4D View Pro Handbuch

Einführung

Über 4D View Pro

4D View Pro bietet Funktionen für Tabellenkalkulationen und zur Darstellung von Listen. 4D View Pro bietet 4D Benutzern eine moderne und integrierte Alternative zu den Features des bisherigen Plug-In 4D View.

4D View Pro besteht aus zwei Teilen:

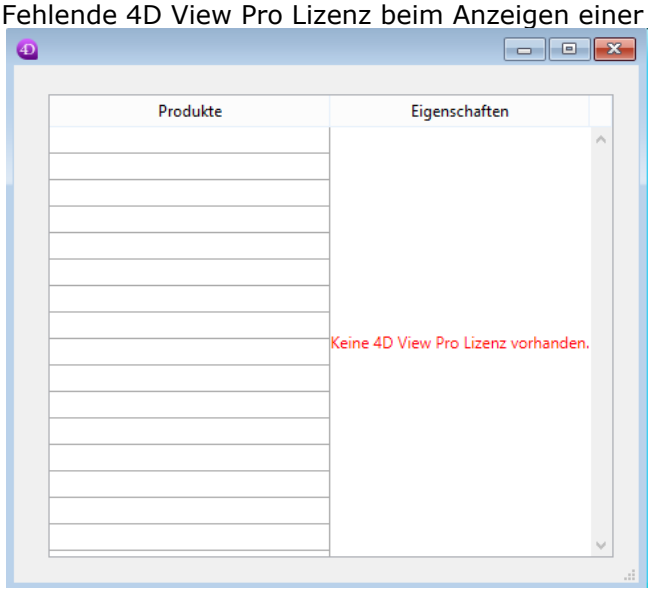
- Erweiterte Features für Listboxen, wie z.B. die volle Kontrolle über individuelle Zeilenhöhen
- Eine Komponente und ein 4D Formularbereich, womit Entwickler in ihre Formulare eine Tabellenkalkulation (spreadsheet) integrieren können

Installation und Aktivierung

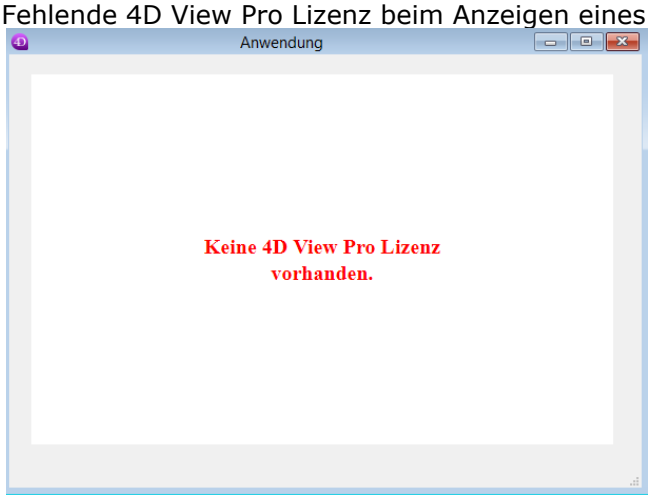
Features von 4D View Pro sind Im Unterschied zum bisherigen 4D View direkt in 4D enthalten. Das macht die Weitergabe und Verwaltung einfacher, denn es ist keine zusätzliche Installation erforderlich.

4D View Pro nutzt dieselbe Lizenz wie 4D View. Ist keine Lizenz für 4D View installiert, erscheinen im laufenden Betrieb keine Inhalte eines Objekts, das ein Feature von 4D View Pro verwendet (Listbox oder 4D View Pro Bereich), sondern stattdessen eine Fehlermeldung.

- Fehlende 4D View Pro Lizenz beim Anzeigen einer Listbox, die ein 4D View Pro Feature verwendet:



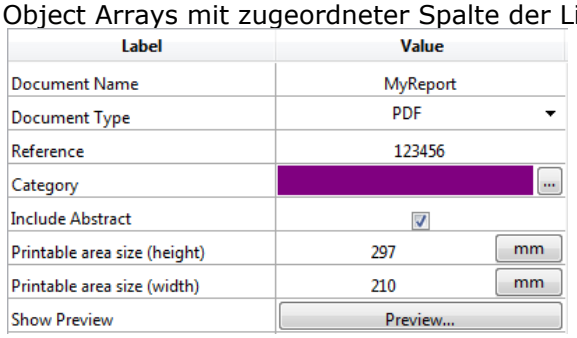
- Fehlende 4D View Pro Lizenz beim Anzeigen eines 4D View Pro Bereichs Tabellenkalkulation:



Erweiterte Funktionalitäten der Listbox

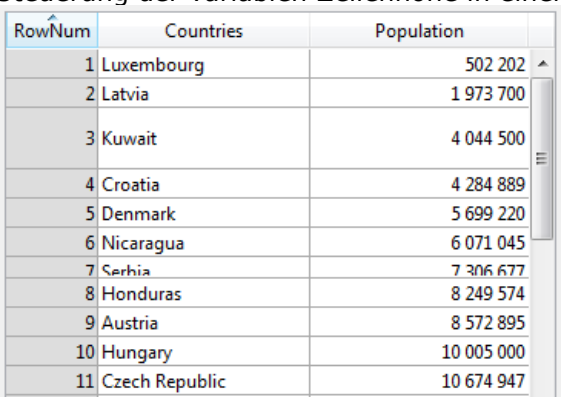
Wie schon in der **Einführung** erläutert, besteht ein Teil von 4D View Pro aus einem Satz erweiterter Funktionalitäten der Listbox. Das beinhaltet folgendes:

- Object Arrays mit zugeordneter Spalte der Listbox:



  Weitere Informationen dazu finden Sie im Abschnitt **Objekt Arrays in Spalten von Listboxen (4D View Pro)**.

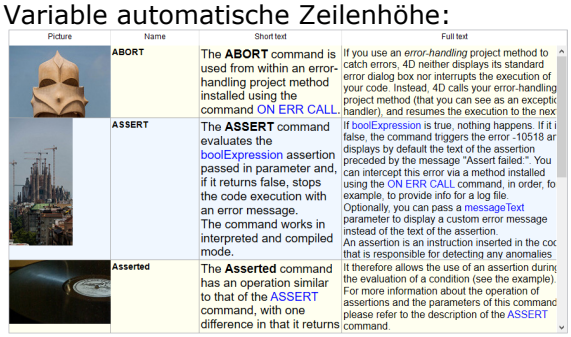
- Steuerung der variablen Zeilenhöhe in einer Listbox:



RowNum	Countries	Population
1	Luxembourg	502 202
2	Latvia	1 973 700
3	Kuwait	4 044 500
4	Croatia	4 284 889
5	Denmark	5 699 220
6	Nicaragua	6 071 045
7	Serbia	7 306 677
8	Honduras	8 249 574
9	Austria	8 572 895
10	Hungary	10 005 000
11	Czech Republic	10 674 947

  Dieses Feature basiert auf folgenden Optionen:
  - **Eigenschaft Zeilenhöhe Array**
  - Befehle **LISTBOX Get row height** und **LISTBOX SET ROW HEIGHT**
  - Konstante lk row height array für die Befehle **LISTBOX Get array** und **LISTBOX SET ARRAY**.
    Weitere Informationen dazu finden Sie im Handbuch *4D Programmiersprache*

- Variable automatische Zeilenhöhe:



  Dieses Feature basiert auf folgenden Optionen:
  - Eigenschaft **Automatische Zeilenhöhe** (verfügbar für die Listbox und einzelne Spalten)
  - Konstante lk auto row height für die Befehle **LISTBOX SET PROPERTY** und **LISTBOX Get property**
  - Befehle **LISTBOX SET AUTO ROW HEIGHT** und **LISTBOX Get auto row height** zum Definieren der Höhengrenzen
    Weitere Informationen dazu finden Sie im Handbuch *4D Programmiersprache*.

4D View Pro Bereich definieren

Überblick

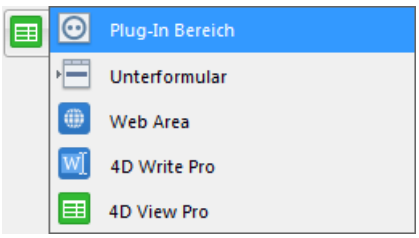
Mit 4D View Pro können Sie in Ihren 4D Formularen einen Bereich mit Tabellenkalkulation einfügen und anzeigen.

Sind 4D View Pro Bereiche in Ihren Formularen angelegt, können Sie über die 4D View Pro Befehle Dokumente mit Tabellenkalkulation importieren und exportieren.

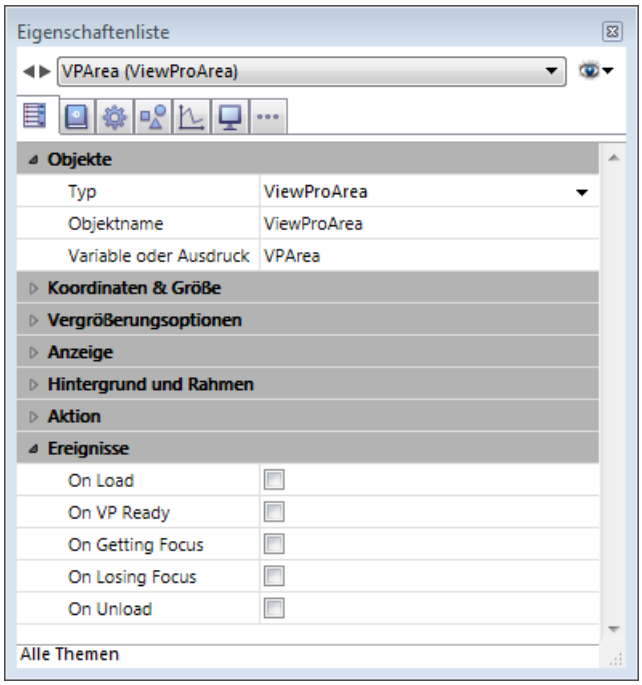
Kompatibilität: 4D View Pro Bereiche sind in 32-bit Versionen von 4D nicht verwendbar.

