combit®

combit CRM®

Version 12

Programmierer-Referenz (SDK)

Inhalt

1 Einführung

Ihnen steht ein mächtiges Objektmodell zur Verfügung, das den Zugriff auf die Ansichten, die Telefonie, das Terminmanagement und vieles mehr erlaubt. Mit Hilfe von Scripten können Sie dieses Objektmodell ansprechen und somit viele Routineaufgaben komfortabel automatisieren.

Sogar eventuell nicht vorhandene spezielle Funktionen können Sie so leicht selbst integrieren. Darüber hinaus können Scripte andere Programme ansprechen und deren Funktionalitäten nutzen, sofern diese eine entsprechende Schnittstelle anbieten.

> Hinweis: Ein Script ist eine Abfolge von Befehlen, die bei der Ausführung sequenziell abgearbeitet werden. Die Befehle entstammen dabei dem "Wortschatz" einer bestimmten Scriptsprache. Dieser Befehlssatz bestimmt dabei, welche Möglichkeiten die Sprache bietet und wie ein Script aufgebaut sein muss.

Scripts sind in der Regel nicht allzu umfangreich und führen schon mit wenigen Befehlen zu beachtlichen Leistungen. Ein durchschnittliches Script umfasst vielleicht 20 bis 40 Zeilen Befehle. Nicht zuletzt aus diesen Gründen sind Scriptsprachen meist sehr leicht zu erlernen.

Obwohl oberflächlich sehr ähnlich, gibt es doch eine ganze Reihe von entscheidenden Unterschieden zwischen Scripts und ausführbaren Programmen.

So sind Scripts beispielsweise nicht selbst lauffähig, sondern benötigen immer eine Umgebung, in der sie ablaufen. Diese sogenannten Hosts sind für die Verwaltung der Scripts verantwortlich und erweitern die Möglichkeiten der jeweiligen Sprache meist in Form von zusätzlichen Objekten.

In unserem Fall ist die Anwendung der Host; eine Erweiterung findet durch dessen Objektmodell statt. Ein weiterer wichtiger Unterschied ist die fehlende Möglichkeit, eigenständig Dialoge zu implementieren. Hierzu sind externe Objekte notwendig.

1.1 Welche Scriptsprachen werden unterstützt?

Zu den unterstützten Scriptsprachen gehören C# sowie die beiden vom Windows Scripting Host ausgeführten Scriptsprachen VBScript und JScript. Wir empfehlen die Verwendung von C# und VBScript. Theoretisch werden auch alle anderen Scriptsprachen des Windows Scripting Host unterstützt. Die Auswahl der Sprache erfolgt dabei über die Dateiendung, z. B. "vbs" bei VBScript, "js" bei Jscript und "csscript" bei C#.

> Hinweis: VBScript und Jscript sind üblicherweise schon auf Ihrem System installiert. Falls nicht oder falls andere Sprachen verwendet werden sollen, müssen diese vom jeweiligen Hersteller bezogen werden und entsprechend seinen Angaben auf dem System installiert werden.

1.1.1 Hinweise zur Benutzung von C#-Scripten

Bei C#-Scripten steht Ihnen der volle Funktionsumfang des .NET Frameworks 4.8 zur Verfügung (abwärtskompatibel bis .NET Framework 4.0), weshalb für das Ausführen von C#-Scripten auch mindestens dieses oder ein neueres .NET Framework installiert sein muss. Die dafür benötigten Module "combit.CSharpScript28.Engine.x86.dll" und "combit.CSharpScript28.Interface.x86.dll" befinden sich im Hauptverzeichnis der combit CRM Installation. Eigene Scripte oder Module, die nach einem Update der combit CRM Hauptversion verwendet werden sollen, brauchen zwingend eine aktualisierte Referenz auf die zuvor genannten combit.CSharpScript*-Module.

> Hinweis: Unterstützt wird die C#-Sprachspezifikation bis einschließlich Version 7.3.

Kleinere C#-Scripte können direkt als einzelne Zeilen in eine Datei oder die Eingabemaske geschrieben werden. Wenn jedoch verschiedene Methoden verwendet werden sollen, so muss der Hauptteil des Scripts in die Main-Methode (public static void Main()) verlagert werden. Ein C#-Script ohne Main-Methoden, mit jedoch mindestens einer eigenen Methodendefinition kann nicht ausgeführt werden. Alle im Script definierten Methoden müssen mit dem Schlüsselwort "static" versehen sein. Normale Klassendefinitionen werden derzeit nicht unterstützt, um dennoch eigene Klassen zu verwenden können diese in externen Bibliotheken definiert und diese im Script inkludiert werden.

Kleinere, für ein einzelnes Script gedachte Klassendefinitionen können als "Nested Classes" innerhalb des Scripts platziert werden.

Die in diesem Dokument aufgelisteten Objekte der Objekt-Referenz dürfen ausschließlich im Ursprungsthread des Scripts erstellt werden. Wenn im Script ein neuer Thread erstellt und in diesem wiederum cRM-Objekte erstellt werden, so können diese beim Beenden des Scripts nicht korrekt aufgeräumt werden. Dadurch kann es bei der weiteren Benutzung von combit CRM zu Fehlern kommen.

Da C#-Scripte vor der Benutzung kompiliert werden müssen und dies unter Umständen länger dauern kann als das Ausführen des Scriptes selbst, werden die 30 neuesten kompilierten Scripte für ein erneutes Ausführen zwischengespeichert. Die entsprechenden Dateien befinden sich im Ordner "%temp%\combitCSharpScriptCache\". Dort gibt es zum einen die Datei "combitCSharpScriptCache.cache", welche Informationen über die gespeicherten Scripte enthält, und zum anderen für jedes Script einen Ordner mit einem Namen in der Form "combitCSharpScript_[GUID]" (z. B. "combitCSharpScript_e9527e037aa149f3ba79bd408a8232db"). In diesem Ordner befinden sich jeweils alle vom Script benutzten .dll-Dateien, Debug-Informationen und das Script selbst (ebenfalls in Form einer .dll-Datei).

Hinweis: Die Definition einer Ausnahme für den Cache-Ordner bei Echtzeit-Virenscannern kann eine Geschwindigkeitssteigerung bei der Ausführung der Scripte bewirken.

Es wird empfohlen, beim Durchlaufen von Collections in For-Each-Schleifen, alle verwendeten Objekte explizit mit dem Aufrufen der Methode Dispose() freizugeben. Dies verhindert einen erhöhten Speicherverbrauch und hilft dabei Instabilitäten zu vermeiden.

Sollte es Scripte geben, welche große Referenzen mit `<!--#include ref="[Pfad einer .dll]"-->` einbinden, so können diese eine spürbare Verzögerung beim Start aufweisen - bedingt durch das Laden der Referenzen in den Speicher. Normalerweise werden diese Referenzen bei jedem Scriptaufruf erneut geladen. Um Referenzen einmalig beim Programmstart zu laden und damit die Startzeit aller betroffenen Scripte zu verbessern, ist es möglich ein spezielles `"preload.csscript"`-Script im Programmverzeichnis (`%APPDIR%`) anzulegen. Dieses Script sollte lediglich die entsprechenden `"include ref"`–Anweisungen enthalten. Alle angegebenen Referenzen bleiben somit bis zum Beenden des combit CRM-Prozesses geladen.

Einen Einstieg in das C# Scripting erhalten Sie auch im Artikel C# Scripting mit Microsoft Visual Studio in unserer Knowledgebase.

1.2 Wo lassen sich Scripte integrieren?

Die Integration von Scripten ist über Schaltflächen innerhalb der Eingabemaske (Scriptdatei ausführen, Scriptzeilen ausführen, Funktionsdefinitionen) und dem Menüband (Funktionsdefinitionen), über den Befehl Extras > Script > Ausführen, über Folgeverknüpfungen oder über Ereignisse möglich.

Eine Beschreibung, wie die Scripts dort konkret hinterlegt werden, finden Sie innerhalb der zugehörigen Kapitel des Handbuches. Sie können dabei jeweils entscheiden, ob Sie das Script direkt eingeben oder in Form einer externen Textdatei hinterlegen möchten. Insbesondere bei größeren Scripts ist letzteres die bessere Wahl, da diese eine leichte Wiederverwertbarkeit an anderer Stelle erlaubt.

