die Buttons anhand ihres actionData Wertes. Sinnvoller Weise weisen Sie genau diesen Actionhandler auch der Instancevariablen ButtonAction des DocumentGuardian-Objekts zu. Dann behandelt er auch die Klicks auf die Buttons in "Neu/Öffnen" Dialog (Routine DTShowNewOpenDialog). Diese Buttons haben automatisch die korrekten actionData-Werte gesetzt.

Fast alle Routinen der DocumentTools Library erwarten als ersten Parameter ein DocumentGuardian-Objekt. Die DocumentTools Library ist mit dem PC/GEOS-SDK (nicht in R-BASIC) geschrieben. Deswegen ist es erforderlich, die Referenz auf das DocumentGuardian-Objekt vor der Übergabe an die Library in eine für das PC/GEOS-SDK lesbare Form zu konvertieren. Dazu wird, wie in den folgenden Beispielen gezeigt, die R-BASIC-Routine ConvertObjForSDK verwendet. ConvertObjForSDK muss bei jedem Aufruf einer Routine aus der DocumentTools Library gerufen werden. Alternativ kann man das Ergebnis der Konvertierung auch in einer eigenen OBJECT-Variablen abspeichern (Beispiel 2).

Beispiel 1:

```
DIM ret as DialogReturnStruct
   ret = DTOpenDialog(ConvertObjForSDK(DocumentObj), "")
   IF ret.retInfo = DRI_CANCEL THEN RETURN
   ret = DTRenameDialog(ConvertObjForSDK(DocumentObj), "")
```

Beispiel 2:

```
DIM ret as DialogReturnStruct
DIM convertedObj as OBJECT
   convertedObj = ConvertObjForSDK(DocumentObj)
   ret = DTOpenDialog(convertedObj, "")
   IF ret.retInfo = DRI_CANCEL THEN RETURN
   ret = DTRenameDialog(convertedObj, "")
```

2 Dialogboxen

Alle Funktionen aus dieser Gruppe liefern eine Struktur zurück, die folgendermaßen definiert ist:

```
STRUCT DialogReturnStruct
   fileName$ as String[32]
   retInfo As Word
   End Struct
```

Das Feld "fileName$" enthält den vom Nutzer eingegeben bzw. ausgewählten Dateinamen. Das Feld "retInfo" enthält die Information, welchen Button im Dialog der Nutzer gedrückt hat. Dafür dien folgenden Konstanten definiert:

Konstante	Wert	Bedeutung
DRI_CANCEL	1	Der Nutzer hat "Abbrechen" gewählt.
DRI_OK	2	Der Nutzer hat "Speichern", "Öffnen" oder ähnliches gewählt.
DRI_READ_ONLY	3	Der Nutzer hat "Öffnen" oder ähnliches gewählt, die ausgewählte Datei ist jedoch schreibgeschützt oder soll schreibgeschützt geöffnet werden.

DTShowNewOpenDialog, DTCustomNewOpenDialog

Die Routinen zeigen den "Neu/Öffnen" Dialog an. Es wird **nicht** gewartet bis der Nutzer etwas im Dialog ausgewählt hat. Das ermöglicht das Anzeigen des Dialogs während das Programm startet.
Klickt der Nutzer auf einen der Buttons im Dialog so wird der "ButtonHandler" des übergebenen DocumentGuardian-Objekts aufgerufen und der Dialog wird geschlossen. Der actionData-Wert, der dem Handler übergeben wird, beschreibt, welcher Button gedrückt wurde:

Button	ActionData-Konstante	Wert
Neu	ID_NEW_DOC	1102
Öffnen	ID_OPEN_DOC	1103
Muster	ID_OPEN_TEMPLATE	1110
Importieren	ID_IMPORT	1112
Benutzerebene	ID_CONFIG	1113
Programm beenden	ID_EXIT_PROGRAM	1114
Abbrechen	–	–

Beim Klick auf "Abbrechen" wird der ButtonHandler nicht gerufen, der Dialog wird nur geschlossen.

DTCustomNewOpenDialog unterscheidet sich von DTShowNewOpenDialog dadurch, dass die Texte, die neben den Buttons angezeigt werden, geändert werden können. Die Grafiken auf den Buttons können nicht geändert werden.

Deklaration
SUB DTShowNewOpenDialog(docObj as Object, flags as word,
helpCtx$ as string(20))

SUB DTCustomNewOpenDialog(docObj as Object, flags as word,
helpCtx$ as string(20), newText$, openText$, templateText$, importText$, userLevelText$ as String)

docObj: Ein DocumentGuardian-Objekt. Das Objekt muss
- einen ButtonHandler gesetzt haben,
- durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

flags: Kombination von NOF_–Flags, die bestimmt, welche UI im Dialog angezeigt werden soll. Siehe Tabelle unten.

helpCtx$: Help-Context-String für den Dialog. Der Dialog erzeugt dann automatisch einen Hilfe-Button. Leerstring wenn kein Hilfebutton gewünscht ist.

newText$, openText$, templateText$, importText$, userLevelText$:
Nur für DTCustomNewOpenDialog. Texte, die neben den entsprechenden Buttons angezeigt werde sollen. Wird ein Leerstring übergeben so wird der originale Text verwendet.

Für den Parameter "flags" sind die folgenden Konstanten definiert:

Konstante	Wert	UI-Gruppe
NOF_NEW	1	Neue Datei anlegen
NOF_OPEN	2	Datei öffnen
NOF_TEMPLATE	4	Muster öffnen
NOF_IMPORT	8	Daten importieren
NOF_CONFIG	16	Benutzerebene einstellen
NOF_STARTUP	32	"Programm beenden" Button statt "Abbrechen" Button

Das Flag NOF_STARTUP sollte beim Aufruf der Dialogbox am Programmstart gesetzt werden. Dadurch erscheint statt des "Abbrechen" Buttons der Button "Programm beenden", der den actionData-Wert ID_EXIT_PROGRAM hat.