Bereich erstellen

4D View Pro Dokumente werden in einem 4D Formularobjekt mit Namen **4D View Pro** angezeigt und manuell bearbeitet. Sie können es in der Objektleiste am linken Rand unter dem letzten Tool auswählen (Plug-in Bereich, Web Area, etc.):

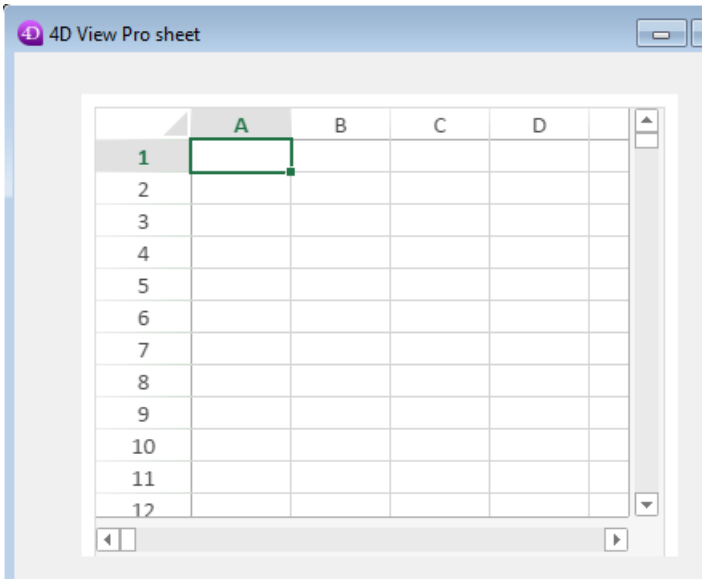


Der 4D View Pro Bereich wird in der Eigenschaftenliste über Standardeigenschaften, wie **Objektname** und **Variable oder Ausdruck**, **Koordinaten**, **Anzeige**, **Aktion u**nd **Ereignisse** konfiguriert.



- **Objektname**: Name des 4D Formularbereichs, welches das 4D View Pro Dokument enthält und anzeigt.
- **Variable oder Ausdruck**: Name der Variable für 4D View Pro Bereich

Beim Ausführen des Formulars zeigt der 4D View Pro Bereich standardmäßig eine Tabellenkalkulation:


4D View Pro sheet
A
B
C
D
1
2
3
4
5
6
7
8
9
10
11
12

4D View Pro Bereich verwenden

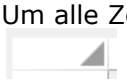
Überblick

4D View Pro Bereiche bieten beim Ausführen in Formularen grundlegende Funktionen der Tabellenkalkulation. Dazu gehören auch Bearbeiten von Zellen und Eingabe von Formeln. Komplexere Features sind über die 4D View Pro Programmiersprache verfügbar.

Auswahl, Eingabe und Navigation

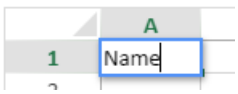
Eine Tabellenkalkulation besteht aus Zeilen und Spalten. Jeder Zeile ist eine Nummer und jeder Spalte ein Buchstabe zugewiesen (oder Buchstabengruppe, wenn die Anzahl Zeilen die Anzahl Buchstaben im Alphabet übersteigt). Die Schnittstelle einer Zeile mit einer Spalte ergibt eine Zelle. Sie können Zellen auswählen und ihren Inhalt bearbeiten.

Auswahl

- Um eine Zelle auszuwählen, klicken Sie direkt darauf oder verwenden die Pfeiltasten auf der Tastatur. In der Zelle erscheint der Inhalt oder eine Formel.
- Um mehrere zusammenhängende Zellen auszuwählen, ziehen Sie die Maus von Anfang bis Ende der gewünschten Auswahl. Sie können auch mit gedrückter **Shift**-Taste Anfang und Ende der Auswahl markieren.
- Um alle Zellen der Tabellenkalkulation auszuwählen, klicken Sie im Bereich auf die erste Zelle  links oben.
- Um eine Spalte auszuwählen, klicken Sie auf den entsprechenden Buchstaben (oder Buchstabengruppe).
- Um eine Zeile auszuwählen, klicken Sie auf die entsprechende Nummer.
- Um nicht-zusammenhängende Zellen auszuwählen, klicken Sie unter Windows mit gedrückter **Strg-Taste**, auf macOS **Befehlstaste** auf jede Zelle, die Sie auswählen wollen.
- Um die Auswahl von Zellen aufzuheben, klicken Sie einfach in der Tabellenkalkulation auf eine beliebige Stelle.

Eingabe und Navigation

Durch Doppelklick auf eine Zelle gelangen Sie in den Eingabemodus. Ist die Zelle nicht leer, wird der Cursor an das Ende des Inhalts gesetzt.



Ist eine Zelle bereits ausgewählt, lassen sich Daten direkt eingeben, selbst wenn der Cursor nicht sichtbar ist. Die Eingabe ersetzt dann den Inhalt der Zelle.

Die **Tabulatortaste** bestätigt die Eingabe und wählt die nächste Zelle rechts davon aus. Die Tastenkombination **Shift + Tab** bestätigt die Eingabe und wählt die nächste Zelle links davor aus.

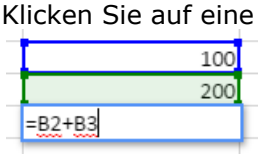
Die **Zeilenschaltung** bestätigt die Eingabe und wählt die Zelle darunter aus. Die Tastenkombination **Shift + Zeilenschaltung** bestätigt die Eingabe und wählt die Zelle darüber aus.

Mit den Pfeiltasten können Sie eine Zelle in der Pfeilrichtung verschieben.

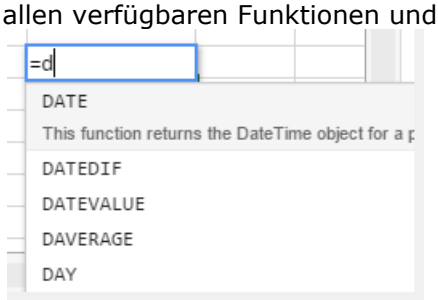
Eine Formel, Funktion oder Referenz eingeben

Um eine Formel oder Funktion in einen 4D View Pro Bereich einzugeben:

1. Wählen Sie die gewünschte Zelle aus.
2. Geben Sie **=** ein (Ist-gleich Zeichen).
3. Geben Sie die Formel ein
   ODER
   Klicken Sie auf eine Zelle, um ihre Referenz in die Formel zu übernehmen

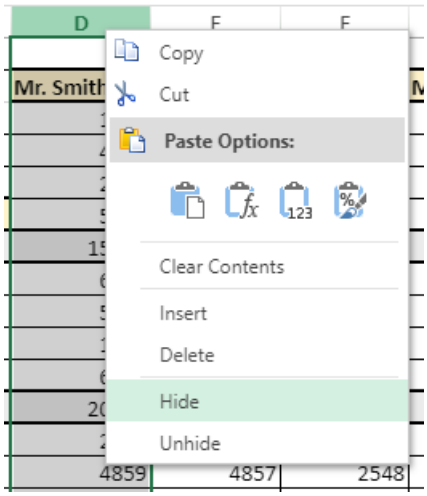


ODER
Tippen Sie den ersten Buchstaben der einzugebenden Funktion. Es erscheint ein PopUp-Menü mit allen verfügbaren Funktionen und Referenzen. Hier können Sie das gewünschte Element auswählen:



Kontextmenü

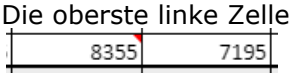
4D View Pro Bereiche enthalten ein automatisches Kontextmenü mit Standardoperationen wie Kopieren und Einsetzen, aber auch Grundfunktionen zur Tabellenkalkulation.



Hinweis: Die Menüeinträge **Kopieren/Ausschneiden** und **Einsetzen** des Kontextmenüs funktionieren nur innerhalb des 4D View Bereichs, sie haben keinen Zugriff auf die Zwischenablage des Systems. Die Tastenkürzel des Systems, wie **Ctrl+c**/**Ctrl+v** funktionieren jedoch und sind zum Datenaustausch zwischen dem 4D View Bereich und anderen Applikationen verwendbar.

Dieses Menü bietet je nach angeklicktem Bereich weitere Funktionen:

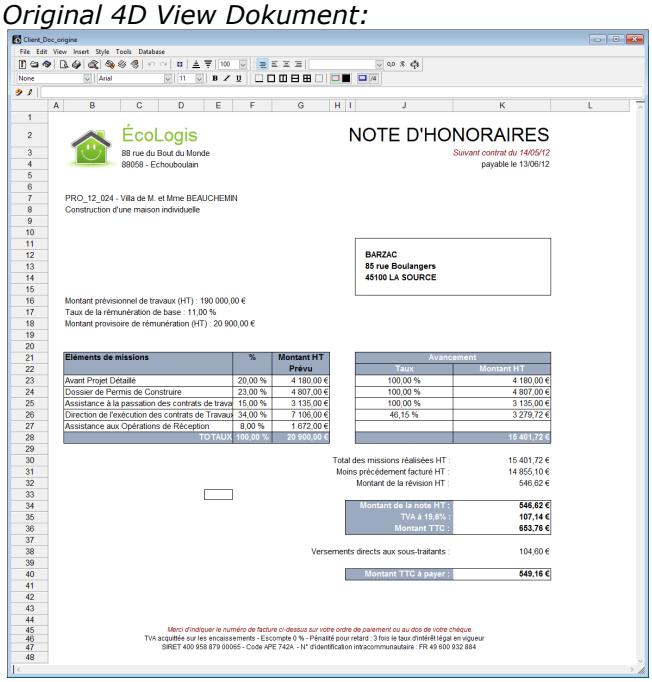
- Klick in den Kopfteil einer Spalte oder Zeile: **Einfügen**, **Löschen**, Inhalt **Einblenden/Ausblenden**
- Klick in eine Zelle oder einen Zellenausschnitt:
  - **Filter**: Damit lassen sich Zeilen ausblenden (siehe Filtering rows in der *Spread.Sheets Documentation*).
  - **Sortieren**: Sortiert den Spalteninhalt
  - **Kommentar einfügen**: Damit kann der Benutzer für einen Bereich einen Kommentar einfügen. Die oberste linke Zelle dieses Bereichs zeigt dann ein kleines rotes Dreieck:



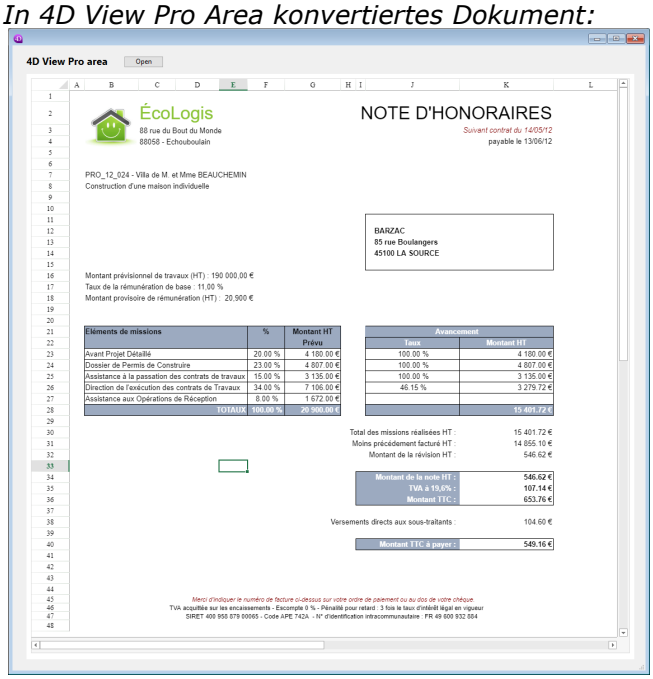
4D View Dokumente konvertieren

Über die Funktion **VP Convert from 4D View** können Sie Ihre 4D View Dokumente in 4D View Pro Bereiche konvertieren. Die meisten in 4D View Dokumenten gespeicherten Eigenschaften und Angaben werden automatisch konvertiert, wie z.B. Formate, Stilarten, Rahmen, Werte, Formeln, Auswahlen, Zoom. Im allgemeinen werden konvertierte 4D View Dokumente in 4D View Pro Bereiche genauso wie in 4D View Bereichen gerendert. Hier sehen Sie ein Beispiel:

Original 4D View Dokument:



In 4D View Pro Area konvertiertes Dokument:



Wir tun unser Bestes, damit konvertierte Dokumente so originalgetreu wie möglich bleiben. Jedoch werden u.U. ein paar Funktionalitäten nicht komplett gerendert. Ausführliche Informationen dazu finden Sie am Ende unter **Verfügbare Funktionalitäten**.