1.2.1 Hinweise zur Benutzung von Scripten in Folgeverknüpfungen

In Scripten, welche in Folgeverknüpfungen der Eingabemaske ausgeführt werden, darf nicht mit einem View-Objekt gearbeitet werden, welches über folgenden Weg erzeugt wurde: cRM.CurrentProject.ActiveViews.ActiveView. Ebenso betrifft dies alle Unterobjekte, die von einem derartigen View-Objekt abzweigen können.

Bitte verwenden Sie stattdessen ein View-Objekt, welches über den nachfolgenden Weg erzeugt wurde: WScript.Event.View.

Diese Vorgehensweise ist notwendig, da ein Ansichtswechsel implizit eine Folgeverknüpfung auslösen kann, deren View-Objekt im Fall von ActiveView auf die Ansicht zeigen kann, auf die gewechselt wird. Dadurch würden etwaige Script-Aufrufe nicht in der gewünschten Ansicht ausgeführt werden.

Des Weiteren gilt zu beachten, dass in Scripten von Folgeverknüpfungen keine Benutzerinteraktionen durchgeführt werden dürfen (Dialoge, Fortschrittsanzeigen, MessageBoxen, ...).

Hintergrund: Ein angezeigter Dialog verändert den Ablauf der Folgeverknüpfung. Der Fokuswechsel von einem Eingabefeld zu einem weiteren Eingabefeld kann hierdurch verhindert werden - insbesondere dann, wenn der darzustellende Dialog anschließend den Fokus besitzt.

1.3 Support zu Scripting-Funktionalitäten

Die Möglichkeiten der Scripting-Technologie sind sehr weitreichend und deren Beschreibung daher sicherlich Stoff für ein eigenes Buch. Wir bitten um Ihr Verständnis, dass wir keine Beschreibung der verwendeten Scriptsprachen liefern können. Entsprechende Werke finden Sie im Buchprogramm vieler größerer Fachverlage oder im Internet.

Eine ausführliche Referenz zu VBScript und Jscript bietet Microsoft in seinem Scripting Blog an.

In unserer Knowledgebase finden Sie unter dem Tag "Scripting" ebenfalls viele Artikel mit Hintergrund-Informationen, Tipps und Tricks. Weitere Artikel zur COM-Schnittstelle finden Sie meistens unter den Suchbegriffen *COM*, *OLE, C#* oder *Script*.

Selbstverständlich wollen wir Ihnen im Rahmen unseres Supports bei Ihren Fragen und Wünschen gerne mit Rat und Tat zur Seite stehen, um Ihnen einen optimalen Einsatz unseres Produktes zu ermöglichen. Wir bitten jedoch um Verständnis, das wir im Rahmen unseres Supports nur auf allgemeine Fragen zum Objektmodell eingehen können. Direkter Scripting-Support bzw. Fehleranalayse sind ausgeschlossen.

1.4 Allgemeine Script-Erweiterungen

Diese Script-Erweiterungen gelten nur für interne Scripte, d.h. für Scripte die über Schaltflächen (Script Direkt, Script Datei, Funktionsdefinitionen) innerhalb der Eingabemaske, über den Befehl Extras > Script > Ausführen oder über Ereignisse ausgeführt werden.

> Hinweis: Präprozessor-Direktiven wie "#include..." und "#pragma..." können nicht auskommentiert werden und werden daher in jedem Fall ausgeführt.

1.4.1 Scripte in Scripten einbinden

Bei häufig benötigten Funktionen bietet es sich an, diese zentral abzulegen, so dass sich eventuell notwendige Änderungen auf alle darauf basierenden Scripts auswirken. Hierzu wird die Einbindung von Scripts über eine spezielle Anweisung der folgenden Form unterstützt:

```
<!--#include file="c:\scripts\include.vbs"-->
```

Die Anweisung wird dabei durch den kompletten Inhalt der angegebenen Datei ersetzt. Um eventuellen Syntaxfehlern vorzubeugen, empfiehlt es sich daher, die Anweisung in eine eigene Zeile zu setzen.

Es besteht zudem die Möglichkeit ein Script nur einmalig in Scripte einzubinden. Alle nachfolgenden Einbindungen dieses Scripts, auch in anderen eingebundenen Scripten, werden ignoriert. Diese Funktion ist besonders bei verschachtelten Einbindungen von Scripten nützlich, um sicherzustellen, dass beispielsweise ausgelagerte Hilfsroutinen, die unter Umständen auch von anderen Scripten eingebunden wurden, keinen Scriptfehler verursachen. Die dafür zu nutzende Anweisung lautet wie folgt:

```
<!--#include-once file="c:\scripts\include.vbs"-->
```

> Hinweis: Alle auf diese Weise eingebundenen Scripts müssen die gleiche Scriptsprache verwenden wie das Hauptscript. Eine Mischung von mehreren Sprachen ist nicht möglich und führt zu Syntaxfehlern.

Sofern Sie Ihre Scripts unterhalb des Programmverzeichnisses abgelegt haben, können Sie statt einer festen Angabe des Verzeichnisses die %APPDIR% Variable verwenden:

```
<!--#include file="%APPDIR%\include.vbs"-->
<!--#include-once file="%APPDIR%\include.vbs"-->
```

> *Tipp:* ***Die Variable %APPDIR% steht auch an anderen Stellen zur Verfügung, z. B. in der Eingabemaske, bei den Funktionsdefinitionen und innerhalb des Menübands.***
>
> ***%APPDIR% kann als Platzhalter verwendet werden, der vom Programm durch den Pfad der Anwendung ersetzt wird, wie z. B. C:\Program Files (x86)\combit\combit CRM\.***

%PRJDIR% wird analog dazu durch den Pfad des Projekts ersetzt.

1.4.2 Synchrone und Asynchrone Scripte

Da die Einsatzgebiete von Scripten sehr unterschiedlich sind, stehen zwei verschiedene Ausführungsarten zur Auswahl.

1.4.2.1 Synchrone Scripte

Ein synchrones Script hält den Host solange auf, bis es vollständig abgearbeitet ist. Wenn Sie also bspw. Innerhalb eines Scriptes zeitlich längere Operationen durchführen, können Sie während der kompletten Ausführungszeit nicht mit dem Programm arbeiten.

Synchrone Scripte bieten sich daher für Abläufe an, die im direkten Kontext zur aktuellen Arbeit stehen, wie z. B. die Automatisierung von mehreren kleineren Routineaufrufen, wie einem Filter und dem anschließenden Export. Jedes nicht anders gekennzeichnete Script ist automatisch ein synchrones Script.

1.4.2.2 Asynchrone Scripte

Diese Art von Scripts wird parallel zum Host ausgeführt, so dass Sie währenddessen mit diesem arbeiten können. Es wird hierfür ein eigener sogenannter "Thread" geöffnet. Auf diese Art ist der Host während der Ausführung nicht blockiert.

Wir empfehlen die Verwendung von asynchronen Scripten bei längerfristigeren Scripten, die nicht im direkten Kontext zur aktuellen Arbeit stehen, wie dem Ansteuern anderer Applikationen.

Damit ein Script asynchron ausgeführt wird, muss es in der ersten Zeile die folgende Anweisung enthalten:

```
<!--#pragma asynchronous-->
```

(Längere) asynchrone Scripte sollten eine Abbruchsbedingung vorsehen und hierzu regelmäßig den Wert der Terminate Eigenschaft abfragen.

Bitte beachten Sie dabei, dass die noch laufenden, asynchronen Scripte insgesamt 4 Sekunden Zeit haben, sich zu beenden, bevor die Beendigung erzwungen wird. Sleeps und andere Aktionen im Script sollten also keinesfalls länger als 3 Sekunden dauern, bevor wieder die .Terminate Eigenschaft geprüft und das Script dann ggf. sich so schnell wie möglich kontrolliert beenden muss.

Wichtig: Bei C#-Scripten kann die Beendigung nicht erzwungen werden, es ist daher umso wichtiger die .Terminate Eigenschaft rechtzeitig zu prüfen und das Script kontrolliert zu beenden. Sollte sich das Script nicht innerhalb von 4 Sekunden beendet haben, kann es zu Programmfehlern kommen!