Zur Vereinfachung sind außerdem folgenden Konstanten definiert:

```
CONST NOF_NEW_OPEN = NOF_NEW + NOF_OPEN
CONST NOF_NEW_OPEN_TEMPLATE = NOF_NEW + NOF_OPEN + NOF_TEMPLATE
```

Beispiele:

```
DTShowNewOpenDialog(ConvertObjForSDK(MYGuardian),
    NOF_NEW_OPEN + NOF_STARTUP, "")

DTShowNewOpenDialog(ConvertObjForSDK(MYGuardian),
    NOF_NEW_OPEN_TEMPLATE + NOF_IMPORT, "")

DTCustomNewOpenDialog(ConvertObjForSDK(MYGuardian),
    NOF_NEW_OPEN_TEMPLATE, "", "Erstellen.\rEin neues Level für
    Supergame basteln.", "Bearbeiten\rEin vorhandenes Level für
    Supergame verändern.", "", "", "")
```

Ein einfacher ButtonHandler des DocumentGuardian-Objekts könnte wie folgt aussehen. Es wird vorausgesetzt, dass die Do~-Routinen irgendwo definiert sind.

```
BUTTONACTION DocumentAndToolButtonHandler
   ON actionData SWITCH
      CASE ID_NEW_DOC:
        DoCreateNewDoc
        END CASE
      CASE ID_OPEN_DOC:
        DoOpenDoc
        END CASE
      CASE ID_CLOSE_DOC:
        DoCloseDoc
        End CASE
      CASE ID_EXIT_PROGRAM:
        EXIT
        END CASE
   End SWICTH
END ACTION           ' DocumentAndToolButtonHandler
```

DTOpenDialog

Die Routine zeigt den "Öffnen"-Dialog an, führt aber selbst keine Dateioperationen aus. Wenn der Nutzer "Öffnen" wählt, wechselt die Routine in den vom FileSelector angezeigten Ordner.

Der Fileselektor in der Dialogbox wird entsprechend den Instancevariablen des übergebenen DocumentGuardian-Objekts initialisiert.

Die Routine gibt eine Variable vom Typ DialogReturnStruct (siehe vorn) zurück. Die Felder haben folgende Bedeutung:

fileName$: Name der vom Nutzer ausgewählten Datei.
Leerstring wenn der Nutzer "Abbrechen" gewählt hat.
retInfo: Zusätzliche Informationen
DRI_CANCEL: Der Nutzer hat "Abbrechen" gewählt.

DRI_OK: Der Nutzer hat "Öffnen" gewählt.
DRI_READ_ONLY: Der Nutzer hat "Öffnen" gewählt, aber die Option "zum Bearbeiten" deaktiviert.

Deklaration
FUNCTION DTOpenDialog(docObj As Object, helpCtx$ As String(20)) As DialogReturnStruct

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

helpCtx$: Help-Context-String für den Dialog. Der Dialog erzeugt dann automatisch einen Hilfe-Button. Leerstring wenn kein Hilfebutton gewünscht ist.

Return: DialogReturnStruct - Struktur

Beispiel:

```
DIM ret as DialogReturnStruct
DIM fh as FILE

   ret = DTOpenDialog(ConvertObjForSDK(MYGuardian), "")
   IF ret.retInfo = DRI_CANCEL THEN RETURN
   fh = FileOpen ( ret.fileName$ )
   ...
   FileClose (fh)
```

DTSaveAsDialog

Die Routine zeigt den "Speichern unter"-Dialog an, führt aber selbst keine Dateioperationen aus. Wenn der Nutzer "Speichern" wählt, wechselt die Routine in den vom FileSelector angezeigten Ordner.

Der Fileselektor in der Dialogbox und das Textobjekt für den neuen Namen werden entsprechend den Instancevariablen des übergebenen Document-Guardian-Objekts initialisiert (Pfad, Name der zu speichernden Datei usw.). Es werden also nur die Dateien angezeigt, die auch zum "Öffnen" angeboten werden.

Die Routine gibt eine Variable vom Typ DialogReturnStruct (siehe vorn) zurück. Die Felder haben folgende Bedeutung:

fileName$: Name, unter dem die Datei gespeichert werden soll.
Leerstring wenn der Nutzer "Abbrechen" gewählt hat.
retInfo: Zusätzliche Informationen
DRI_CANCEL: Der Nutzer hat "Abbrechen" gewählt.
DRI_OK: Der Nutzer hat "Speichern" gewählt.

Deklaration
FUNCTION DTSaveAsDialog(docObj as Object, helpCtx$ as string(20), disableFiles as Integer) as DialogReturnStruct

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

helpCtx$: Help-Context-String für den Dialog. Der Dialog erzeugt dann automatisch einen Hilfe-Button. Leerstring wenn kein Hilfebutton gewünscht ist.

disableFiles:
TRUE: Dateien disabled (grau) anzeigen.
FALSE: Dateien enabled anzeigen.

Return: DialogReturnStruct - Struktur

DTOpenTemplateDialog

Die Routine zeigt den "Muster Öffnen"-Dialog an, führt aber selbst keine Dateioperationen aus. Wenn der Nutzer "Öffnen" wählt, wechselt die Routine in den vom FileSelector angezeigten Ordner.

Konzeptionell ist die Routine gleichwertig zu DTOpenDialog, alle dort gemachten Ausführungen gelten auch für DTOpenTemplateDialog. DTOpenTemplateDialog startet aber immer im Template-Ordner.