Vorgehensweise

Hinweis: Die Funktionalität zum Konvertieren eines 4D View Dokuments wird anhand von Feedback der Kunden kontinuierlich ergänzt. Deshalb sollten Sie immer eine Kopie Ihrer Original 4D View BLOBs oder

Dokumente aufbewahren, auch wenn die Konvertierung erfolgreich verlaufen ist.

Je nach dem aktuellen Status Ihres Dokuments im 4D View Plug-In Bereich müssen Sie beim Konvertieren folgendes ausführen.

1. Laden Sie Ihr 4D View Dokument (.4pv) in ein BLOB:
   Hinweis: Ist ihr 4D View Dokument bereits in einem BLOB Feld gespeichert, gehen Sie zu Schritt 2.

```
C_BLOB($pvblob)
DOCUMENT TO BLOB("document.4PV";$pvblob)
```

2. Rufen Sie **VP Convert from 4D View** mit dem BLOB auf, welches das 4D View Dokument enthält:

```
C_OBJECT($vpObj)
$vpObj:=VP Convert from 4D View($pvblob)
```

3. Weisen Sie das entstandene Objekt einem Formularobjekt oder Dokument im 4D View Pro Bereich zu, um das Ergebnis anzusehen.

```
VP IMPORT FROM OBJECT("4DViewProArea";$vpObj)
```

Verfügbare Funktionalitäten

Nachfolgende Tabelle zeigt den aktuellen Status beim Konvertieren in 4D View Pro. Beachten Sie, dass diese Übersicht regelmäßig aktualisiert wird, da der Konvertierungsprozess kontinuierlich verbessert wird.

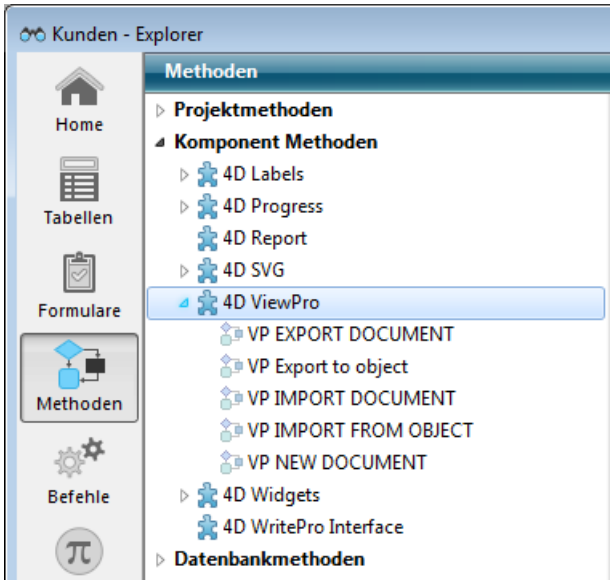
Feature	Status der Konvertierung	Kommentar
Dokumenteigenschaften	Anzeigeeigenschaften des Dokuments werden konvertiert: Ausgewählte Zellen, Zoom, Anzeige Gitter.	
	Dokumentinformation wird konvertiert: Version, Titel, Betreff, Autor, Firma, Notiz, Erstellungs- und Änderungsdatum.	
Spalten und Zeilen	Alle definierten Spalten und Zeilen werden in ihrer ursprünglichen Größe konvertiert.	
	Spalten- und Zeilentitel werden ohne Einschränkung konvertiert.	
Ränder	Ränder werden mit ihrer Dicke und Farbe konvertiert. Einstellungen zur Gitteranzeige werden konvertiert.	In 4D View Pro ist jeweils ein Modell für Rand mit einer Linie und mit doppelter Linie verfügbar. Ränder mit einer Linie werden analog zum Original konvertiert, Ränder mit doppelter Linie werden in das in 4D View Pro vorhandene Modell für doppelte Linie konvertiert.
Splitter	Werden derzeit nicht konvertiert.	
Stile & Schriften	Stile und Stilvorlagen werden konvertiert. Bedingte Stile werden nicht unterstützt.	Veraltete Textstile (z.B. Schatten, schmal) und QuickDraw Schriften werden nicht konvertiert.
	Stile für Richtung (Textausrichtung) werden derzeit nicht konvertiert	
Formate & Zellennamen	Zellenformate werden in verfügbare Formate mit ähnlichem Rendern konvertiert. Alle Formate für Datentypen werden unterstützt: Text, Zahl, Datum und Zeit, Boolean, Bild. Zellennamen werden konvertiert.	Konvertieren von Formaten kann evtl. zu unerwarteten Ergebnissen führen, da Formate sich manchmal nach Systemeinstellungen richten (*z.B.* für Dezimaltrenner) und global oder getrennt definiert sein können.
	Benutzerdefinierte 4D Formate (beginnen mit "|") werden derzeit nicht konvertiert	
	Mit PV SET CELL CONTROL angelegte Kontrollobjekte (Schaltfläche, Kontrollkästchen, Optionsfeld, DropDown Liste oder Combo Box) werden derzeit nicht unterstützt	
Bilder	Bilder werden unterstützt und mit Einschränkungen konvertiert	Das 4D View Bildformat enthielt mehrfache Codecs und ist überholt. Konvertierte Bilder behalten nur das am besten geeignete Codec für html Rendern bei (svg, png, jpeg, gif) und werden in base64 gesichert.
	Bilder im Format abgeschnitten/zentriert/wiederholt werden derzeit nicht konvertiert.	Hintergrundbilder werden nicht auf jeder Seite wiederholt (dieses Konzept existiert nicht in 4D View Pro).
Drucken	Wird derzeit nicht unterstützt	Im 4D View Dokument definierte Druckoptionen und Druckeinstellungen werden konvertiert.
Dynamische Links	Mit Feldern oder Variablen verlinkte Zellen oder Spalten werden derzeit nicht unterstützt.	
Formeln	Formeln werden konvertiert, aber Referenzen auf 4D Befehle, Methoden, Variablen und Felder werden derzeit nicht unterstützt.	

4D View Pro Programmiersprache

- Über 4D View Pro Befehle
- 4D View Pro Bereiche verwalten
- Formularereignis On VP Ready
- VP Convert from 4D View
- VP EXPORT DOCUMENT
- VP Export to object
- VP IMPORT DOCUMENT
- VP IMPORT FROM OBJECT
- VP NEW DOCUMENT

Über 4D View Pro Befehle

Das Tool 4D View Pro zur Tabellenkalkulation und für Listboxen ist als 4D Komponente integriert. Der Eintrag **4D ViewPro** erscheint deshalb im Explorer der Datenbank auf der Seite Methoden unter "Komponent Methoden":



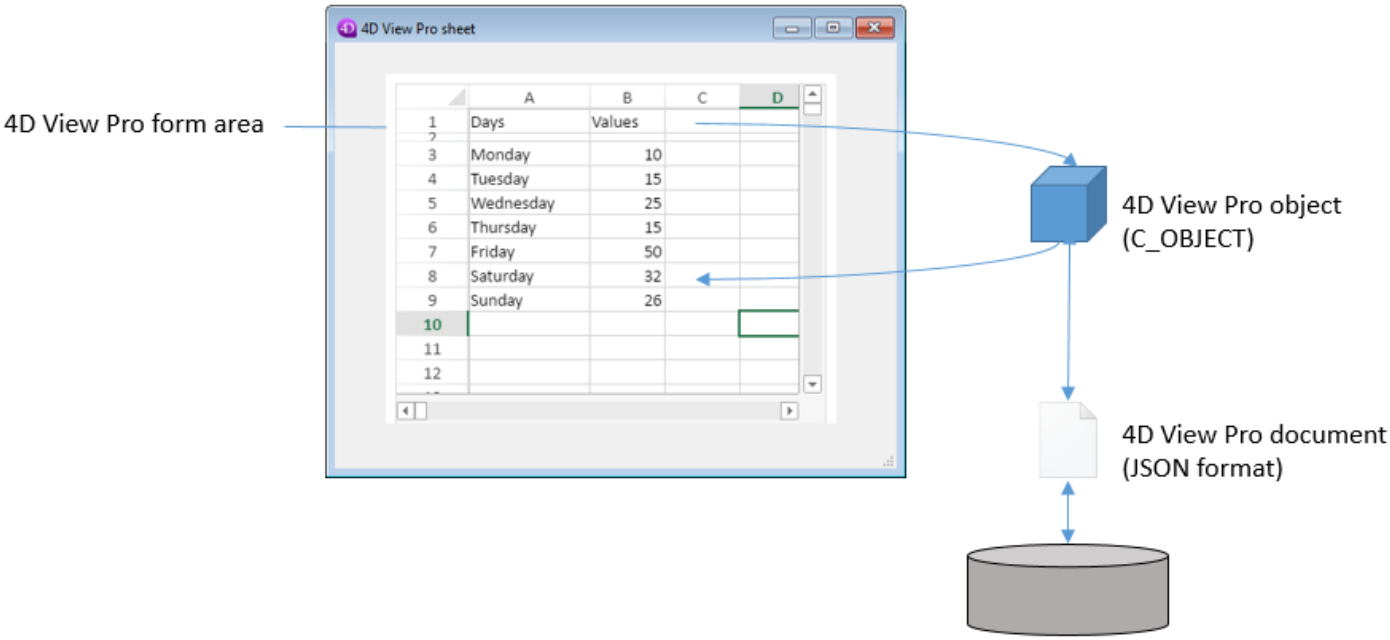
Klappen Sie diesen Eintrag auf, sehen Sie alle Befehle dieser Komponente. Sie können sie im 4D Methodeneditor genauso wie die Befehle der 4D Programmiersprache verwenden.