1.4.3 Scripte und der Edit-Modus

Soll ein Script bei Klicken einer Schaltfläche in der Eingabemaske ohne Speichern-Frage gestartet werden, muss folgende Anweisung enthalten sein:

```
<!--#pragma keepeditmode-->
```

Auf diese Weise wird die Speichern-Frage unterdrückt und Sie bleiben im Edit-Modus.

1.4.4 Verschlüsselung von Scripten

Wählen Sie Extras > Script > Verschlüsseln um den Scriptcode zu verschlüsseln und den Inhalt zu schützen. Wenn Sie beim Verschlüsseln als Zieldatei die Quelldatei angegeben, wird diese überschrieben. Ein Entschlüsseln der verschlüsselten Scriptdatei ist nicht mehr möglich.

Enthält ein Script die Anweisung

```
<!--#pragma encrypted-->
```

so wird es ab dieser Stelle bis zum Ende komplett verschlüsselt. Damit kann ein Teil des Scriptes bearbeitbar bleiben, damit bestimmte Parameter oder Variablen veränderbar sind, der Rest des Scriptes bleibt jedoch geschützt. Ein verschlüsseltes Script wird automatisch komprimiert.

1.4.5 Automatische Speicherung von Änderungen in Eingabemaske

Der Befehl

```
<!--#pragma autosave-->
```

kann verwendet werden, wenn eine automatische Speicherung etwaiger Änderungen in der Eingabemaske ohne weitere Rückfrage erfolgen soll. Ist dies erfolgreich möglich oder gab es gar keine Änderung, so wird das Script ausgeführt. Schlägt das Speichern fehl (z. B. durch eine Verletzung einer erzwungenen Eingaberegel), so wird das Script nicht ausgeführt.

Bitte beachten Sie, dass `<!--#pragma autosave-->` unwirksam wird, wenn gleichzeitig `<!--#pragma keepeditmode-->` angegeben wurde.

1.4.6 Auswahl der Scriptsprache

Über den Befehl

```
<!--#language="[Scriptsprache]"-->
```

kann die Scriptsprache explizit gesetzt werden. Das ist immer dann nötig, wenn ein Script nicht als Datei ausgeführt und die Sprache anhand der Dateiendung ermittelt werden kann (z. B. beim Ausführen als Scriptzeilen in der Eingabemaske)

Mögliche Werte sind u.a. "VBScript", "Jscript", "C#", "CSharp" oder "C#Script".

Beispiel:

```
<!--#language="C#"-->
```

1.5 C#-spezifische Script-Erweiterungen

Diese Optionen stehen nur bei C#-Scripten zur Verfügung.

1.5.1 Protokollierung

Um die Protokollierung eines Scriptes zu aktivieren, muss folgende Anweisung enthalten sein:

```
<!--#pragma forcelogging-->
```

Die Protokollausgaben erfolgen in die Datei "%temp%\combitCSharpScript.log".

1.5.2 Debugmodus

Enthält ein Script die Anweisung

```
<!--#pragma debugmode-->
```

dann wird zum einen zu Beginn des Scripts das Starten eines Debuggers ausgelöst (sofern auf Ihrem System ein solcher installiert ist), mit dem Sie das Script Schritt für Schritt auf Fehler überprüfen können und zum anderen enthalten durch Ausnahmen ausgelöste Fehlertexte mehr Informationen und Zeilennummern.

1.5.3 Hinzufügen von Referenzen

Über den Befehl

```
<!--#include ref="[Pfad einer .dll]"-->
```

können einem Script externe Referenzen / Komponenten, wie z. B. die Windows Forms Bibliothek von Microsoft (System.Windows.Forms.dll), hinzugefügt werden.

Geben Sie keinen Pfad, sondern nur einen Dateinamen an, so wird versucht die Referenz aus dem "Global Assembly Cache" zu laden. Bei Angabe eines kompletten Pfads wird eine Kopie der Datei in einem temporären Ordner angelegt und von dort geladen.

> Hinweis: Dieser Befehl muss im Script nach den für alle Scriptsprachen geltenden Präprozessor-Direktiven erfolgen. Die folgenden Referenzen müssen nicht manuell hinzugefügt werden: System.dll, System.Core.dll, Microsoft.CSharp.dll.

1.5.4 Hinzufügen von Namespaces

Über den Befehl

```
<!--#include using="[Namespace]"-->
```

können using-Anweisungen zum Script hinzugefügt werden. Dies hat den Vorteil, dass Namespace-Namen nicht immer explizit angegeben werden müssen.

Anstatt des Aufrufs "System.Windows.Forms.MessageBox.Show(…);" für jede einzelne MessageBox würde der Aufruf nach dem Hinzufügen von "System.Windows.Forms" als using-Anweisung nur noch "MessageBox.Show(…);" lauten.

> Hinweis: Dieser Befehl muss im Script nach den für alle Scriptsprachen geltenden Präprozessor-Direktiven erfolgen. Das Hinzufügen des Namespace "System" muss nicht manuell erfolgen.

1.6 Benutzerrechte

Es gibt Benutzerrechte, die festlegen, ob ein Benutzer Scripts grundsätzlich ausführen darf. Mehr dazu finden Sie im Kapitel "Benutzer- und Rechteverwaltung" des Handbuchs.

1.7 Zugriff von außen

Mit Ausnahme des WScript Objektes steht Ihnen auch außerhalb von Scripten der Leistungsumfang des Objektmodells zur Verfügung. Sie können so beispielsweise eine Anwendung schreiben, die Ihre Daten aus der Anwendung bezieht und diese in einer beliebigen Weise bearbeitet, um sie anschließend zurück zu schreiben.

Sie benötigen zum Zugriff lediglich eine Instanz des Application Objektes. Die sogenannte ProgID des Application Objektes lautet "cRM.Application".

Die Instanzierung erfolgt in der Regel über einen CreateObject Befehl (Late Binding). Die zur Verfügung stehenden Möglichkeiten sowie die genaue Syntax hängen von der verwendeten Programmiersprache ab. Entsprechende Hinweise finden Sie in der jeweiligen Dokumentation.

> Hinweis: C#-Scripte können momentan ausschließlich intern benutzt werden.

1.8 Komprimieren von Scripten

Wählen Sie Extras > Script > Komprimieren um den Scriptcode zu komprimieren und die Dateigröße einer Script-Datei zu verringern. Hierbei werden nicht zwingend benötigte Leerzeichen, Tabulatoren, Leerzeilen sowie Kommentare entfernt.

1.9 Kodierung von Scripten

Die Kodierung von Scriptdateien kann ANSI, Unicode (UCS-2) mit BOM (BYTE ORDER MARK), oder UTF-8 mit BOM sein. Eine UTF8-Kodierung ohne BOM führt zu Problemen, wenn im Script-Code länderspezifische Zeichen (Umlaute etc.) verwendet werden. Eine Kodierung in ANSI führt zu Problemen, wenn Sie im Script-Code Zeichen aus fremden Zeichensätzen (z. B. kyrillische, slavische u.a. Buchstaben) verwenden möchten. Wir empfehlen die Kodierung entweder in UTF-8 mit BOM oder Unicode (UCS-2) mit BOM.

1.10 Ereignisse für Scripte

Über die Ereignisverwaltung kann auf verschiedene Ereignisse mit automatischen Aktionen reagiert werden. Hierzu können für jede der angebotenen Ereignisse Scripte hinterlegt werden. Weitere Informationen finden Sie im Abschnitt "Ereignis Verwaltung" im Kapitel "Workflows und Ereignisse" des Handbuchs.