Deklaration
FUNCTION DTOpenTemplateDialog(docObj as Object, helpCtx$ as string(20)) as DialogReturnStruct

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

helpCtx$: Help-Context-String für den Dialog. Der Dialog erzeugt dann automatisch einen Hilfe-Button. Leerstring wenn kein Hilfebutton gewünscht ist.

Return: DialogReturnStruct - Struktur

DTSaveAsTemplateDialog

Die Routine zeigt den "Speichern als Muster"-Dialog an, führt aber selbst keine Dateioperationen aus. Wenn der Nutzer "Als Muster speichern" wählt, wechselt die Routine in den vom FileSelector angezeigten Ordner.

Konzeptionell ist die Routine gleichwertig zu DTSaveAsDialog, alle dort gemachten Ausführungen gelten auch für DTSaveAsTemplateDialog. DTSaveAsTemplateDialog startet aber immer im Template-Ordner.

Deklaration
FUNCTION DTSaveAsTemplateDialog(docObj as Object, helpCtx$ as string(20), disableFiles as Integer) as DialogReturnStruct

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

helpCtx$: Help-Context-String für den Dialog. Der Dialog erzeugt dann automatisch einen Hilfe-Button. Leerstring wenn kein Hilfebutton gewünscht ist.

disableFiles:
TRUE: Dateien disabled (grau) anzeigen.
FALSE: Dateien enabled anzeigen.

Return: DialogReturnStruct - Struktur

DTMoveCopyDialog

Die Routine zeigt den Dialog zum Verschieben oder Kopieren des vom DocumentGuardian geöffneten Dokuments an, führt aber selbst keine Dateioperationen aus. Wenn der Nutzer "Verschieben" bzw. "Kopieren" wählt, wechselt die Routine in den vom FileSelector angezeigten Ordner.

Der Fileselektor in der Dialogbox wird entsprechend den Instancevariablen des übergebenen DocumentGuardian-Objekts initialisiert.

Die Routine gibt eine Variable vom Typ DialogReturnStruct (siehe vorn) zurück. Die Felder haben folgende Bedeutung:

fileName$: Name der neuen Datei.
Leerstring wenn der Nutzer "Abbrechen" gewählt hat.
retInfo: Zusätzliche Informationen
DRI_CANCEL: Der Nutzer hat "Abbrechen" gewählt.
DRI_OK: Der Nutzer hat "Verschieben" bzw. "Kopieren" gewählt.

Deklaration
FUNCTION DTMoveCopyDialog(docObj As Object, helpCtx$ As String(20), moveTo as Integer) As DialogReturnStruct

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

helpCtx$: Help-Context-String für den Dialog. Der Dialog erzeugt dann automatisch einen Hilfe-Button. Leerstring wenn kein Hilfebutton gewünscht ist.

moveTo: TRUE: "Verschieben nach"-Dialog
FALSE: "Kopieren nach"-Dialog

Return: DialogReturnStruct - Struktur

DTRenameDialog

Die Routine zeigt den Dialog zum Umbenennen der vom DocumentGuardian-Objekt geöffneten Datei an, führt aber selbst keine Dateioperationen aus.

Die Routine gibt eine Variable vom Typ DialogReturnStruct (siehe vorn) zurück. Die Felder haben folgende Bedeutung:

fileName$: Neuer Name der Datei.
Leerstring wenn der Nutzer "Abbrechen" gewählt hat.
retInfo: Zusätzliche Informationen
DRI_CANCEL: Der Nutzer hat "Abbrechen" gewählt.
DRI_OK: Der Nutzer hat "Umbenennen" gewählt.

Deklaration
FUNCTION DTRenameDialog(docObj As Object, helpCtx$ As String(20)) As DialogReturnStruct

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

helpCtx$: Help-Context-String für den Dialog. Der Dialog erzeugt dann automatisch einen Hilfe-Button. Leerstring wenn kein Hilfebutton gewünscht ist.

Return: DialogReturnStruct - Struktur

3 Bearbeiten von Dateien

DTConfirmAndDelete

Die Routine prüft ob die übergebene Datei im aktuellen Ordner existiert, fragt beim Nutzer nach, ob die Datei gelöscht werden soll und löscht die auf Anforderung. Tritt dabei ein Fehler auf gibt es eine Fehlermeldung und die globale Variable fileError wird auf den entsprechenden Fehlerwert gesetzt.

Situation	Rückgabewert	fileError-Wert
Die Datei existiert nicht	FALSE	0
Die Datei existiert und wurde erfolgreich gelöscht	FALSE	0
Die Datei existiert, soll aber nicht gelöscht werden	TRUE	0
Die Datei existiert, konnte aber nicht gelöscht werden	TRUE	Fehlerwert, je nach Problem

Deklaration
FUNCTION DTConfirmAndDelete(fileName$ as String) as Integer

fileName$: zu überprüfende Datei

Return: TRUE: Datei existiert nicht oder wurde erfolgreich gelöscht
FALSE: Datei existiert weiterhin

DTCloneFile

Die Routine dupliziert die die aktuell vom DocumentGuardian Objekt geöffnete Datei im aktuellen Ordner mit dem angegebenen Namen. Die Datei darf im aktuellen Ordner noch nicht existieren! Existiert die Datei oder tritt ein anderer Fehler auf gibt es eine entsprechende Fehlermeldung.
DTCloneFile setzt die globale Variable fileError (Null oder Fehlerwert).

Deklaration
SUB DTCloneFile(docObj as Object, newName$ as String)

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

newName$: Name für die Kopie

DTCloneAndOpenFile, DTCloseClone

Die Routine DTCloneAndOpenFile dupliziert die die aktuell vom Document-Guardian Objekt geöffnete Datei im aktuellen Ordner mit dem angegebenen Namen und öffnet sie anschließend. Die Datei darf im aktuellen Ordner noch nicht existieren! Existiert die Datei oder tritt ein anderer Fehler auf gibt es eine entsprechende Fehlermeldung.
DTCloseClone schließt eine mit DTCloneAndOpenFile geöffnete Datei.
Beide Routinen die globale Variable fileError (Null oder Fehlerwert).