4D View Pro Bereiche verwalten

Architektur

Beim Arbeiten mit 4D View Pro Bereichen in Ihren Formularen können Sie verschiedene Elemente verwalten:

- **4D View Pro Area imFormular** (4D Formularobjekt): Enthält das 4D View Pro Objekt und zeigt es an. Dieser Bereich wird über einen Namen definiert (Eintrag "Objektname" in der Eigenschaftenliste).
- **4D View Pro Objekt** (Variable oder Ausdruck vom Typ **C_OBJECT**): Speichert den gesamten Inhalt der Tabellenkalkulation. Dieses Objekt weisen Sie dem 4D View Pro Formularbereich über den Eintrag "Variable oder Ausdruck" zu.
- **4D View Pro Dokument** (.4vp documents): Speichert den gesamten Inhalt der Tabellenkalkulation im JSON Format.



Wird ein 4D View Pro Objekt in einen Formularbereich geladen, erzeugt 4D nach dem kompletten Laden das Formularereignis On VP Ready. Erst in diesem Ereignis können Sie 4D View Pro Code zum Verwalten des Bereichs ausführen, sonst wird ein Fehler erzeugt.

4D View Pro Objekt

Das 4D View Pro Objekt beschreibt das Dokument und wird automatisch über die 4D View Pro Befehle verwaltet. Es enthält folgende Eigenschaften:

Eigenschaft	Typ des Wertes	Beschreibung
version	Lange Ganzzahl	Version der internen Komponente
dateCreation	Zeitstempel	Datum der Erstellung
dateModified	Zeitstempel	Datum der letzten Änderung
meta	Objekt	Freier Inhalt, reserviert für den 4D Entwickler
spreadJS	Objekt	Reserviert für die 4D View Pro Komponente

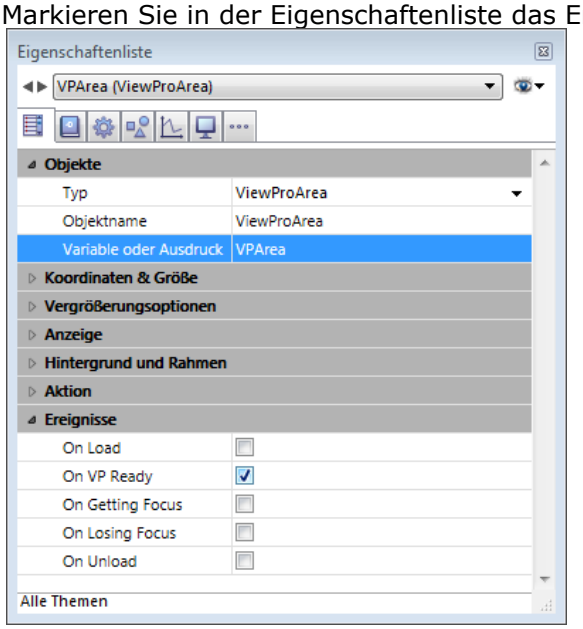
Formularereignis On VP Ready

Jeder Code zum Initialisieren des 4D View Pro Bereichs, zum Laden oder Lesen von Werten im Bereich, muss im Formularereignis On VP Ready des Bereichs liegen. **Form event** wird ausgelöst, sobald das Laden des Bereichs vollständig ist. Durch Testen dieses Ereignisses stellen Sie sicher, dass der Code in einem gültigen Kontext ausgeführt wird. Wird ein 4D View Pro Befehl aufgerufen, bevor das Formularereignis On VP Ready generiert ist, wird ein Fehler zurückgegeben.

Hinweis: 4D View Pro Bereiche werden asynchron in 4D Formulare geladen, d.h. das standardmäßige Formularereignis On load lässt sich nicht für den Code zum Initialisieren von 4D View Pro verwenden, da dieses u.U. ausgeführt wird, bevor das Laden des Bereichs abgeschlossen ist. On VP Ready wird immer nach On load generiert.

Um Initialisierungscode für einen 4D View Pro Bereich einzufügen:

1. Markieren Sie in der Eigenschaftenliste das Ereignis **On VP Ready** für den 4D View Pro Bereich:



2. Schreiben Sie den Initialisierungscode für den 4D View Pro Bereich in das Ereignis On VP Ready:

```
//Formularmethode oder Objektmethode des Bereichs
Case of
   :(Form event=On VP Ready)
      ... //Initialisierungscode
End case
```

VP Convert from 4D View

VP Convert from 4D View (4DViewDokument) -> Funktionsergebnis

Parameter	Typ		Beschreibung
4DViewDokument	BLOB	➡	4D View Dokument
Funktionsergebnis	Objekt	⮌	4D View Pro Objekt

Beschreibung

Die Funktion **VP Convert from 4D View** ermöglicht, ein bisheriges 4D View Dokument in ein 4D View Pro Objekt umzuwandeln.

Hinweis: Das bisherige Plug-In 4D View muss dafür nicht in Ihrer Umgebung installiert sein.

Im Parameter *4DViewDokument* übergeben Sie eine BLOB Variable oder Feld mit dem umzuwandelnden 4D View Dokument. Die Funktion gibt ein 4D View Pro *Objekt* zurück, in dem alle im 4D View Dokument gespeicherten Informationen in 4D View Pro Attribute konvertiert sind.

Die meisten Angaben im 4D View Dokument werden direkt konvertiert. NIcht unterstützte Eigenschaften oder solche, die beim Konvertieren verändert werden müssen, sind auf der Seite **4D View Dokumente konvertieren** aufgelistet.

Beispiel

Ein 4D View Pro Objekt aus einem 4D View Bereich erhalten, das in einem BLOB gespeichert ist:

```
C_OBJECT($vpObj)
$vpObj:=VP Convert from 4D View($pvblob)
```

VP EXPORT DOCUMENT

VP EXPORT DOCUMENT (vpAreaName ; DateiPfad)

Parameter	Typ		Beschreibung
vpAreaName	Text	➡	4D View Pro area form object name
DateiPfad	Text	➡	Pfadname des Dokuments

Beschreibung

Der Befehl **VP EXPORT DOCUMENT** exportiert das 4D View Pro Objekt, das im 4D View Pro Bereich, angegeben im Parameter *vpAreaName,* übergeben ist, gemäß dem Parameter *DateiPfad* auf die Festplatte.

In *vpAreaName* übergeben Sie die Eigenschaft Objektname des Bereichs im 4D Formular. Übergeben Sie einen ungültigen Namen, wird ein Fehler zurückgegeben.

In *DateiPfad* übergeben Sie den Zielpfad und -namen des Dokuments zum Exportieren. Übergeben Sie nur den Dokumentnamen, wird er auf derselben Ebene wie die 4D Strukturdatei gespeichert.

Die Endung des 4D View Pro Dokuments ist ".4vp". Sie wird automatisch an den Dokumentnamen angehängt, wenn sie in *DateiPfad* nicht übergeben ist.

Beispiel

Den Inhalt des Bereichs "VPArea" in ein 4D View Pro Dokument auf der Festplatte exportieren:

```
C_TEXT($docPath)

$docPath:="C:\\Bases\\ViewProDocs\\MyExport.4VP"
VP EXPORT DOCUMENT("VPArea";$docPath)
 //MyExport.4VP wird auf Ihrer Festplatte gesichert
```

VP Export to object

VP Export to object (vpAreaName) -> Funktionsergebnis

Parameter	Typ		Beschreibung
vpAreaName	Text	➡	Name des Formularobjekts 4D View Pro Area
Funktionsergebnis	Objekt	⮌	4D View Pro Objekt

Beschreibung

Die Funktion **VP Export to object** gibt das 4D View Pro Objekt zurück, das im 4D View Pro Bereich, angegeben im Parameter *vpAreaName,* übergeben ist. Mit dieser Funktion können Sie den 4D View Pro Bereich z.B. in einem Objektfeld der 4D Anwendung speichern.

In *vpAreaName* übergeben Sie die Eigenschaft Objektname des Bereichs im 4D Formular. Übergeben Sie einen ungültigen Namen, wird ein Fehler zurückgegeben.

Weitere Informationen dazu finden Sie Abschnitt **4D View Pro Objekt**.

Beispiel

Die Eigenschaft "version" des aktuellen 4D View Pro Bereichs erhalten:

```
C_OBJECT($vpAreaObj)
C_LONGINT($vpVersion)
$vpAreaObj:=VP Export to object("vpArea")
 // $vpVersion:=OB Get($vpAreaObj;"version")
$vpVersion:=$vpAreaObj.version
```

VP IMPORT DOCUMENT

VP IMPORT DOCUMENT (vpAreaName ; DateiPfad)

Parameter	Typ		Beschreibung
vpAreaName	Text	➡	Name des Formularobjekts 4D View Pro Area
DateiPfad	Text	➡	Pfadname des Dokuments

Beschreibung

Der Befehl **VP IMPORT DOCUMENT** importiert das 4D View Pro Dokument, angegeben durch *DateiPfad* im 4D View Pro Bereich *vpAreaName*. Das importierte Dokument ersetzt alle bereits im Bereich eingefügten Daten.

In *vpAreaName* übergeben Sie die Eigenschaft Objektname des Bereichs im 4D Formular. Übergeben Sie einen ungültigen Namen, wird ein Fehler zurückgegeben.

In *DateiPfad* übergeben Sie den Zielpfad und -namen des Dokuments zum Importieren. Der Befehl unterstützt nur 4D View Pro Dokumente (Endung ".4vp"). Sie müssen einen vollständigen Pfad angeben, außer das Dokument liegt auf derselben Ebene wie die Strukturdatei der Anwendung. Dann genügt es, nur den Namen zu übergeben.

Ist der Parameter *DateiPfad* ungültig, oder ist die Datei fehlerhaft formatiert bzw. fehlt ganz, wird ein Fehler zurückgegeben.

Beispiel

Beim Öffnen des Formulars ein standardmäßiges 4D View Pro Dokument von der Festplatte laden:

```
C_TEXT($docPath)
If(Form event=On VP Ready)  //Fertig geladener 4D View Pro Bereich
    $docPath:="C:\\Bases\\ViewProDocs\\MyExport.4VP"
    VP IMPORT DOCUMENT("VPArea";$docPath)
End if
```

VP IMPORT FROM OBJECT

VP IMPORT FROM OBJECT (vpAreaName ; viewPro)

Parameter	Typ		Beschreibung
vpAreaName	Text	➡	Name des Formularobjekts 4D View Pro Area
viewPro	Objekt	➡	Objekt 4D View Pro

Beschreibung

Der Befehl **VP IMPORT FROM OBJECT** importiert das 4D View Pro Objekt *viewPro* im 4D View Pro Bereich, angegeben im Parameter *vpAreaName*. Der importierte Objektinhalt ersetzt alle bereits im Bereich eingefügten Daten.