Beispiel VBScript: Ereignis 'Datensatz wird gespeichert' der Ansicht Firma, um dem Anwender eine Hinweismeldung anzuzeigen, dass er für den aktuellen Datensatz nur Leserechte besitzt.

```
Dim oEvent
Set oEvent = WScript.Event
```

```
Dim DoCancel
DoCancel = false
Dim Value
Value = oEvent.Record1.GetContentsByName("ABC")
If(Value = "A") Then
    DoCancel = True
    MsgBox "Sie haben nur Leserechte auf diesen Datensatz!", vbInformation,
cRM.AppTitle
End If
oEvent.Cancel = DoCancel
```

Beispiel C#-Script:

```
object event = Event;
bool doCancel = false;
string value = event.Record1.GetContentsByName("ABC");
if (value == "A")
{
    doCancel = true;
    MessageBox.Show("Sie haben nur Leserechte auf diesen Datensatz!",
cRM.AppTitle, MessageBoxButtons.OK);
}
event.Cancel = doCancel;
```

1.11 Script-Konstanten

Nachfolgende Konstanten können für verschiedene Methoden verwendet werden, welche sich am Funktionsumfang einer MsgBox bzw. einer MessageBox orientieren. Die einzelnen Konstanten bzw. deren Enumerationswerte können in den dafür vorgesehenen Parametern auch mit "or" (C#) oder "+" (VBScript) verknüpft werden, um eine Kombination aus unterschiedlichen Werten zu ermöglichen (z. B. Anzeige von "Ja"-, "Nein"- und "Abbrechen"-Schaltflächen innerhalb eines Dialogs).

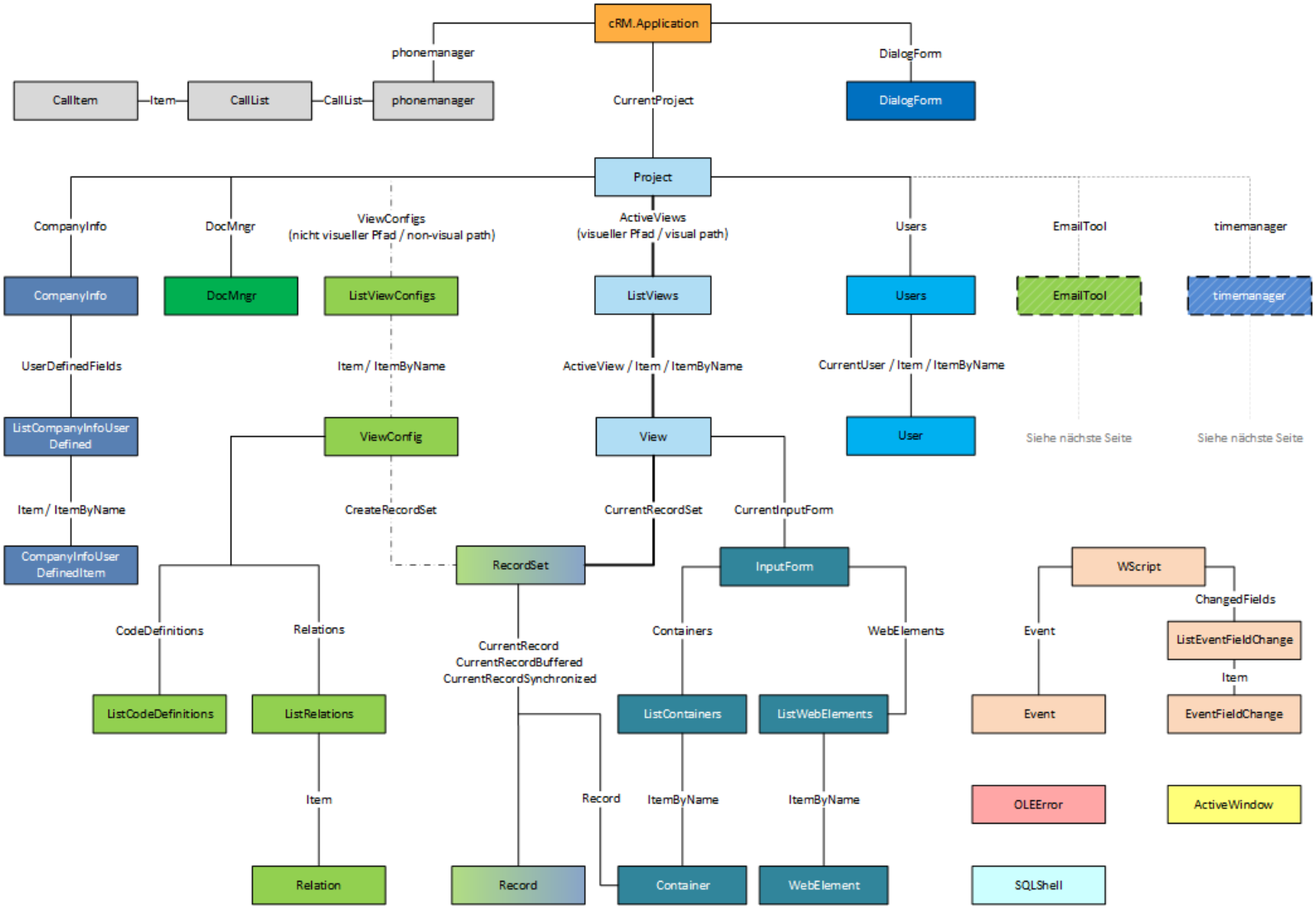
Methoden, die diese Konstanten unterstützen sind z. B. DialogChoiceMessageBox, DialogMessageBox und DialogMessageBoxAuto aus dem cRM/Application-Objekt. Selbstverständlich können die Konstanten auch in den jeweiligen Dialogen der Programmiersprachen C# (MessageBox) und VBScript (MsgBox) verwendet.

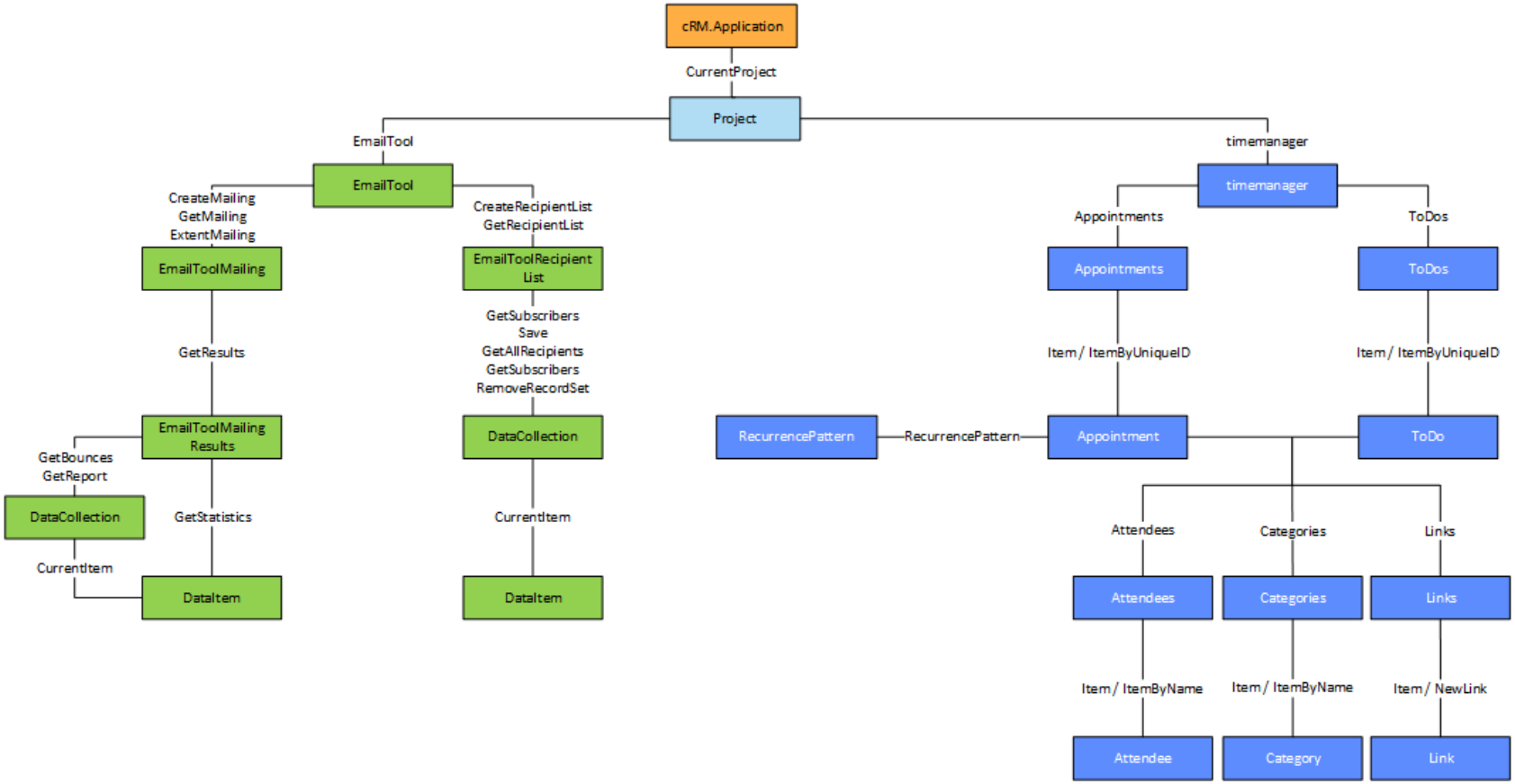
Name der Konstante	Enumerations-wert	Prog.-Spr. / Verwendung	Beschreibung
OK	0	C# / MessageBoxButtons	Der Dialog enthält die Schaltfläche "OK".
OKCancel	1	C# / MessageBoxButtons	Der Dialog enthält die Schaltflächen "OK" und "Abbrechen".
AbortRetryIgnore	2	C# / MessageBoxButtons	Der Dialog enthält die Schaltflächen "Abbrechen", "Wiederholen" und "Ignorieren".
YesNoCancel	3	C# / MessageBoxButtons	Der Dialog enthält die Schaltflächen "Ja", "Nein" und "Abbrechen".
YesNo	4	C# / MessageBoxButtons	Der Dialog enthält die Schaltflächen "Ja" und "Nein".
RetryCancel	5	C# / MessageBoxButtons	Der Dialog enthält die Schaltflächen "Wiederholen" und "Abbrechen".
None	0	C# / MessageBoxIcons	Der Dialog beinhaltet keine Symbole.
Error / Hand / Stop	16	C# / MessageBoxIcons	Der Dialog beinhaltet ein Kreissymbol mit rotem Hintergrund und weißem X in der Mitte.
Question	32	C# / MessageBoxIcons	Der Dialog beinhaltet ein Kreissymbol mit blauem Hintergrund und weißem Fragezeichen in der Mitte.
Exclamation / Warning	48	C# / MessageBoxIcons	Der Dialog beinhaltet ein Dreiecksymbol mit gelbem Hintergrund und schwarzem Ausrufezeichen in der Mitte.