Deklaration
FUNCTION DTCloneAndOpenFile(docObj as Object, newName$ as String) as FILE

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

newName$: Name für die Kopie

Return: Handle auf die geöffnete Kopie. Der Dateityp (DOS, GEOS oder VM-Datei) entspricht dem Typ der von docObj verwalteten Datei.

Deklaration
SUB DTCloseClone(fh as FILE)

fh: FILE-Handle auf die von DTCloneAndOpenFile duplizierte Datei.

DTChangeUserNotes

DTChangeUserNotes erlaubt das Ändern der Dokumentnotizen des aktuell vom DocumentGuardian-Objekt geöffneten Dokuments. Dazu zeigt die Routine eine Dialogbox an, die das Ändern der Dokumentnotizen zulässt und ändert sie dann.

Deklaration
SUB DTChangeUserNotes(docObj As Object)

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

4 Sonstige Tools

DTFindNameForNew

DTFindNameForNew findet einen Dateinamen, der zu den Instancedaten des DocumentGuardian-Objekts passt und der im aktuellen Ordner noch nicht existiert.

Deklaration
FUNCTION DTFindNameForNew(docObj as Object) as String

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

Return: Name, der sich für eine neue Datei eignet.

DTCheckFileType

Die Routine prüft, ob eine Datei kompatibel zum Dateityp des DocumentGuardian-Objekts ist. Sie vergleicht zunächst den Dateityp (DOS-, GEOS oder VM-File). Für GEOS und VM-Dateien werden zusätzlich das Token und das CreatorToken verglichen, für DOS-Dateien wird stattdessen die Dateierweiterung verglichen.
DTCheckFileType liefert Null (FALSE) wenn die Datei kompatibel ist, andernfalls liefert sie einen Fehlerwert z.B. ERROR_FILE_FORMAT_MISMATCH.

Deklaration
FUNCTION DTCheckFileType(docObj as Object, fileName$ as String,
dosExt$ as String) as Integer

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

fileName$: Name der zu prüfenden Datei im aktuellen Ordner.

dosExt$: Nur für DOS-Dateien: Dateierweiterung. Leerstring wenn die DOS-Datei keine Erweiterung haben darf (z.B. "LIESMICH").
Wildcards (*,?) sind NICHT zugelassen!
Ausnahme: Um das Prüfen der DOS-Erweiterung zu umgehen übergeben Sie "*" (ein Sternchen).

Return: Null (FALSE) wenn die Datei kompatibel ist
Fehlerwert wenn die Datei nicht kompatibel ist

Beispiele:

```
DIM err, fileName$
fileName$ = FileFindNext$( ... )

err = DTCheckFileType(ConvertObjForSDK(MyDocGuardian),
    fileName$, "*")
IF err THEN MsgBox "Nicht kompatibel: "+ErrorText$(err)

err = DTCheckFileType(ConvertObjForSDK(MyDocGuardian),
    fileName$, "HTM")
```

DTConfirmClose

Die Funktion DTConfirmClose prüft die Instancevariable "documentState" des übergebenen DocumentGuardian-Objekts, ob das vom DocumentGuardian-Objekt geöffnete Dokument einfach geschlossen werden kann oder ob weitere Aktionen nötig sind. Im Zweifelsfall wird der Nutzer durch eine Dialogbox gefragt, wie weiter zu verfahren ist. DTConfirmClose handelt **alle denkbaren Fälle** und liefert einen der folgenden Werte zurück:

Konstante	Wert	Vorgehen
CLOSE_DISCARD	0	Die Datei soll ohne Speichern geschlossen werden, dh. Änderungen werden verworfen.
CLOSE_SAVE	1	Die Datei soll vor dem Schließen gespeichert werden.
CLOSE_SAVE_AS	2	Die Datei ist neu oder schreibgeschützt und soll vor dem Schließen unter neuem Namen gespeichert werden.
CLOSE_CANCEL	3	Die Datei soll doch nicht geschlossen werden weil der Nutzer "Abbrechen" gewählt hat.
CLOSE_NO_FILE	4	Es ist keine Datei offen.

Folgende Situationen sind möglich:

- Es ist keine Datei offen:
  Nachfrage beim Nutzer: nein
  Rückgabewert: CLOSE_NO_FILE
- Die Datei ist nicht als geändert markiert:
  Nachfrage beim Nutzer: nein
  Rückgabewert: CLOSE_DISCARD
- Die Datei ist als geändert markiert:
  Nachfrage beim Nutzer: Änderungen speichern?
  Rückgabewert: CLOSE_SAVE, CLOSE_DISCARD oder CLOSE_CANCEL
- Die Datei ist als geändert markiert und unbenannt:
  Nachfrage beim Nutzer: Neue Datei speichern?

Rückgabewert: CLOSE_SAVE_AS, CLOSE_DISCARD
oder CLOSE_CANCEL

- Die Datei ist als geändert markiert aber schreibgeschützt:
  Nachfrage beim Nutzer: Unter neuem Namen speichern?
  Rückgabewert: CLOSE_SAVE_AS, CLOSE_DISCARD
  oder CLOSE_CANCEL

Deklaration
FUNCTION DTConfirmClose(docObj as Object, allowCancel as Integer) as word

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

allowCancel: "Abbrechen" Button anzeigen
TRUE: "Abbrechen" Button anzeigen
FALSE: "Abbrechen" Button nicht anzeigen

Return: CLOSE_SAVE, CLOSE_SAVE_AS, CLOSE_DISCARD,
CLOSE_CANCEL oder CLOSE_NO_FILE

DTFindEnabled

Wenn Sie ein Datei-Menü implementieren müssen Sie sicherstellen, dass die Buttons (z.B. Öffnen, Speichern) in bestimmten Situationen enabled sind, in anderen nicht. Die Funktion **DTFindEnabled** liefert für jeden Button genau diese Information. Welcher Button gemeint ist, wird über eine Identifikationsnummer bestimmt. Eine Liste der dafür definierten Konstanten finden Sie vor, im Abschnitt 1.