In *vpAreaName* übergeben Sie die Eigenschaft Objektname des Bereichs im 4D Formular. Übergeben Sie einen ungültigen Namen, wird ein Fehler zurückgegeben.

In *viewPro* übergeben Sie ein gültiges 4D View Pro Objekt. Es kann über **VP Export to object** oder manuell erstellt worden sein. Weitere Informationen dazu finden Sie im Abschnitt **4D View Pro Objekt**.

Ist das Objekt *viewPro* ungültig, wird ein Fehler zurückgegeben.

Beispiel

Eine Tabellenkalkulation importieren, die zuvor in einem Objektfeld gespeichert wurde:

```
QUERY([VPWorkBooks];[VPWorkBooks]ID=10)
VP IMPORT FROM OBJECT("ViewProArea1";[VPWorkBooks]SPBook)
```

VP NEW DOCUMENT

VP NEW DOCUMENT (vpAreaName)

Parameter	Typ		Beschreibung
vpAreaName	Text	➡	Name des Formularobjekts 4D View Pro Area

Beschreibung

Der Befehl **VP NEW DOCUMENT** lädt und zeigt ein neues standardmäßiges Dokument im Formularobjekt 4D View Pro Area, angegeben im Parameter *vpAreaName*. Das neue leere Dokument ersetzt alle bereits in diesen Bereich eingefügten Daten.

In *vpAreaName* übergeben Sie die Eigenschaft Objektname des Bereichs im 4D Formular. Übergeben Sie einen ungültigen Namen, wird ein Fehler zurückgegeben.

Beispiel

Ein leeres Dokument im Formularobjekt "myVPArea" anzeigen:

```
VP NEW DOCUMENT("myVPArea")
```

Formeln in 4D View Pro

Überblick

4D View Pro Funktionen werden in Formeln verwendet. Jede 4D View Pro Formel ist ein Ausdruck, der einen Wert zurückgibt. Alle Ausdrücke bestehen aus Operanden und Operatoren:

- **Operatoren:** siehe **Operatoren und Werte**
- **Operanden** sind in verschiedene Kategorien unterteilt:
  - Werte
  - Referenzen auf andere Zellen (relativ, absolut, gemischt oder über Name)
  - 4D Variablen, Felder und Funktionen
  - 4D View Pro Funktionen

Zum Eingeben einer Formel:

1. Wählen Sie die Zelle zum Eintragen der Formel oder Funktion
2. Geben Sie das Zeichen = (ist gleich) ein
3. Geben Sie die Formel ein und drücken dann die Eingabetaste

In 4D View Pro gibt es eine große Anzahl an Formeln. Dieser Abschnitt zeigt die wesentlichen Formeln. Die komplette Liste der von 4D View Pro unterstützten Formeln finden Sie unter der Spreadsheets documentation.

- ROW
- SIN
- SQRT
- STDEV.P
- SUBSTITUTE
- SUM
- TAN
- TEXT
- TODAY
- TRUE
- TYPE
- VAR.P

Operatoren und Werte

Operatoren nach Datentyp

4D View Pro unterstützt fünf Datentypen. Für jeden Datentyp gibt es spezifische Wertangaben und Operatoren.

Datentyp	Wert	Operatoren
Zahl	1.2 1.2 E3 1.2E-3 10.3x	+ (Addition) - (Subtraktion) * (Multiplikation) / (Division) ^ (Exponent -- eine Potenz setzen) % (Prozent -- die Zahl vor dem Operator durch hundert dividieren)
Datum	10/24/2017	+ (Datum + Anzahl Tage -> Datum) + (Datum + Zeit -> Datum + Tageszeit) - (Datum - Anzahl Tage -> Datum) - (Datum - Datum -> Anzahl Tage zwischen beiden)
Zeit	10:12:10	Operatoren für Zeitspanne: + (Addition) - (Subtraktion) * (Dauer * Zahl -> Zeitspanne) / (Dauer / Zahl -> Zeitspanne)
String	'Sophie' oder "Sophie"	& (Verbindung)
Boolean	TRUE oder FALSE	-

Vergleichsoperatoren

Folgende Operatoren lassen sich mit zwei Operanden vom gleichen Typ verwenden:

Operator	Vergleich
=	ist gleich
<>	ist ungleich
>	größer als
<	kleiner als
>=	größer als oder gleich
<=	kleiner als oder gleich

Rangfolge der Operatoren

Liste der Operatoren sortiert nach Vorrang:

1. ()
2. -
3. +
4. %
5. ^
6. * /
7. + -
8. &
9. = > < >= <= <>

Rangfolge der Operanden in Formeln

Haben zwei oder mehr Operanden den gleichen Namen, bestimmt 4D View Pro den Typ jedes Elements mit folgender Rangfolge

Rangfolge	Elementtyp
1	Referenz der Zelle
2	Name der Zelle
3	4D View Pro Funktion
4	Projektmethode
5	4D Befehl
6	Variable

Zellenreferenzen

Formeln verweisen oft auf andere Zellen über Zellenreferenzen. Sie können solche Formeln in andere Zellen kopieren. Kopieren oder setzen Sie diese Zellen an eine andere Position, kann sich jede Zellenreferenz in dieser Formel je nach definierter Referenz ändern oder gleich bleiben. Eine Referenz, die sich ändert, wird relative Referenz genannt. Sie bezieht sich auf eine Zelle, wie weit sie links/rechts und oben/unten von der Zelle mit der Formel entfernt ist. Eine Referenz, die immer auf eine bestimmte Zelle zeigt, wird absolute Referenz genannt.

Sie können auch eine gemischte Referenz erstellen, die immer auf eine feste Zeile oder Spalte zeigt.

Beispiel: In Zelle C8 steht die Formel = C6 + C7. Sie addiert die Werte in den beiden darüberliegenden Zellen und zeigt das Ergebnis. Diese Formel bezieht sich auf die Zellen C6 und C7, d.h. 4D View Pro erhält die Anweisung, diese beiden Zellen für Werte in der Formel zu verwenden.

Referenzarten

Verwenden Sie nur Zellenkoordinaten, z.B. C5, interpretiert 4D View Pro die Referenz als relativ. Für eine absolute Referenz setzen Sie ein Dollarzeichen vor den Buchstaben und die Nummer, also C5.

Sie können auch absolute and relative Referenz mischen. Dazu setzen Sie das Dollarzeichen nur vor den Buchstaben oder die Nummer, z.B. $C5 oder C$5. Mit einer gemischten Referenz können Sie die Zeile oder Spalte als absolut festlegen, während der jeweils andere Teil relativ ist.

Ein einfacher und schneller Weg für eine absolute Referenz ist, der Zelle einen Namen zu geben und diesen Namen anstelle der Zellenreferenz zu verwenden. Eine Referenz auf eine Zelle mit Namen ist immer absolut.

Diese Übersicht zeigt die Auswirkung der verschiedenen Arten:

Beispiel	Art der Referenz	Beschreibung
C5	Relativ	Referenz verweist auf die relative Position von Zelle C5, wo die Referenz zuerst benutzt wird.
C5	Absolut	Referenz ist absolut. Verweist immer auf Zelle C5, egal, wo sie verwendet wird.
$C5	Gemischt	Referenz ist immer auf Spalte C, Referenz auf Zeile ist relativ zur Position der Zelle, wo die Referenz zuerst benutzt wird.
C$5	Gemischt	Referenz ist immer auf Zeile 5, Referenz auf Spalte ist relativ zur Position der Zelle, wo die Referenz zuerst benutzt wird.
Zellenname	Absolut	Referenz ist absolut. Bezieht sich immer auf die genannte Zelle, unabhängig, wo die Referenz benutzt wird.

Formeln vom Plug-In 4D View konvertieren

Überblick

Wie auf der Seite **4D View Dokumente konvertieren** bereits erläutert ist, lassen sich die meisten Eigenschaften und Angaben in Dokumenten aus dem Plug-In 4D View in Dokumente in 4D View Pro übernehmen.

Auch Formeln werden konvertiert. Jedoch unterscheiden sich die Formularsprachen von 4D View und 4D View Pro in ein paar Punkten. Es gibt drei Fälle der Kompatibilität:

- Ein 4D View Feature (Operator, Konstante, Funktion) ist genau dasselbe wie in 4D View Pro: In diesem Fall ist die Konvertierung transparent.
- Ein 4D View Feature wird in 4D View Pro unterstützt, jedoch über eine unterschiedliche Funktion oder Operator: In diesem Fall wird die Anpassung automatisch durchgeführt
- Ein 4D View Feature wird teilweise oder nicht in 4D View Pro unterstützt: In diesem Fall müssen Ihre konvertierten Formeln angepasst werden, damit sie wie erwartet funktionieren.

Nachfolgende Tabellen zeigen die Funktionalitäten in 4D View Formeln und ihre Entsprechung in 4D View Pro Formeln.