Information / Asterisk	64	C# / MessageBoxIcons	Der Dialog beinhaltet ein Kreissymbol mit blauem Hintergrund und weißem i in der Mitte.
Button1	0	C# / MessageBoxDefaultButtons	Die erste Schaltfläche des Dialogs wird als Standardschaltfläche definiert.
Button2	256	C# / MessageBoxDefaultButtons	Die zweite Schaltfläche des Dialogs wird als Standardschaltfläche definiert.
Button3	512	C# / MessageBoxDefaultButtons	Die dritte Schaltfläche des Dialogs wird als Standardschaltfläche definiert.
None	0	C# / Rückgabewerte MessageBox (DialogResult)	Es wurde noch kein Rückgabewert geliefert, der Dialog wird derzeit noch angezeigt.
OK	1	C# / Rückgabewerte MessageBox (DialogResult)	Der Dialog wurde mit Klick auf die Schaltfläche "OK" beendet.
Cancel	2	C# / Rückgabewerte MessageBox (DialogResult)	Der Dialog wurde mit Klick auf die Schaltfläche "Abbrechen" beendet.
Abort	3	C# / Rückgabewerte MessageBox (DialogResult)	Der Dialog wurde mit Klick auf die Schaltfläche "Abbrechen" beendet.
Retry	4	C# / Rückgabewerte MessageBox (DialogResult)	Der Dialog wurde mit Klick auf die Schaltfläche "Wiederholen" beendet.
Ignore	5	C# / Rückgabewerte MessageBox (DialogResult)	Der Dialog wurde mit Klick auf die Schaltfläche "Ignorieren" beendet.
Yes	6	C# / Rückgabewerte MessageBox (DialogResult)	Der Dialog wurde mit Klick auf die Schaltfläche "Ja" beendet.
No	7	C# / Rückgabewerte MessageBox (DialogResult)	Der Dialog wurde mit Klick auf die Schaltfläche "Nein" beendet.
vbOkOnly	0	VBScript / MsgBox Buttons	Der Dialog enthält die Schaltfläche "OK".
vbOkCancel	1	VBScript / MsgBox Buttons	Der Dialog enthält die Schaltflächen "OK" und "Abbrechen".
vbAbortRetryIgnore	2	VBScript / MsgBox Buttons	Der Dialog enthält die Schaltflächen "Abbrechen", "Wiederholen" und "Ignorieren".
vbYesNoCancel	3	VBScript / MsgBox Buttons	Der Dialog enthält die Schaltflächen "Ja", "Nein" und "Abbrechen".
vbYesNo	4	VBScript / MsgBox Buttons	Der Dialog enthält die Schaltflächen "Ja" und "Nein".
vbRetryCancel	5	VBScript / MsgBox Buttons	Der Dialog enthält die Schaltflächen "Wiederholen" und "Abbrechen".
vbCritical	16	VBScript / MsgBox Icons	Der Dialog beinhaltet ein Kreissymbol mit rotem Hintergrund und weißem X in der Mitte.
vbQuestion	32	VBScript / MsgBox Icons	Der Dialog beinhaltet ein Kreissymbol mit blauem Hintergrund und weißem Fragezeichen in der Mitte.
vbExclamation	48	VBScript / MsgBox Icons	Der Dialog beinhaltet ein Dreiecksymbol mit gelbem Hintergrund und schwarzem Ausrufezeichen in der Mitte.
vbInformation	64	VBScript / MsgBox Icons	Der Dialog beinhaltet ein Kreissymbol mit blauem Hintergrund und weißem i in der Mitte.
vbDefaultButton1	0	VBScript / MsgBox Default Button	Die erste Schaltfläche des Dialogs wird als Standardschaltfläche definiert.

vbDefaultButton2	256	VBScript / MsgBox Default Button	Die zweite Schaltfläche des Dialogs wird als Standardschaltfläche definiert.
vbDefaultButton3	512	VBScript / MsgBox Default Button	Die dritte Schaltfläche des Dialogs wird als Standardschaltfläche definiert.
vbDefaultButton4	768	VBScript / MsgBox Default Button	Die vierte Schaltfläche des Dialogs wird als Standardschaltfläche definiert.
vbOK	1	VBScript / Rückgabewerte MsgBox	Der Dialog wurde mit Klick auf die Schaltfläche "OK" beendet.
vbCancel	2	VBScript / Rückgabewerte MsgBox	Der Dialog wurde mit Klick auf die Schaltfläche "Abbrechen" beendet.
vbAbort	3	VBScript / Rückgabewerte MsgBox	Der Dialog wurde mit Klick auf die Schaltfläche "Abbrechen" beendet.
vbRetry	4	VBScript / Rückgabewerte MsgBox	Der Dialog wurde mit Klick auf die Schaltfläche "Wiederholen" beendet.
vbIgnore	5	VBScript / Rückgabewerte MsgBox	Der Dialog wurde mit Klick auf die Schaltfläche "Ignorieren" beendet.
vbYes	6	VBScript / Rückgabewerte MsgBox	Der Dialog wurde mit Klick auf die Schaltfläche "Ja" beendet.
vbNo	7	VBScript / Rückgabewerte MsgBox	Der Dialog wurde mit Klick auf die Schaltfläche "Nein" beendet.

2 Objekthierarchie




cRM.Application
CurrentProject
Project
EmailTool
EmailTool
CreateMailing
GetMailing
ExtentMailing
EmailToolMailing
GetResults
EmailToolMailing Results
GetBounces
GetReport
DataCollection
CurrentItem
DataItem
GetStatistics
CreateRecipientList
GetRecipientList
EmailToolRecipient List
GetSubscribers
Save
GetAllRecipients
GetSubscribers
RemoveRecordSet
DataCollection
CurrentItem
DataItem
timemanager
timemanager
Appointments
Appointments
Item / ItemByUniqueID
ToDos
ToDos
Item / ItemByUniqueID
RecurrencePattern
RecurrencePattern
Appointment
ToDo
Attendees
Categories
Links
Attendees
Categories
Links
Item / ItemByName
Item / ItemByName
Item / NewLink
Attendee
Category
Link

2.1 Hinweise zur Objekthierarchie

Der Aufbau der Objekthierarchie besteht im Wesentlichen aus einer Sammlung von Objekten, die miteinander verbunden sind und auf unterschiedlichen Ebenen organisiert sind. Die Objekte in dieser Hierarchie besitzen unterschiedliche Eigenschaften und Methoden, können untereinander interagieren und somit beispielsweise Automatisierungen ermöglichen.