Deklaration
FUNCTION DTFindEnabled(docObj As Object, actionID as Word) As integer

docObj: Ein DocumentGuardian-Objekt. Die Referenz auf das Objekt muss durch den Aufruf von **ConvertObjForSDK** konvertiert werden.

actionID: Identifizierungsnummer des Buttons

Beispiel:

```
MenuOpenButton.enabled = DTFindEnabled(
      ConvertObjForSDK(MyDocGuardian), ID_OPEN_DOC)
ToolsTemplateButton.enabled = DTFindEnabled(
      ConvertObjForSDK(MyDocGuardian), ID_OPEN_TEMPLATE)
```

1 Concepts

The DocumentTools library provides a lot of functions to help you working with documents. One example are the typically dialog boxes to open a file or to save it with a new name. To use the DocumentTools library you have to include it:

```
Include "DocumentTools"
```

The routines of the DocumentTools library works closely together with the DocumentGuardian object. To make this work, the instance variables **ConfigData** and **ButtonHandler** of the DocumentGuardian objects have to be set correctly. ConfigData stores some common information about the documents, managed by the DocumentGuardian, for example the file type and the token. As an example, the library routine DTOpenDialog communicates directly with the Document-Guardian object. Therefore, it can configure the FileSelector object inside of the dialog box properly. The ButtonHandler is used by the routine DTShowNewOpen-Dialog to make it possible to call an action handler if the user hits one of the buttons inside of the dialog box.

Using the routines from the DocumentTools library makes it very easy to imple-ment a document interface, which provides the common functionality. This is explained in detail in the "Special Topics" manual, volume 3, chapter 15. All routines described there can be found in the example "Document Interface", which can be found in the folder "R-BASIC\Examples\Objects\Working with files".
For convenience, the whole document interface code can be included into your program from a R-BASIC menu. Use "Extras", "Code Sequences", "document Interface" for this purpose. Additionally, the code can be customized here, for example if you wish to support templates or not.

The routines of the library may be sorted into three groups:
1. Routines, which displays dialog boxes to select a folder or a file. These routines do not perform any file operation.
2. Routines, which may modify, create or delete files.
3. Tools, which performs some helpful other things.

All routines react on errors or problems automatically. For example, when the user enters the name of a folder as a file name, the name will not be accepted. The following pages describe the routine definitions and what it does. See manual mentioned above to learn how to use the routines together with a Document-Guardian object to implement a working document interface.

Routines, which shows dialog boxes

DTShowNewOpenDialog and **DTCustomNewOpenDialog** provide the typically "New/Open" dialog box. The names of the routines **DTOpenDialog**, **DTOpenTemplateDialog**, **DTSaveAsDialog**, **DTSaveAsTemplateDialog** and **DTRenameDialog** should be self-explaining. **DTMoveCopyDialog** can be configured to show the "Move to" or the "Copy to" dialog box.

Because the routines are working together with the DocumentGuardian object they know the properties of the document. This allows, for example, ensuring that the filename, entered by the user, is valid.

Routines which may change files

The SUB **DTCloneFile** creates a copy of the file, currently opened by the DocumentGuardian object. **DTCloneAndOpenFile** creates a copy an opens it. This open copy must be closed with **DTCloseClone**. **DTConfirmAndDelete** deletes a file, but it asks the user beforehand. **DTChangeUserNotes** allows editing the user notes of the document file, currently opened by the DocumentGuardian object. It provides a dialog box where the user can see and modify the user notes.

Other routines

The function **DTFindNameForNew** finds out a name for a new file. It accesses the ConfigStruct instance variable of the DocumentGuardian object to choose a DOS or a GEOS name. **DTCheckFileType** checks if the passed file is compatible to the data, stored in the DocumentGuardians ConfigStruct. **DTConfirmClose** checks the document state and asks the user if he wish to save or discard changes before closing the file.
When you implement a file menu, you have to know if a given button (for example "Open" or "Save") should be enabled or disabled in the current situation. The function **DTFindEnabled** returns exactly this information. The button is specified by an id number. The following constants are defined for this purpose:

Button	Constant	Value
New / Open	ID_NEW_OPEN_DIALOG	1101
New	ID_NEW_DOC	1102
Open	ID_OPEN_DOC	1103
Save	ID_SAVE_DOC	1104
Save as	ID_SAVE_AS_DOC	1105
Close	ID_CLOSE_DOC	1106
Revert to last saved state	ID_REVERT_DOC	1107
Create backup	ID_QUICK_BACKUP	1108
Revert from backup	ID_RESTORE_FROM_BACKUP	1109
Open template	ID_OPEN_TEMPLATE	1110
Save as template	ID_SAVE_AS_TEMPLATE	1111
Import	ID_IMPORT	1112
Change user level	ID_CONFIG	1113
Exit program	ID_EXIT_PROGRAM	1114
Copy to	ID_COPY_TO	1115
Move to	ID_MOVE_TO	1116

Rename	ID_RENAME	1117
Edit user notes	ID_CHANGE_NOTES	1118
(reserved)		1119 - 1130

These constants also may be used as actionData values for the corresponding buttons. This allows you to set the same action handler for every button. The handler distinguishes the buttons by its actionData value. You also should assign this action handler to the ButtonAction instance variable of the des DocumentGuardian object. Then, the handler also handles the buttons of the "New/Open" dialog (routine DTShowNewOpenDialog). The buttons in this dialog box have set the correct actionData value automatically.