Operatoren

	4D View	4D View Pro
Numerische Operatoren		
Addition	+	+
Substraktion	-	-
Multiplikation	*	*
Division	/	/
Restwert	\	MOD
Ganzzahlige Division	÷	TRUNC(a/b)
Exponent	^	^
Modulo	%	%
Boolean Operatoren		
AND	&	**AND**
OR	|	**OR**
Not	~	**NOT**
String Operatoren		
Verbindung	+	&
Ersetzung	-	**SUBSTITUTE**, ex: "Down Trend" - "Down" is replaced by SUBSTITUTE("Down Trend","Down","")
Position	\	FIND (unterscheidet Klein- und Großschreibung) oder SEARCH (keine Unterscheidung zwischen Klein- und Großschreibung)
Datumsoperatoren		
Datum+Tage->Datum	+	+
Datum+Zeit->Datum+Tageszeit	+	+
Datum-Tage->Datum	-	-
Datum-Datum->Anzahl Tage	-	-
Zeitoperatoren		
Addition	+	+
Substraktion	-	-
Multiplikation	*	*
Division	/	/
Vergleichsoperatoren		
ist gleich	=	=
ist ungleich	#	<>
größer als	>	>
kleiner als	<	<
größer als oder gleich	>=	>=
kleiner als oder gleich	<=	<=

Funktionen

4D View	4D View Pro	Kommentar
Abs	**ABS**	
AddToDate	DATE	AddToDate(date;years;months;days) wird ersetzt durch **DATE**(YEAR(date)+years,MONTH(date)+months,DAY(date)+days)
date+time	TIME	DATE(date) + **TIME**(time)
And	**AND**	
ArcCos	**ACOS**	
ArcSin	**ASIN**	
ArcTan	**ATAN**	
Area	-	n/a (es gibt keinen Plug-In Bereich mehr)
Average	**AVERAGE**	
Cell	**INDIRECT**	
Column	**COLUMNLETTER**	COLUMN gibt eine Nummer zurück (keinen Buchstaben)
Cos	**COS**	
Count	**COUNTA**	
CurrentDate	**TODAY**	
CurrentTime	**NOW**	
CVCompound	**PV**	CVCompound(1%;5;1000) wird ersetzt durch **PV**(1%,5,-1000)
CVSimple	**PV**	CVSimple(1%;5;5*1000) wird ersetzt durch **PV**(1%,5,,-1000) -- Achten Sie auf die beiden aufeinanderfolgenden Kommas
Empty	**ISBLANK**	
Eval4D	-	Derzeit nicht verfügbar
Exp	**EXP**	
False	**FALSE**	
Find	**LOOKUP**	
FindCell	**FINDCELL**	
FVCompound	**FV**	FVCompound(1%;35;35*1000) wird ersetzt durch **FV**(1%,35,-1000)
FVSimple	**FV**	FVSimple(12%;35;35*1000) wird ersetzt durch **FV**(12%,35,,-35*1000) -- Achten Sie auf die beiden aufeinanderfolgenden Kommas
If	**IF**	
Length	**LEN**	
Log	**LN**	
Max	**MAX**	
Min	**MIN**	
MonthlyValue	**PMT**	MonthlyValue(10.5%/12,48,6500) wird ersetzt durch **PMT**(10.5%/12,48,-6500)
Not	**NOT**	
Or	**OR**	
PeriodNumber1	**NPER**	PeriodNumber1(10.5%/12;166.42;6500) wird ersetzt durch **NPER**(10.5%/12,-166.42,6500)
PeriodNumber2	**NPER**	PeriodNumber2(10.5%/12,5000,3000) wird ersetzt durch **NPER**(10.5%/12,,3000,-5000) -- Achten Sie auf die beiden aufeinanderfolgenden Kommas
Pi	**PI**	
Random	**RAND**	
Range	**INDIRECT**	Range("A1";"A3") wird ersetzt durch **INDIRECT**("A1:A3") -- Achten Sie auf das Spaltenzeichen zwischen A1 und A3
Rate1	**RATE**	Rate1(5;1000;3000) wird ersetzt durch **RATE**(5,-1000,3000)
Rate2	**RATE**	Rate2(5,6000,2800) wird ersetzt durch **RATE**(5,,2800,-6000) -- Achten Sie auf die beiden aufeinanderfolgenden Kommas
Rounding	**ROUND**	
Row	**ROW**	
Sin	**SIN**	
SquareRoot	**SQRT**	
StdDeviation	**STDEV.P**	
String	**TEXT**	

SubString	**MID**	
Sum	**SUM**	
Tan	**TAN**	
True	**TRUE**	
Type	**TYPE**	In 4D View Pro zurückgegebene Typen unterscheiden sich von 4D View
Variance	**VAR.P**	

ABS

ABS (value)

Parameter	Typ		Beschreibung
value	Zahl, Ausdruck	➡	Number whose absolute value is returned

Beschreibung

The **ABS** function calculates the absolute value(s) of the specified *value*. If *value* is negative, a positive value will be returned. It accepts numeric data as values or expressions and returns numberic data.

Beispiel

```
ABS(-6) //result:= 6

ABS(16-26) //result:= 10

ABS(6) //result:= 6
```

ACOS

ACOS (value)

Parameter	Typ		Beschreibung
value	Zahl	⇨	Angle whose arccosine is returned. Must be between –1 and +1.

Beschreibung

The **ACOS** function calculates the angle of the arccosine specified in *value*. *value* must be included in the range -1 to +1.

The returned angle is in radians between 0 and PI . To convert the result to degrees, multiply the result by 180/PI.

Beispiel

```
ACOS(0.5) //result:= 1.0471975512
```

AND

AND (LogicalValue {, LogicalValue2 , ... , LogicalValueN})

Parameter	Typ		Beschreibung
LogicalValue	Boolean, Zahl, Ausdruck	⇨	Value(s) to evaluate

Beschreibung

The **AND** function returns TRUE if all arguments are true; otherwise, it returns FALSE if at least one argument is false.

It accepts boolean values as numeric (0 or 1) or logical expressions (TRUE or FALSE) for up to 255 arguments. You can also specify a single array instead of listing the values separately, or up to 255 arrays. You can also specify the *logicalValue* as an expression.

Beispiel

```
AND(D12,E12)

AND(D2:D12)

AND(5+3=8,5+1=6) //TRUE

AND(1,TRUE) //TRUE
```

ASIN

ASIN (value)

Parameter	Typ		Beschreibung
value	Zahl	⇨	Angle whose sine is returned. Must be between –1 and +1.

Beschreibung

The **ASIN** function calculates the arcsine, the angle whose sine is specified in *value*. It accepts and returns numeric data.

In *value*, specify the sine of the angle. The sine must be a value between -1 and +1.

The angle is returned in radians between -PI/2 and PI/2. To convert the result to degrees, multiply it by 180/PI.

Beispiel

```
ASIN(0.5) //0.5235987756
```

ATAN

ATAN (value)

Parameter	Typ		Beschreibung
value	Zahl	⇨	Tangent of an angle. Must be between –1 and +1.

Beschreibung

The ATAN function calculates the arctangent, i.e. the angle whose tangent is specified in *value*. It accepts and returns numeric data.

In *value*, specify the tangent of the angle to return. It must be between -1 and +1.

The angle is returned in radians between -PI/2 and PI/2. To convert the result to degrees, multiply the result by 180/PI.

Beispiel

```
ATAN(1) //result:= 0.7853981634
```

AVERAGE

AVERAGE (value {, value2 , ... , valueN})

Parameter	Typ		Beschreibung
value	Zahl, Array	➡	Number(s) whose mean is to be calculated

Beschreibung

The **AVERAGE** function calculates the average of *value*.
In *value*, you can pass:

- real or longint values,
- a range or several ranges of cells (array).

Up to 255 arguments may be included.

Beispiel

```
AVERAGE(98,72,85) //result=85

AVERAGE(A1,B3,D5,E9,L8,L9)

AVERAGE(R1C1,R3C2)

AVERAGE(A1:A9)

AVERAGE(A1:A9,B1:B9,D5:D8)
```

COLUMNLETTER

COLUMNLETTER ({reference})

Parameter	Typ		Beschreibung
reference	CellRef	➡	A cell or a range of cells

Beschreibung

The **COLUMNLETTER** function returns the column letter of *reference*.
reference can be a cell or a range of cells. If the *reference* argument is omitted, the default argument is the reference of the cell in which the **COLUMNLETTER** function is placed.

Beispiel

```
COLUMNLETTER(A9) //A

COLUMNLETTER(B1:B5) //B

COLUMNLETTER() //current column letter
```

COS

COS (value)

Parameter	Typ		Beschreibung
value	Zahl	⇨	Angle whose cosine is returned

Beschreibung

The **COS** function returns the cosine of the angle specified in *value*. It accepts and returns numeric data.
In *value*, pass any real number (an angle) for which to return the cosine. It must be expressed in radians. If the angle is in degrees, multiply it by PI/180 to convert it to radians.

Beispiel

```
COS(45*PI()/180) //0.7071067812
```

COUNTA

COUNTA (value {, value2 , ... , valueN})

Parameter	Typ		Beschreibung
value	Zahl, Array	➡	Cells or cell range to be counted

Beschreibung

The **COUNTA** function counts the number of cells specified in *value* that are not empty *(i.e.* cells that contain numbers, text, or logical values). It accepts cell references and returns numeric data.

In *value*, you can pass up to 255 separate cells or a single array of values.

Beispiel

```
COUNTA(B2,D2,E4,E5,E6)

COUNTA(A1:G5)
```

EXP

EXP (value)

Parameter	Typ		Beschreibung
value	Zahl	⇨	Number to evaluate

Beschreibung

The **EXP** function returns the natural log base (e = 2.71828...) raised to the power of the number specified in *value*. It accepts and returns numeric data.

This function is the inverse of **LN**, so EXP(LN(x)) results in x.

Beispiel

```
EXP(B3)

EXP(1)  //2.17828…
```

FALSE

FALSE ()
Dieser Befehl benötigt keine Parameter

Beschreibung

The **FALSE** function returns the logical FALSE value (0).

Beispiel

```
NOT(FALSE) //TRUE
```

FINDCELL

FINDCELL (toFind , searchRange)

Parameter	Typ		Beschreibung
toFind	CellRef	⇒	Value to find
searchRange	CellRef	⇒	Cells to search

Beschreibung

The **FINDCELL** function searches for the *toFind* value in the *searchRange* range of cells and returns the reference of the cell in which it was found. This reference cannot be displayed, but can be used by other 4D View Pro functions that accept a cell reference (*CellRef*) as parameter.

toFind must contain the reference of a cell that actually contains the value to find.

Beispiel

Assuming cell C3 contains 10:

```
FINDCELL(C3,A1:B9) //returns 10 if the value is actually found in the A1:B9 cell range, otherwise it returns an error.
```

FV

FV (i , n , m {, f})

Parameter	Typ		Beschreibung
i	Zahl	➡	The interest rate for a period
n	Zahl	➡	The number of periods
m	Zahl	➡	For compound interest: the monthly payment at the end of each period (use a negative value). For single interest: pass an empty parameter (see example)
f	Zahl	➡	For single interest rate: the final value at the end of a period (use a negative value)

Beschreibung

The **FV** function calculates:

- the final value of a sum using compound interest, or
- the final value of a sum using single interest

To calculate the value acquired during an investment, if the monthly payments are paid at the end of the period, pass the *m* parameter and omit the *f* parameter. Here is the formula for this calculation:

$$FV(i,n,m) = m \times \frac{(1+i)^n - 1}{i}$$

To calculate the final value of a sum using single interest, pass the *f* parameter and pass an empty parameter (,,) for the *m* placeholder. Here is the formula for this calculation:

$$FV(i,n,,f) = f \times (1+i)^n$$

Beispiel 1

Compound interest: you plan on depositing €1,000 each month in a savings account, which earns you 12% annual interest, for 35 months.

```
FV(1%,35,-1000) //41660.275603126
```

Beispiel 2

Single interest rates: same type of scenario as above.

```
FV(12%,35,,-35*1000) //1847986.69
```

IF

IF (valueTest , valueTrue , valueFalse)

Parameter	Typ		Beschreibung
valueTest	Ausdruck	➡	Value or expression to evaluate
valueTrue	Ausdruck	➡	Value to return if the test evaluates to true
valueFalse	Ausdruck	➡	Value to return if the test evaluates to false or 0

Beschreibung

The **IF** function performs a comparison and returns one of two provided values based on that comparison. It accepts numeric (boolean) data and returns any data type.