2.1.1 Arbeit mit Objekten, Methoden und Eigenschaften

Was sind Objekte innerhalb der combit CRM Objekthierarchie?

Rein visuell sind alle Elemente der Objekthierarchie als Objekt zu bezeichnen, die in rechteckigen Formen und mit einer entsprechenden Füllfarbe innerhalb der SDK-Dokumentation hinterlegt wurden. Darunter fallen z. B. das cRM-Objekt, das Project-Objekt und das Record-Objekt.

Objekte besitzen Eigenschaften, die gewisse Attribute dieser Objekte beschreiben, und Methoden oder Funktionen, die sie ausführen können. Eine Eigenschaft könnte z. B. die Versionsnummer des geöffneten combit CRM (*cRM.Version*) sein oder der Name des geladenen Projekts (*Project.Name*). Methoden bzw. Funktionen sind beispielsweise für das Versenden von E-Mails an einen Datensatz verantwortlich (*Record.SendSingleMailDialog()*) oder ermöglichen die Erstellung einer Objektkopie (Instanziierung) von weiteren Objekten (*Project.ListViewConfigs()*).

Die Instanziierung von Objekten beschreibt prinzipiell die Erstellung eines Objektes eines bestimmten Objekttyps. Nach Auslesen oder Zuweisen von Eigenschaften kann ein eigens erstelltes Objekt auch entsprechende Aktionen in Form von Methoden oder Funktionen ausführen.

Im Objektmodell des combit CRM ist es mithilfe von Methoden oder Eigenschaften der unterschiedlichen Objekte auch möglich auf andere Objekte zuzugreifen oder diese zu initialisieren. Die wichtigsten Methoden werden auf den Verbindungslinien zwischen den einzelnen Objekten genannt. Es wird hierbei immer von oben nach unten gelesen.

2.1.1.1 Das cRM Root-Objekt

Das Root-Objekt der combit CRM-Objekthierarchie ist das oberste Element von dem alle anderen Objekte abgeleitet werden. Es stellt den Ursprungspunkt für die gesamte Hierarchie dar und dient als zentrales Element, an dem alle anderen Objekte anknüpfen.

Bei der Nutzung von Scripts, die innerhalb des combit CRM ausgeführt werden, wird das cRM-Root-Objekt automatisch bereitgestellt. Wird ein Script dazu verwendet, um eine eigene cRM -Instanz zu erstellen bzw. um den combit CRM zu starten, muss das cRM-Objekt, welches für den weiteren Scriptablauf verwendet wird, separat initialisiert werden.

Nachfolgend finden Sie unterschiedliche Beispiele zur Nutzung bzw. Initialisierung des cRM-Objekts, der Ausgabe unterschiedlicher Objekteigenschaften sowie die Verwendung der Methoden *cRM.DialogMessageBox* und *Project.Login*:

```
' Zuweisen des Root-Objekts in ein eigenes Objekt
Dim ocRM : Set ocRM = cRM

' Ausgabe des Applikationspfads des combit CRM
Call ocRM.DialogMessageBox("Der combit CRM wurde in folgendes Verzeichnis
installiert: " & ocRM.AppDir, "Installationspfad", vbOkOnly)


' Verwendung des Root-Objekts ohne Zuweisung eines eigenen Objekts für die Ausgabe
der Titelzeile des geöffneten combit CRM
Call cRM.DialogMessageBox("Die Titelzeile enthält folgenden Inhalt: " &
cRM.AppTitle, "Titelzeile", vbOkOnly)


' Starten einer neuen combit CRM-Instanz per Script, anschließendes Laden eines
Projekts und Login
Dim ocRM : Set ocRM = CreateObject("cRM.Application")
Dim oProject : Set oProject = ocRM.Login(sProject, sUsername, sPassword)

' Funktion zum Prüfen, ob bereits eine combit CRM-Instanz vorhanden ist, falls
nicht, neue combit CRM-Instanz erzeugen
Function CheckcRM()
```

```
  On Error Resume Next
      If (cRM Is Nothing) Then
          Set ocRM = CreateObject("cRM.Application")
      Else
          Set ocRM = cRM
      End If

  On Error GoTo 0
End Function
```

2.1.1.2 Weitere Objekte

Die drei vom cRM-Objekt abzweigenden Objekte können wie folgt zugewiesen werden:

```
' Zuweisen des Project-Objekt über die Methode CurrentProject() des cRM-Objekts
Dim oProject : Set oProject = cRM.CurrentProject()

' Zuweisen eines DialogForm-Objekts über die Eigenschaft DialogForm des cRM-
Objekts
Dim oDialogForm : Set oDialogForm = cRM.DialogForm

' Zuweisen eines phonemanager-Objekts über die Eigenschaft phonemanager des cRM-
Objekts
Dim oPhonemanager : Set oPhonemanager = cRM.phonemanager
```

Ausgehend von dieser Logik können wir den Weg zum häufig verwendeten Record-Objekt erstellen. Dabei kann auf das separate Zuweisen von Objektvariablen auch verzichtet werden.

```
' Zuweisen eines Record-Objekts über den visuellen Pfad (aktuell angezeigter
Datensatz)
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView().CurrentRecordSet().CurrentRecord()
```

Nutzen Sie gerne die Möglichkeit den Pfad innerhalb der Grafik der Objekthierarchie nachzuvollziehen.

Wenn Sie im weiteren Verlauf des Scripts auf Eigenschaften und Methoden von Objekten zugreifen möchten, dann wird empfohlen die Objekte in Objektvariablen zu übernehmen. Der Weg zu einem Record-Objekt könnte dann wie folgt aussehen:

```
Dim ocRM : Set ocRM = cRM
Dim oProject : Set oProject = ocRM.CurrentProject
Dim oListViews : Set oListViews = oProject.ActiveViews
Dim oActiveView : Set oActiveView = oListViews.ActiveView()
Dim oRecordSet : Set oRecordSet = oActiveView.CurrentRecordSet()
Dim oRecord : Set oRecord = oRecordSet.CurrentRecord()
```

2.2 Visueller vs. nicht visueller Pfad

Innerhalb der combit CRM-Objekthierarchie erkennt man einen Hinweis bei den Methoden *Project.ActiveViews()* sowie *Project.ViewConfigs()*: visueller Pfad bzw. nicht visueller Pfad – Was verwendet man wann?

2.2.1 Objekte des visuellen Pfads

ListViews – entspricht allen geöffneten Ansichten, die ein Anwender im combit CRM als Tab angezeigt bekommt.

View – entspricht einer geöffneten Ansicht, die entweder im Vordergrund angezeigt wird (*ListViews.ActiveView()*) oder einer der Ansichten, die vom Anwender zwar geöffnet wurden, jedoch aktuell nicht aktiv sind (*ListViews.Item()/ItemByName()*).

InputForm – entspricht der Eingabemaske der aktiven Ansicht und ermöglicht den Zugriff auf Datensatzinhalte, auch wenn der Datensatz ggf. neu angelegt und noch nicht gespeichert wurde (Bearbeitungsmodus).

RecordSet (instanziiert über die Methode: *View.CurrentRecordSet()*) – entspricht allen Datensätzen der aktiven Ansicht. Sollte ein Anwender einen Filter ausgeführt haben werden auch nicht alle Datensätze in das RecordSet übernommen, sondern nur die derzeit im Filter befindlichen Datensätze.

Record – entspricht dem aktuell aktiv angezeigten Datensatz bzw. dem markierten Datensatz innerhalb einer Listenansicht.

2.2.2 Objekte des nicht-visuellen Pfads

ListViewConfigs – entspricht allen konfigurierten Ansichten des Projekts.

ViewConfig – entspricht einer Ansichtskonfiguration einer anzugebenden Ansicht im Projekt. Mittels Eigenschaften und Methoden erhält man so z. B. Zugriff auf den physikalischen Datenbanktabellennamen, die Relationen der Ansicht oder den Feldalias aller konfigurierten Felder.