Mostly all of the routines of the DocumentTools library expect a Document-Guardian object as first parameter. Because the DocumentTools library was written with PC/GEOS-SDK (not with R-BASIC), the reference to this object has to be converted. This will be done by the R-BASIC routine ConvertObjForSDK, as shown in the examples below.

Example 1:

```
DIM ret as DialogReturnStruct
   ret = DTOpenDialog(ConvertObjForSDK(DocumentObj), "")
   IF ret.retInfo = DRI_CANCEL THEN RETURN
   ret = DTRenameDialog(ConvertObjForSDK(DocumentObj), "")
```

Example 2:

```
DIM ret as DialogReturnStruct
DIM convertedObj as OBJECT
   convertedObj = ConvertObjForSDK(DocumentObj)
   ret = DTOpenDialog(convertedObj, "")
   IF ret.retInfo = DRI_CANCEL THEN RETURN
   ret = DTRenameDialog(convertedObj, "")
```

2 Dialog Boxes

All functions described in this section return a structure as defined below:

```
STRUCT DialogReturnStruct
   fileName$ as String[32]
   retInfo As Word
   End Struct
```

The filed "fileName$" contains the name of the file, entered or selected by the user. The field "retInfo" contains information about the button pressed. The following constants are defined for this purpose:

Constant	Value	Meaning
DRI_CANCEL	1	The user has chosen "Cancel".
DRI_OK	2	The user has chosen "Save", "Open" or similar.
DRI_READ_ONLY	3	The user has chosen "Open" or similar, but the selected file is write protected or should be opened for read-only access.

DTShowNewOpenDialog, DTCustomNewOpenDialog

The routines show the "New/Open" dialog box. They will **not** wait until the user has selected anything from the dialog box. This allows the dialog box to appear when the program starts up.
When the user hits one of the buttons inside of the dialog box, the "ButtonHandler" of the passed DocumentGuardian object will be called an the dialog box will be closed automatically. The actionData value, passed to the handler, describes the button which was clicked.

Button	ActionData Constant	Value
New	ID_NEW_DOC	1102
Open	ID_OPEN_DOC	1103
Template	ID_OPEN_TEMPLATE	1110
Import	ID_IMPORT	1112
Change user level	ID_CONFIG	1113
Exit program	ID_EXIT_PROGRAM	1114
Cancel	–	–

When the user selects "Cancel", the ButtonHandler will not be called, only the dialog becomes closed.

DTCustomNewOpenDialog differs from DTShowNewOpenDialog by the fact, that the texts, shown on the right side of the buttons, may be changed. The button images cannot be changed.

Declaration
SUB DTShowNewOpenDialog(docObj as Object, flags as word,
helpCtx$ as string(20))

SUB DTCustomNewOpenDialog(docObj as Object, flags as word,
helpCtx$ as string(20), newText$, openText$, templateText$, importText$, userLevelText$ as String)

docObj: A DocumentGuardian object. The object has to be
- set a ButtonHandler,
- converted by calling **ConvertObjForSDK**.

flags: Combination of NOF_–flags: They describe, which UI should be visible in the dialog box. See table below.

helpCtx$: Help context string for the dialog. The dialog will create a help button automatically then. Pass an empty string to prevent the dialog from creating a help button.

newText$, openText$, templateText$, importText$, userLevelText$:
Only for DTCustomNewOpenDialog. Text to show on the right side of the button. If you wish to use the original text, pass an empty string.

For the parameter "flags" the following constants are defined:

Constant	Value	UI Group
NOF_NEW	1	Create new document
NOF_OPEN	2	Open document
NOF_TEMPLATE	4	Open template
NOF_IMPORT	8	Import data
NOF_CONFIG	16	Change user level
NOF_STARTUP	32	"Exit Program" button instead of "Cancel" button

The flag NOF_STARTUP should be set when the dialog box is initiated on program startup. Then the button "Exit Program", which has the actionData value ID_EXIT_PROGRAM set, will appear instead of the "Cancel" button.

Additionally, the following constants are defined:

```
CONST NOF_NEW_OPEN = NOF_NEW + NOF_OPEN
CONST NOF_NEW_OPEN_TEMPLATE = NOF_NEW + NOF_OPEN + NOF_TEMPLATE
```

Examples:

```
DTShowNewOpenDialog(ConvertObjForSDK(MYGuardian),
    NOF_NEW_OPEN + NOF_STARTUP, "")

DTShowNewOpenDialog(ConvertObjForSDK(MYGuardian),
    NOF_NEW_OPEN_TEMPLATE + NOF_IMPORT, "")

DTCustomNewOpenDialog(ConvertObjForSDK(MYGuardian),
    NOF_NEW_OPEN_TEMPLATE, "", "Create.\rBuild a new level for
    Supergame.", "Edit.\rChange an existing level for
    Supergame.", "", "", "")
```

A simple ButtonHandler of the DocumentGuardian object may look as follows. We assume that the Do~-routines are declared elsewhere.

```
BUTTONACTION DocumentAndToolButtonHandler
   ON actionData SWITCH
      CASE ID_NEW_DOC:
        DoCreateNewDoc
        END CASE
      CASE ID_OPEN_DOC:
        DoOpenDoc
        END CASE
      CASE ID_CLOSE_DOC:
        DoCloseDoc
        End CASE
      CASE ID_EXIT_PROGRAM:
        EXIT
        END CASE
   End SWICTH
END ACTION           ' DocumentAndToolButtonHandler
```

DTOpenDialog

The routine shows the "Open" dialog, but does not perform any file operation. If the user selects "Open", the routine changes to the folder, which is shown by the FileSelector.

The FileSelector in the dialog box will initialized according to the instance variables of the passed DocumentGuardian object.