The value of *valueTest* is evaluated. It must be or evaluate to numeric data, where non-zero values indicate TRUE, and a value of zero indicates FALSE. It may contain one of the relational operators: greater than (>), less than (<), equal to (=), or not equal to (<>). If *valueTest* is:

- not zero (TRUE), then *valueTrue* is returned.
- zero (FALSE), then *valueFalse* is returned.

Beispiel

You want to evaluate B1, giving the value of sales.

```
IF(B1<200,"Declining result","Good result") //"Good result" is written if B1>200
```

INDIRECT

INDIRECT (cell | cellRange)

Parameter	Typ	Beschreibung
cell | cellRange	CellRef ➡	Reference to a cell, a name defined as a reference, or a text string reference to a cell or to a range

Beschreibung

The **INDIRECT** function returns the contents of *cell*. The *cell* parameter (mandatory) can be any cell reference, including absolute reference such as A1 or a character string. Use **INDIRECT** when you want to change the reference to a cell within a formula without changing the formula itself.

The **INDIRECT** function can also return the internal reference of a range of cells (reference cannot be displayed but can be used by other 4D View Pro functions).

Beispiel

```
INDIRECT("A1") //returns the contents of cell A1
```

```
COLUMN(INDIRECT("A1:A3")) //column 1
ROW(INDIRECT("A1:F1")) //row 1
```

ISBLANK

ISBLANK (value)

Parameter	Typ		Beschreibung
value	CellRef, Ausdruck, Zahl, Text	➡	Value to evaluate

Beschreibung

The **ISBLANK** function tests if the contents of a cell, a number, a text, or any expression, is empty. This function returns TRUE if the value refers to an empty cell or to no data.

Note: Cells containing an empty string ("") are considered as blank.

Beispiel

```
IF(ISBLANK(A1);”Error”;0) //"Error" if cell A1 is empty

ISBLANK(B1)

ISBLANK(A4-52)

ISBLANK(4) //FALSE
```

LEN

LEN (value)

Parameter	Typ		Beschreibung
value	Text, CellRef	➡	Text whose character length to count

Beschreibung

The **LEN** function returns the number of characters in the *value* string.

In *value*, pass the text whose length you want to find. It must be a string or a cell reference to a string value.

Note: Spaces count as characters

Beispiel

```
LEN("4D, Inc.") //8
```

LN

LN (value)

Parameter	Typ		Beschreibung
value	Zahl	➡	Number greater than 0 to evaluate

Beschreibung

The **LN** function returns the natural logarithm of *value*. It accepts and returns numeric data.
In *value*, pass a positive number (greater than zero).
Note: **LN** is the inverse of **EXP**, so LN(EXP(x)) is x.

Beispiel

```
LN(10) //2.30258509...

LN(EXP(1)) //1
```

LOOKUP

LOOKUP (toFind , intervalToSearch , returnInterval)

Parameter	Typ		Beschreibung
toFind	CellRef, Text, Zahl, Boolean	➡	Value to find
intervalToSearch	CellRef	➡	Cell range to search
returnInterval	CellRef	➡	Cell range to find corresponding value

Beschreibung

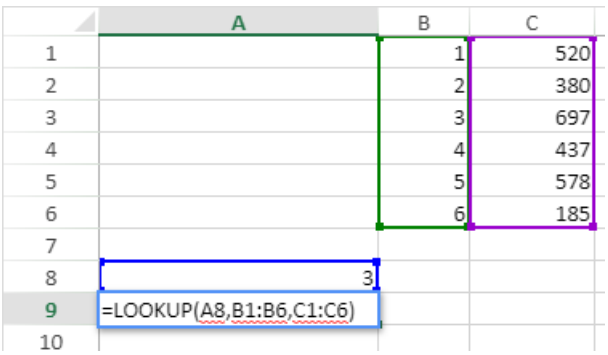
The **LOOKUP** function searches for the value *toFind* in the *intervalToSearch* cell range and returns the corresponding value used in the *returnInterval* range.

toFind must contain the reference of a cell that actually contains the value to find.

intervalToSearch must be sorted in ascending order since **LOOKUP** uses the first value higher or equal to the value set as *toFind*.

If *toFind* cannot be found, it matches the largest value in *intervalToSearch* that is less than or equal to *toFind*.

Beispiel



"3", located in the A8 cell, is the value to find. B1:B6 is the interval to search. C1:C6 is the return interval. The B3 cell contains the value to find. The corresponding value in the return interval is in the C3 cell, that is, "697".

MAX

MAX (value {, value2 , ... , valueN})

Parameter	Typ		Beschreibung
value	Zahl, Array	⇨	Number or numeric array to evaluate

Beschreibung

The **MAX** function returns the greatest (maximum) value of *value*. It accepts and returns numeric data.
In *value*, pass the values to evaluate. Each argument can be a number or an array of numbers. You can use a single array (cell range) or multiple arrays (cell ranges).
If an array or reference contains text, logical values, or empty cells, **MAX** ignores those values; however, cells with the value zero are included in calculations.

Beispiel

MAX(A1,B2,C3,D4,E5)

MAX(A1:A9)

MAX(2,15,12,3,7,19,4) //19

MID

MID (value , startFrom {, numChars})

Parameter	Typ		Beschreibung
value	Text, CellRef, Ausdruck	➡	Text containing the characters to extract
startFrom	Zahl	➡	Number designating the first character to extract in text
numChars	Zahl	➡	Number of characters to return

Beschreibung

The **MID** function returns the requested number of characters from *value* starting at the position specified in *startFrom*. It accepts text data for *value* and numeric data for *startFrom* and *numChars*. It returns text data.

In *value*, pass the text string containing the characters you want to extract. It can be a string, a formula that returns a string, or a reference to a cell containing a string.

In *startFrom*, pass a number representing the first character you want to extract in text, with the first character in the text having a value of one (1); if not an integer, the number is truncated. It can be a string, a formula that returns a string, or a reference to a cell containing a string. If *startFrom*:

- is greater than the length of *value*, an empty string (" ") is returned
- is less than the length of *value*, but *startFrom* + *numChars* exceeds the length of *value*, the characters up to the end of text are returned.

In *numChars*, pass the number of characters to return from *value*; if an integer is not specified, the number is truncated.

Beispiel

```
MID(B17,5,8)

MID("hello world",7,20) //"world"
```

MIN

MIN (value {, value2 , ... , valueN})

Parameter	Typ		Beschreibung
value	Zahl, Array	⇨	Number or numeric array to evaluate

Beschreibung

The **MIN** function returns the lowest (minimum) value of *value*. It accepts and returns numeric data.

In *value*, pass the values to evaluate. Each argument can be a number, or an array of numbers (a single array (cell range) or multiple arrays (cell ranges)).

If an array or reference contains text, logical values, or empty cells, **MIN** ignores those values; however, cells with the value zero are included in calculations.

Beispiel

```
MIN(2,15,12,3,7,19,4) //2
```

NOT

NOT (logicalValue)

Parameter	Typ		Beschreibung
logicalValue	Boolean, Zahl, Ausdruck	➡	Value to evaluate

Beschreibung

The **NOT** function reverses the logical value of its parameter.
logicalValue can be a boolean or a number. If its value is zero, then the function returns TRUE. If it is a value other than zero, then the function returns FALSE.

Beispiel

```
NOT(A3)

NOT(D5>100)

NOT(0) //TRUE

NOT(TRUE) //FALSE

NOT(12) //FALSE
```

NOW

NOW ()
Dieser Befehl benötigt keine Parameter

Beschreibung

The **NOW** function returns the time of the current date. It returns a DateTime object.
This function is updated when the spreadsheet or cell containing the function is recalculated.

Beispiel

If today is Monday 8 January 2018 at 11:25:42:

```
NOW() //1/8/2018 11:25:42
```

NPER

NPER (i , m , p {, f})

Parameter	Typ		Beschreibung
i	Zahl	➡	The interest rate for a period
m	Zahl	➡	The monthly payment paid at the end of the period (use a negative value). For acquired value: pass an empty parameter (see example)
p	Zahl	➡	The current value of the loan
f	Zahl	➡	The future value of the loan (use a negative value)

Beschreibung

The **NPER** function returns the number of periods needed to reimburse a loan. Two formulas can be used:

- First formula, when you know the monthly payment:
  $$\mathrm{NPER}(i,m,p) = \frac{\log((m - 1 \times p)/m)}{\log(1+i)}$$
- Second formula, when you know the total payment:
  $$\mathrm{NPER}(i,,f,p) = \frac{\log(f/p)}{\log(1+i)}$$

Beispiel 1

You borrowed 6,500 euros with 10.5% annual interest and you reimburse 166.42 euros per month:

```
NPER(10.5%/12,-166.42,6500) //48
```

Beispiel 2

You borrowed 3,000 euros with 10.5% annual interest and you know that the total monthly payments will be 5,000 euros:

```
NPER(10.5%/12,,3000,-5000) //58
```

OR

OR (logicalValue {, logicalValue2 , ... , logicalValueN})

Parameter	Typ		Beschreibung
logicalValue	Boolean, Array, Ausdruck	➡	Value(s) to evaluate

Beschreibung

The **OR** function returns False if the evaluation of all parameters is false. It returns True if the evaluation of at least on the parameters is true.

The function accepts boolean values as numeric (0 or 1) or logical expressions (TRUE or FALSE) for up to 255 arguments. You can also specify a single array instead of listing the values separately, or up to 255 arrays. You can also specify the *logicalValue* as an expression.

Beispiel

```
OR(B3,B6,B9)

OR(D2:D12)

OR(TRUE,FALSE,FALSE) //TRUE

OR(TRUE()) //TRUE

OR(FALSE(),FALSE()) //FALSE

OR(1+1=1,2+2=5) //FALSE

OR(5+3=8,5+4=12) //TRUE
```

PI

PI ()
Dieser Befehl benötigt keine Parameter

Beschreibung

The **PI** function returns the value of Pi as 3.1415926536.