RecordSet (instanziiert über die Methode: *ViewConfig.CreateRecordSet()*) – erzeugt wird eine neue Sammlung von Datensätzen, je nach angegebener Ansicht. Es werden alle Datensätze aufgenommen. Etwaige vom Benutzer aktivierte Filter werden nicht verwendet, es gibt keinen Bezug zu einer vom Nutzer geöffneten Ansicht und den dort dargestellten Datensätzen. Mit Hilfe von Methoden kann innerhalb des RecordSet zu unterschiedlichen Datensätzen gesprungen werden (*RecordSet.MoveFirst()*, *RecordSet.MoveNext()*, *RecordSet.MovePrevious()*, *RecordSet.MoveLast()*).

Record – entspricht einem Datensatz, nachdem innerhalb des RecordSet eine der *RecordSet.Move…()*-Methoden erfolgreich war.

2.2.3 Unterschiede zwischen den beiden Pfaden

Wie man an der Auflistung der Objekte erkennen kann, gibt es wesentliche Unterschiede zwischen den beiden Pfaden, weswegen die richtige Verwendung essenziell ist – es sollte z. B. auf jeden Fall verhindert werden, dass ein Nutzer das Durchlaufen eines RecordSet über die *RecordSet.Move…()*-Methoden visuell wahrnehmen kann. Nicht nur ist hierbei die Performance durch das notwendige Anzeigen der Datensätze nicht ideal, es kann auch zu Flackern der Oberfläche führen, wenn beispielsweise in einer Schleife ein sichtbares RecordSet durchlaufen wird. Wenn z. B. eine Summe eines Feldes für mehrere Datensätze, in denen jeweils nur ein Einzelwert steht, ermittelt werden soll, so würde man in jedem Fall den nicht visuellen Pfad nutzen.

3 Objekt-Referenz

3.1 ActiveWindow Objekt

Liefert Informationen zum aktuellen, also aktiven, Windows-Fenster.

Bei der Verwendung im C# Script verwenden Sie als Typ für die Eigenschaften ‚dynamic'.

3.1.1 Eigenschaften

Class, read-only

Beschreibung:

Gibt den Klassennamen des aktiven Fensters zurück.

Typ:

String (VBScript) / dynamic (C#)

Beispiel VBScript:

```
Call cRM.DialogMessageBox("ActiveWindows.Class: " & ActiveWindow.Class,
"ActiveWindow.Class", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("ActiveWindow.Class: " + ActiveWindow.Class,
"ActiveWindow.Class", 0);
```

Handle, read-only

Beschreibung:

Gibt das Handle (hWnd) des aktiven Fensters zurück.

Typ:

Long (VBScript) / dynamic (C#)

Beispiel VBScript:

```
Call cRM.DialogMessageBox("ActiveWindows.Handle: " & ActiveWindow.Handle,
"ActiveWindow.Handle", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("ActiveWindow.Handle: " + ActiveWindow.Handle,
"ActiveWindow.Handle", 0);
```

Title, read-only

Beschreibung:

Gibt den Titel/Name des aktiven Fensters zurück.

Typ:

String (VBScript) / dynamic (C#)

Beispiel VBScript:

```
Call cRM.DialogMessageBox("ActiveWindows.Title: " & ActiveWindow.Title,
"ActiveWindow.Title", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("ActiveWindow.Title: " + ActiveWindow.Title,
"ActiveWindow.Title", 0);
```

3.2 AddressInfo Objekt

3.2.1 Eigenschaften

FldAdministrativeDistrict, read-only

Beschreibung:

Liefert den Feldnamen des Feldes zurück, welches dem Regierungsbezirk zugeordnet wurde.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("AddressInfo.FldAdministrativeDistrict: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldAdministrativeDistrict,
"AddressInfo.FldAdministrativeDistrict", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("AddressInfo.FldAdministrativeDistrict: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldAdministrativeDistrict,
"AddressInfo.FldAdministrativeDistrict", 0);
```

FldCity, read-only

Beschreibung:

Liefert den Feldnamen des Feldes zurück, welches der Stadt zugeordnet wurde.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("AddressInfo.FldCity: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldCity, "AddressInfo.FldCity", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("AddressInfo.FldCity: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldCity, "AddressInfo.FldCity", 0);
```

FldCountry, read-only

Beschreibung:

Liefert den Feldnamen des Feldes zurück, welches dem Land zugeordnet wurde.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("AddressInfo.FldCountry: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldCountry, "AddressInfo.FldCountry", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("AddressInfo.FldCountry: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldCountry, "AddressInfo.FldCountry", 0);
```

FldCounty, read-only

Beschreibung:

Liefert den Feldnamen des Feldes zurück, welches dem Landkreis zugeordnet wurde.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("AddressInfo.FldCounty: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldCounty, "AddressInfo.FldCounty", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("AddressInfo.FldCounty: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldCounty, "AddressInfo.FldCounty", 0);
```

FldFederalState, read-only

Beschreibung:

Liefert den Feldnamen des Feldes zurück, welches dem Bundesland zugeordnet wurde.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("AddressInfo.FldFederalState: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldFederalState, "AddressInfo.FldFederalState", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("AddressInfo.FldFederalState: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldFederalState, "AddressInfo.FldFederalState", 0);
```

FldHousenumber, read-only

Beschreibung:

Liefert den Feldnamen des Feldes zurück, welches der Hausnummer zugeordnet wurde.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("AddressInfo.FldHousenumber: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldHousenumber, "AddressInfo.FldHousenumber", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("AddressInfo.FldHousenumber: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldHousenumber, "AddressInfo.FldHousenumber", 0);
```

FldStreet, read-only

Beschreibung:

Liefert den Feldnamen des Feldes zurück, welches der Straße zugeordnet wurde.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("AddressInfo.FldStreet: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldStreet, "AddressInfo.FldStreet", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("AddressInfo.FldStreet: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldStreet, "AddressInfo.FldStreet", 0);
```

FldZip, read-only

Beschreibung:

Liefert den Feldnamen des Feldes zurück, welches der Postleitzahl zugeordnet wurde.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("AddressInfo.FldZip: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldZip, "AddressInfo.FldZip", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("AddressInfo.FldZip: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").FldZip, "AddressInfo.FldZip", 0);
```

ID, read-only

Beschreibung:

Liefert die ID der Adresse zurück.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("AddressInfo.ID: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").ID, "AddressInfo.ID", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("AddressInfo.ID: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").ID, "AddressInfo.ID", 0);
```

Name, read-only

Beschreibung:

Liefert den Namen der Adresse zurück.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("AddressInfo.Name: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").Name, "AddressInfo.Name", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("AddressInfo.Name: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").Name, "AddressInfo.Name", 0);
```

3.3 Application/cRM Objekt

Bei Scripten, die innerhalb der Anwendung ausgeführt werden, steht Ihnen direkt das **Application**-Objekt als **cRM**-Objekt zur Verfügung. Beim Zugriff von außen benötigen Sie lediglich eine Instanz des **Application**-Objektes. Die sogenannte ProgID des **Application**-Objektes lautet "cRM.Application". Die Instanzierung erfolgt in der Regel über einen CreateObject-Befehl (Late Binding). Die zur Verfügung stehenden Möglichkeiten sowie die genaue Syntax hängen von der verwendeten Programmiersprache ab.

3.3.1 Eigenschaften

AppDir, read-only

Beschreibung:

Liefert den vollständigen Pfad des Hauptverzeichnisses der Anwendung, d.h. dort wo sich die Applikationsdateien (EXE und OVL) befinden: z. B. C:\Program Files (x86)\combit\combit CRM\.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("cRM.AppDir: " & cRM.AppDir, "cRM.AppDir", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("cRM.AppDir: " + cRM.AppDir, "cRM.AppDir", 0);
```

AppTitle, read-only

Beschreibung:

Liefert den Namen der Anwendung zurück.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("cRM.AppTitle: " & cRM.AppTitle, "cRM.AppTitle",
vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("cRM.AppTitle: " + cRM.AppTitle, "cRM.AppTitle", 0);
```

DBServerName, read-only

Beschreibung:

Liefert den konfigurierten Datenbankservernamen zurück.