The routine returns a variable of type DialogReturnStruct (see above). The particular fields mean:

fileName$: Name of the file, selected by the user.
Empty string, if "Cancel" was selected.
retInfo: Additional information
DRI_CANCEL: The user has chosen "Cancel".

DRI_OK: The user has chosen "Open".
DRI_READ_ONLY: The user has chosen "Open", but the option "For Edit" is not selected.

Declaration
FUNCTION DTOpenDialog(docObj As Object, helpCtx$ As String(20)) As DialogReturnStruct

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

helpCtx$: Help context string for the dialog. The dialog will create a help button automatically then. Pass an empty string to prevent the dialog from creating a help button.

Return: DialogReturnStruct - structure

Example:

```
DIM ret as DialogReturnStruct
DIM fh as FILE

   ret = DTOpenDialog(ConvertObjForSDK(MYGuardian), "")
   IF ret.retInfo = DRI_CANCEL THEN RETURN
   fh = FileOpen ( ret.fileName$ )
   ...
   FileClose (fh)
```

DTSaveAsDialog

The routine shows the "Save as" dialog, but does not perform any file operation. If the user selects "Save", the routine changes to the folder, which is shown by the FileSelector.

The FileSelector inside of the dialog box and the text object for the new name will be initialized according to the instance variables of the passed DocumentGuardian object (Path, name of file to save etc.).

The routine returns a variable of type DialogReturnStruct (see above). The particular fields mean:

fileName$: Name for the new file.
Empty string, if "Cancel" was selected.
retInfo: Additional information
DRI_CANCEL: The user has chosen "Cancel".
DRI_OK: The user has chosen "Save".

Declaration
FUNCTION DTSaveAsDialog(docObj as Object, helpCtx$ as string(20), disableFiles as Integer) as DialogReturnStruct

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

helpCtx$: Help context string for the dialog. The dialog will create a help button automatically then. Pass an empty string to prevent the dialog from creating a help button.

disableFiles:
TRUE: Show file disabled (grayed out).
FALSE: Show file enabled.

Return: DialogReturnStruct - structure

DTOpenTemplateDialog

The routine shows the "Open Template" dialog, but does not perform any file operation. If the user selects "Open", the routine changes to the folder, which is shown by the FileSelector.

The routine is similar to DTOpenDialog, all comments made there also applies to DTOpenTemplateDialog. But DTOpenTemplateDialog always starts with the template folder.

Declaration
FUNCTION DTOpenTemplateDialog(docObj as Object, helpCtx$ as string(20)) as DialogReturnStruct

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

helpCtx$: Help context string for the dialog. The dialog will create a help button automatically then. Pass an empty string to prevent the dialog from creating a help button.

Return: DialogReturnStruct - structure

DTSaveAsTemplateDialog

The routine shows the "Save As Template" dialog, but does not perform any file operation. If the user selects "Save", the routine changes to the folder, which is shown by the FileSelector.

The routine is similar to DTSaveAsDialog, all comments made there also applies to DTSaveAsTemplateDialog. But DTSaveAsTemplateDialog always starts with the template folder.

Declaration
FUNCTION DTSaveAsTemplateDialog(docObj as Object, helpCtx$ as string(20), disableFiles as Integer) as DialogReturnStruct

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

helpCtx$: Help context string for the dialog. The dialog will create a help button automatically then. Pass an empty string to prevent the dialog from creating a help button.

disableFiles:
TRUE: Show file disabled (grayed out).
FALSE: Show file enabled.

Return: DialogReturnStruct - structure

DTMoveCopyDialog

The routine shows the Dialog boxes to move or copy the currently opened document, but does not perform any file operation. If the user selects "Move" or "Copy", the routine changes to the folder, which is shown by the FileSelector.

The FileSelector in the dialog box will initialized according to the instance variables of the passed DocumentGuardian object.

The routine returns a variable of type DialogReturnStruct (see above). The particular fields mean:

fileName$: Name of the new file.
Empty string, if "Cancel" was selected.
retInfo: Additional information
DRI_CANCEL: The user has chosen "Cancel".
DRI_OK: The user has chosen "Move" or "Copy".

Declaration
FUNCTION DTMoveCopyDialog(docObj As Object, helpCtx$ As String(20), moveTo as Integer) As DialogReturnStruct

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

helpCtx$: Help context string for the dialog. The dialog will create a help button automatically then. Pass an empty string to prevent the dialog from creating a help button.

moveTo: TRUE: "Move to" dialog
FALSE: "Copy to" dialog

Return: DialogReturnStruct - structure

DTRenameDialog

The routine shows the dialog box to rename the document file, currently opened by the DocumentGuardian object, but does not perform any file operation.

The routine returns a variable of type DialogReturnStruct (see above). The particular fields mean:

fileName$: New name of the file.
Empty string, if "Cancel" was selected.
retInfo: Additional information
DRI_CANCEL: The user has chosen "Cancel".
DRI_OK: The user has chosen "Rename".

Declaration
FUNCTION DTRenameDialog(docObj As Object, helpCtx$ As String(20)) As DialogReturnStruct

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

helpCtx$: Help context string for the dialog. The dialog will create a help button automatically then. Pass an empty string to prevent the dialog from creating a help button.

Return: DialogReturnStruct - structure

3 Edit Files

DTConfirmAndDelete

The routine checks if the passed file exists in current folder, asks the user if he wishes to delete the file and deletes it, if requested. When an error occurs, a message box is shown and the global variable fileError will be set to the proper error value.