PMT

PMT (i , n , p)

Parameter	Typ		Beschreibung
i	Zahl	➡	The interest rate for a period
n	Zahl	➡	The number of periods
p	Zahl	➡	The current value of the loan (use a negative value)

Beschreibung

The **PMT** function returns the value of monthly loan payments.
Here is the formula for **PMT**:

$$PMT(i,n,p) = \frac{i}{1-(1+i)^{-n}}$$

Beispiel

You borrowed 6,500 euros over 48 months with 10.5% interest:

```
PMT(10.5%/12,48,-6500) //166.42
```

PV

PV (i , n , m {, f})

Parameter	Typ		Beschreibung
i	Zahl	➡	The interest rate for a period
n	Zahl	➡	The number of periods
m	Zahl	➡	For compound interest: the monthly payment at the end of each period (use a negative value). For single interest: pass an empty parameter (see example)
f	Zahl	➡	For single interest rate: the final value at the end of a period (use a negative value)

Beschreibung

The **PV** function calculates:

- the current value of a sum using the compound interest, or
- the current value of a sum using single interest rates.

To calculate the current value of a sum using the compound interest, pass the *m* parameter and omit the *f* parameter. Here is the formula for this calculation:

$$PV(i,n,m) = m \times \frac{1-(1+i)^{-n}}{i}$$

To calculate the current value of a sum using single interest rates, pass the *f* parameter and pass an empty parameter (,,) for the *m* placeholder. Here is the formula for this calculation:

$$PV(i,n,,f) = \frac{f}{(1+i)^{n}}$$

Beispiel 1

Compound interest: you have a loan with an 12% annual interest rate (thus 1% per month) over 5 months with a monthly payment of €1,000.

```
PV(1%,5,-1000) //4853,4312393251
```

Beispiel 2

Single interest rates: you have a loan with an 12% annual interest rate (thus 1% per month) over 5 months with a monthly payment of €1,000.

Note: Pay attention to the double ",," inside the syntax.

```
PV(1%,5,,-5*1000) //4757,328438033744
```

RAND

RAND ()
Dieser Befehl benötigt keine Parameter

Beschreibung

The **RAND** function returns a random number between 0 and 0,9999999...

Beispiel

```
RAND()
RAND()*100
INT(RAND()*100)
```

RATE

RATE (n , m , p {, f})

Parameter	Typ		Beschreibung
n	Zahl	➡	The number of periods
m	Zahl	➡	The monthly payment paid at the end of the period (use a negative number). For future value: pass an empty parameter (see example)
p	Zahl	➡	The current value of the loan
f	Zahl	➡	The future value (use a negative number)

Beschreibung

The **RATE** function returns the interest rate corresponding to the values passed in parameters. Two formulas can be used:

- First formula, when you know the monthly payment paid at the end of the period

$$i = \frac{m \times (1 - (1 + i)^{-n})}{p}$$

- Second formula, when you know the acquired value:

$$u_{n+1} = \frac{m \times (1 - (1 + u_n)^{-n})}{p}$$

Beispiel 1

You borrowed 3,000 euros and your monthly payments are 1,000 euros over 5 months:

```
RATE(5,-1000,3000) //0.19
```

Beispiel 2

You borrowed 2,800 euros and the total paid value is 6,000 euros over 5 months:

```
RATE(5,,2800,-6000) //0.16
```

ROUND

ROUND (value , places)

Parameter	Typ		Beschreibung
value	Zahl	➡	Number to round
places	Zahl	➡	Number of decimal places

Beschreibung

The **ROUND** function rounds the specified *value* to the nearest number, using the specified number of decimal *places*.

Use the *value* parameter to specify the number to round. You can pass any numeric data (or cell reference containing numeric data).

Note: The result may be rounded up or rounded down.

Use the *places* parameter to specify the number of decimal places. The places argument has these rules:

- Set places to a value >0 to round to the specified number of decimal places.
- Set places to zero to round to the nearest whole number.
- Set places to a value <0 to round the value left of the decimal to the nearest ten, hundred, etc.

Beispiel

```
ROUND(A3,-2)

ROUND(C4,B2)

ROUND(PI(),5) //3.14159

ROUND(29.2,-2) //0 because 29.2 is closer to 0 than to 100.

ROUND(-1.963,0) //-2
```

ROW

ROW ({reference})

Parameter	Typ		Beschreibung
reference	CellRef	➡	A cell or a range of cells

Beschreibung

The **ROW** function returns the row number of *reference*.
reference can be a cell or a range of cells. If the *reference* argument is omitted, the default argument is the reference of the cell in which the **ROW** function is placed.

Beispiel

```
ROW(B2) // 2

ROW(B1:B5) //1
```

SIN

SIN (value)

Parameter	Typ		Beschreibung
value	Zahl	➡	Angle whose sine is returned

Beschreibung

The **SIN** function returns the sine of the angle specified in *value*. It accepts and returns numeric data.
In *value*, pass any real number (an angle) for which to return the sine. It must be expressed in radians. If the angle is in degrees, multiply it by PI/180 to convert it to radians.

Beispiel

```
SIN(B4)

SIN(30*PI()/180) //0.5
```

SQRT

SQRT (value)

Parameter	Typ		Beschreibung
value	Zahl	➡	Number >= 0 to get the square root

Beschreibung

The **SQRT** function returns the positive square root of the specified *value*.
value must be a positive number (or 0), otherwise an error is returned.

Beispiel

```
SQRT(B4)

SQRT(256) //16
```

STDEV.P

STDEV.P (Wert {, Wert2 , ... , WertN})

Parameter	Typ		Beschreibung
Wert	Zahl, CellRef	➡	Werte zum Bewerten

Beschreibung

Die Funktion **STDEV.P** gibt die Standardabweichung einer Population zurück, basierend auf einem Teil dieser Population. Standardabweichung ist das Maß der Verteilung der Werte in Bezug auf den Durchschnitt (Durchschnittswert).

In *Wert, Wert2...* übergeben Sie numerische Argumente entsprechend dem Teil der Population. Each argument can be a cell, a cell range, or a number. This function can have up to 255 arguments.

The **STDEV.P** function works on the hypothesis that arguments represent only a sample of the population. The standard deviation is calculated using the "without bias" method, or "n-1". The **STDEV.P** function uses the following formula:

$$\sqrt{\frac{\sum (x-\bar{x})^2}{n}}$$

Beispiel

STDEV.P(A1,B2,C3,D4,E5,F6)

STDEV.P(A1:A9)

STDEV.P(95,89,73,87,85,76,100,96,96) gives the result 8.8079649700473

SUBSTITUTE

SUBSTITUTE (str , toReplace , replacement {, instance})

Parameter	Typ		Beschreibung
str	CellRef	⇒	String or reference to a cell containing the string in which you want to replace characters
toReplace	Text	⇒	String to replace
replacement	Text	⇒	New string to use instead of existing string
instance	Zahl	⇒	Which occurrence of the existing string to replace; otherwise every occurrence is replaced

Beschreibung

The **SUBSTITUTE** function replaces *toReplace* in the *str* text with *replacement* and returns the edited text.

By default, **SUBSTITUTE** replaces the first occurrence of *toReplace*. Pass *instance* to define which occurrence you want to replace.

If no occurrence of *toReplace* is found, **SUBSTITUTE** returns *str*.

Beispiel

```
SUBSTITUTE("Hello You","You","World") //Hello World
```

SUM

SUM (value {, value2 , ... , valueN})

Parameter	Typ		Beschreibung
value	Zahl, CellRef	➡	Number(s) to be added together

Beschreibung

The **SUM** function returns the sum of cells or range of cells.

In *value*, pass the values to evaluate. Each argument can be a number, or an array of numbers (a single array (cell range) or multiple arrays (cell ranges)).

Note: The + operator provides an auto-conversion for non-numeric values passed by constant and for non-numeric values passed by reference. The **SUM** function provides an auto-conversion for non-numeric values passed by constant but, ignores non-numeric values passed by reference.

Beispiel

```
SUM(A1,B7,C11)
SUM(A1:A9)
SUM(A2:A14,B2:B18,D12:D30)
SUM(95,89,73,87,85,76,100,96,96) //797
```

TAN

TAN (value)

Parameter	Typ		Beschreibung
value	Zahl	⇨	Angle whose tangent is returned

Beschreibung

Die Funktion **TAN** gibt den Tangens zum in *Wert* angegebenen Winkel zurück. Sie akzeptiert und gibt numerische Daten zurück.

In *Wert* übergeben Sie eine Zahl (einen Winkel) zum Zurückgeben des Tangens. Er muss in Radiant ausgedrückt werden. Ist er in Grad angegeben, multiplizieren Sie den Winkel mit PI/180, um ihn in Radiant umzuwandeln

Beispiel

```
TAN(B3)

TAN(45*PI()/180) //1
```

TEXT

TEXT (value , format)

Parameter	Typ		Beschreibung
value	Zahl, CellRef	➡	Numeric value to format to text
format	Text	➡	Format to apply to value

Beschreibung

The **TEXT** function returns a string composed of *value* number formatted according to *format*.
Pass a numeric data or a reference to a cell that contains numeric data in *value* and a text argument in *format*.

Beispiel

```
TEXT(A1,"$0.00") //$10.00 if A1 contains 10

TEXT(100,"0.00€") //100.00€
```

TODAY

TODAY ()
Dieser Befehl benötigt keine Parameter

Beschreibung

The **TODAY** function returns the date and time of the current date. It returns a DateTime object.
This function is updated when the spreadsheet or cell containing the function is recalculated.

Beispiel

If the current day is Monday 8 January 2018:

```
TODAY() //1/8/2018 0:00:00
```

TRUE

TRUE ()
Dieser Befehl benötigt keine Parameter

Beschreibung

The **TRUE** function returns the logical TRUE value (1).

Beispiel

```
TRUE() //TRUE
```

TYPE

TYPE (value)

Parameter	Typ	Beschreibung
value	➡	Value to evaluate

Beschreibung

The **TYPE** function returns the type of *value* as a number.
Returned types are listed below:

Type of value	Returned number
Number	1
DateTime object	1
TimeSpan object	1
Text	2
Boolean	4
Error	16
Array	64

Use the **TYPE** function when the execution of another function depends on the type of value contained in a specific cell. The **TYPE** function is particularly useful when calling functions that accept different types of data.

Beispiel

```
TYPE(G15)

TYPE(42) //1

TYPE("String") //2

TYPE(TRUE) //4
```

VAR.P

VAR.P (value {, value2 , ... , valueN})

Parameter	Typ		Beschreibung
value	Zahl, Array	➡	Values to get the variance

Beschreibung

The **VAR.P** function returns the variance based on the entire population, which uses numeric values.

In *value*, pass the values to evaluate. Each argument can be a number, or an array of numbers (a single array (cell range) or multiple arrays (cell ranges)).

Note: This function assumes that its arguments are the entire population. If your data represents only a sample of the population, then compute the variance using the VAR.S function.

Beispiel

```
VAR.P(B3,C4,B2,D10,E5)
VAR.P(A1:A9)
VAR.P(98,85,76,87,92,89,90) //39.265306122449
```