Situation	Return value	fileError value
The file does not exists	FALSE	0
The file exits and was deleted successfully	FALSE	0
The file exits but should not be deleted	TRUE	0
The file still exits, deleting the file fails	TRUE	Error value, depending on problem

Declaration
FUNCTION DTConfirmAndDelete(fileName$ as String) as Integer

fileName$: file to check and to delete

Return: TRUE: File does not exist or was deleted successfully
FALSE: File still exists

DTCloneFile

The routine creates a duplicate copy of the file, currently opened by the DocumentGuardian object. The new file must not currently exist in current folder! If the file exists or if another error occurs, an error message will be shown.
DTCloneFile sets the global variable fileError (zero or error value).

Declaration
SUB DTCloneFile(docObj as Object, newName$ as String)

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

newName$: New name for the copy

DTCloneAndOpenFile, DTCloseClone

The routine creates a duplicate copy of the file, currently opened by the DocumentGuardian object and opens it. The new file must not currently exist in current folder! If the file exists or if another error occurs, an error message will be shown.
DTCloseClone closes a file, opened by DTCloneAndOpenFile.
Both routines set the global variable fileError (zero or error value).

Declaration
FUNCTION DTCloneAndOpenFile(docObj as Object, newName$ as String) as FILE

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

newName$: New name of the copy

Return: Handle to the opened copy. The file type (DOS, GEOS or VM file) is equal to the type specified by the instance data of docObj.

Declaration
SUB DTCloseClone(fh as FILE)

fh: FILE handle of the copy, created by DTCloneAndOpenFile.

DTChangeUserNotes

DTChangeUserNotes shows a dialog box which allows the user to edit the user notes of the file, currently open by the passed DocumentGuardian object. When the user clicks on OK, the user notes will be changed.

Declaration
SUB DTChangeUserNotes(docObj As Object)

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

4 Other tools

DTFindNameForNew

DTFindNameForNew returns a filename, which is compaible to the instance variables of the passed DocumentGuardian object and which currently not exist in current folder.

Declaration
FUNCTION DTFindNameForNew(docObj as Object) as String

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

Return: Name, usable for a new file.

DTCheckFileType

The routine checks if a file is compatible to the file type, specified by the instance variables of the passed DocumentGuardian object. At first, it checks the file type (DOS-, GEOS or VM file). For GEOS and VM files, the token and the creator token will be compared. For DOS file, the file extension will be compared instead.
DTCheckFileType returns zero (FALSE) if the file is compatible, it returns an error value (for example ERROR_FILE_FORMAT_MISMATCH) if the file is not compatible.

Declaration
FUNCTION DTCheckFileType(docObj as Object, fileName$ as String,
dosExt$ as String) as Integer

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

fileName$: Name of the file to check out.

dosExt$: For DOS files only: File extension. Pass an empty string if the file should have no extension ("README" e.g.).
Wildcards (*,?) are NOT allowed!
Exception: Pass "*" (a single star) to skip extension checking.

Return: Zero (FALSE) if the file is compatible
Error value if the file is not compatible

Examples:

```
DIM err, fileName$
fileName$ = FileFindNext$( ... )

err = DTCheckFileType(ConvertObjForSDK(MyDocGuardian),
    fileName$, "*")
IF err THEN MsgBox "Not compatible because: "+ErrorText$(err)

err = DTCheckFileType(ConvertObjForSDK(MyDocGuardian),
    fileName$, "HTM")
```

DTConfirmClose

The function DTConfirmClose checks the instance variable "documentState" of the passed DocumentGuardian object, to see if the document can be closed safely or if there are other operations appropriate beforehand. If so, the user will be asked for proper reaction. DTConfirmClose handles **all possible cases** and returns one of the following values:

Constant	value	Expected reaction
CLOSE_DISCARD	0	Discard changes. Close the file without saving it.
CLOSE_SAVE	1	Save modified data to file and close it then.
CLOSE_SAVE_AS	2	The file is untitled or write protected. Save the file with a new name before closing it.
CLOSE_CANCEL	3	Do not close the file, the user has chosen "Cancel".
CLOSE_NO_FILE	4	There is no file open.

The following situations are possible:

- There is no file open:
  Question to user: none
  Return value: CLOSE_NO_FILE
- The file is not marked as modified:
  Question to user: none
  Return value: CLOSE_DISCARD
- The file is marked as modified:
  Question to user: Save changes?
  Return value: CLOSE_SAVE, CLOSE_DISCARD or CLOSE_CANCEL
- The file is marked as modified but untitled:
  Question to user: Save new file?
  Return value: CLOSE_SAVE_AS, CLOSE_DISCARD or CLOSE_CANCEL
- The file is marked as modified but write protected:
  Question to user: Save file with a new name?
  Return value: CLOSE_SAVE_AS, CLOSE_DISCARD or CLOSE_CANCEL

Declaration
FUNCTION DTConfirmClose(docObj as Object, allowCancel as Integer) as word

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

allowCancel: Show "Cancel" button
TRUE: Show "Cancel" button
FALSE: Hide "Cancel" button

Return: CLOSE_SAVE, CLOSE_SAVE_AS, CLOSE_DISCARD, CLOSE_CANCEL oder CLOSE_NO_FILE

DTFindEnabled

When you implement a file menu, you have to know if a given button (for example "Open" or "Save") should be enabled or disabled in the current situation. The function **DTFindEnabled** returns exactly this information. The button is specified by an id number. The allowed constants are shown above, in chapter 1.

Declaration
FUNCTION DTFindEnabled(docObj As Object, actionID as Word) As integer

docObj: A DocumentGuardian object. The reference to the object has to be converted by calling **ConvertObjForSDK**.

actionID: Number which identifies the button

Examples:

```
MenuOpenButton.enabled = DTFindEnabled(
      ConvertObjForSDK(MyDocGuardian), ID_OPEN_DOC)
ToolsTemplateButton.enabled = DTFindEnabled(
      ConvertObjForSDK(MyDocGuardian), ID_OPEN_TEMPLATE)
```