Hinweis: Der Datenbankservername wird so zurückgegeben, wie er unter **Konto** > **Datenbankverbindung** bzw. per Client-Installation angegeben wurde, inklusive der Groß- und Kleinschreibung. Ein Vergleich mit

bestimmten Werten sollte idealerweise ohne Berücksichtigung von Groß- und Kleinschreibung stattfinden, um ggf. unterschiedliche aber funktionierende Schreibweisen des Datenbankservernamens abzudecken.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("cRM.DBServerName: " & cRM.DBServerName,
"cRM.DBServerName", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("cRM.DBServerName: " + cRM.DBServerName, "cRM.DBServerName",
0);
```

DialogForm, read-only

Beschreibung:

Liefert ein Objekt vom Objekt **DialogForm** zurück. Nach dem Erzeugen des Objekts muss die WScript-Funktion **ConnectObject** aufgerufen werden, um auf die Events des Dialog-Objektes (Button-Clicks) reagieren zu können. Analog dazu muss **DisconnectObject** aufgerufen werden, bevor das Dialog-Objekt wieder freigegeben wird.

Wichtig: Dieses Objekt steht erst ab der Professional-Edition zur Verfügung.

Typ:

DialogForm

Beispiel VBScript:

```
Dim oDialogForm : Set oDialogForm = cRM.DialogForm
'...
Set oDialogForm = Nothing
```

Beispiel C#-Script:

```
DialogForm dialogForm = cRM.DialogForm;
// ...
dialogForm.Dispose();
```

Edition, read-only

Beschreibung:

Liefert die Edition zurück.

Typ:

Long

Wert	Beschreibung
2	Professional-Edition
3	Enterprise-Edition
4	Basic-Edition

Beispiel VBScript:

```
Dim ncRMEdition : ncRMEdition = cRM.Edition
If (ncRMEdition = 2) Then
    Call cRM.DialogMessageBox("Es wird eine Professional-Edition eingesetzt.",
"cRM.Edition", vbOkOnly)
ElseIf (ncRMEdition = 3) Then
    Call cRM.DialogMessageBox("Es wird eine Enterprise-Edition eingesetzt.",
"cRM.Edition", vbOkOnly)
```

```
ElseIf (ncRMEdition = 4) Then
    Call cRM.DialogMessageBox("Es wird eine Basic-Edition eingesetzt.",
"cRM.Edition", vbOkOnly)
End If
```

Beispiel C#-Script:

```
long cRMEdition = cRM.Edition;
if (cRMEdition == 2)
{
    cRM.DialogMessageBox("Es wird eine Professional Edition eingesetzt.",
"cRM.Edition", 0);
}
else if (cRMEdition == 3)
{
    cRM.DialogMessageBox("Es wird eine Enterprise Edition eingesetzt.",
"cRM.Edition", 0);
}
else if (cRMEdition == 4)
{
    cRM.DialogMessageBox("Es wird eine Basic Edition eingesetzt.", "cRM.Edition",
0);
}
```

FileVersion, read-only

Beschreibung:

Rückgabe der Dateiversion der installierten Anwendung.

Typ:

Long

Wert	Beschreibung
HIWORD	Hauptversion (Dividiert durch 65535)
LOWORD	Nebenversion (Modulo 0xFFFF0000)

Beispiel VBScript:

```
Dim scRMFileVersion
Dim ncRMFileVersion : ncRMFileVersion = (cRM.FileVersion Mod &HFFFF0000)
If Len(ncRMFileVersion) = 1 Then
    scRMFileVersion = CStr(CInt(cRM.FileVersion / 65535) & ".00" &
CStr(ncRMFileVersion))
ElseIf Len(ncRMFileVersion) = 2 Then
    scRMFileVersion = CStr(CInt(cRM.FileVersion / 65535) & ".0" &
CStr(ncRMFileVersion))
ElseIf Len(ncRMFileVersion) = 3 Then
    scRMFileVersion = CStr(CInt(cRM.FileVersion / 65535) & "." &
CStr(ncRMFileVersion))
End If
Call cRM.DialogMessageBox("Aktuell installierte Version des combit CRM: " &
scRMFileVersion, "cRM.FileVersion", vbOkOnly)
```

Beispiel C#-Script:

```
long fileVersion = (cRM.FileVersion % 0xFFFF0000);
int fileVersionNumberCount = fileVersion.ToString().Length;

if (fileVersionNumberCount == 1)
    cRM.DialogMessageBox((fileVersion / 65535).ToString() + ".00" +
fileVersion.ToString(), cRM.AppTitle, 0);
else if (fileVersionNumberCount == 2)
    cRM.DialogMessageBox((fileVersion / 65535).ToString() + ".0" +
fileVersion.ToString(), cRM.AppTitle, 0);
else if (fileVersionNumberCount == 3)
    cRM.DialogMessageBox((fileVersion / 65535).ToString() + "." +
fileVersion.ToString(), cRM.AppTitle, 0);
else
    cRM.DialogMessageBox("Information ber Dateiversion nicht verfgbar",
cRM.AppTitle, 0);
```

MainWindowHandle, read-only

Beschreibung:

Liefert das Haupt-Fensterhandle zurück.

Typ:

Long

Beispiel VBScript:

```
Call cRM.DialogMessageBox("cRM.MainWindowHandle: " & CStr(cRM.MainWindowHandle),
"cRM.MainWindowHandle", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("cRM.MainWindowHandle: " + cRM.MainWindowHandle.ToString(),
"cRM.MainWindowHandle", 0);
```

phonemanager, read-only

Beschreibung:

Liefert das phonemanager Objekt vom Typ **phonemanager** zurück.

Typ:

phonemanager

Beispiel VBScript:

```
Dim oPhonemanager : Set oPhonemanager = cRM.phonemanager
'...
Set oPhonemanager = Nothing
```

Beispiel C#-Script:

```
PhoneManager pm = cRM.PhoneManager;
// ...
pm.Dispose();
```

ProductVersion, read-only

Beschreibung:

Rückgabe der Produkt-Versionsnummer der installierten Anwendung.

Typ:

Long

Wert	Beschreibung
HIWORD	Hauptversion (Dividiert durch 65535)
LOWORD	Nebenversion (Modulo 0xFFFF0000)

Beispiel VBScript:

```
Dim scRMProductVersion
Dim ncRMProductVersion : ncRMProductVersion = (cRM.ProductVersion Mod &HFFFF0000)
If Len(ncRMProductVersion) = 1 Then
    scRMProductVersion = CStr(CInt(cRM.ProductVersion / 65535) & ".00" &
CStr(ncRMProductVersion))
ElseIf Len(ncRMProductVersion) = 2 Then
    scRMProductVersion = CStr(CInt(cRM.ProductVersion / 65535) & ".0" &
CStr(ncRMProductVersion))
ElseIf Len(ncRMProductVersion) = 3 Then
    scRMProductVersion = CStr(CInt(cRM.ProductVersion / 65535) & "." &
CStr(ncRMProductVersion))
End If
```

```
Call cRM.DialogMessageBox("Aktuell installierte Version des combit CRM: " &
scRMProductVersion, "cRM.ProductVersion", vbOkOnly)
```

Beispiel C#-Script:

```
long productVersion = (cRM.ProductVersion % 0xFFFF0000);
int productVersionNumberCount = productVersion.ToString().Length;

if (productVersionNumberCount == 1)
    cRM.DialogMessageBox((productVersion / 65535).ToString() + ".00" +
productVersion.ToString(), cRM.AppTitle, 0);
else if (productVersionNumberCount == 2)
    cRM.DialogMessageBox((productVersion / 65535).ToString() + ".0" +
productVersion.ToString(), cRM.AppTitle, 0);
else if (productVersionNumberCount == 3)
    cRM.DialogMessageBox((productVersion / 65535).ToString() + "." +
productVersion.ToString(), cRM.AppTitle, 0);
else
    cRM.DialogMessageBox("Information ber Produktversion nicht verfgbar.",
cRM.AppTitle, 0);
```

ServerAppDir, read-only

Beschreibung:

Rückgabe des Pfads der Serverinstallation. Im Fall einer Einzelplatzversion wird der Wert der Eigenschaft AppDir zurückgegeben.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("cRM.ServerAppDir: " & cRM.ServerAppDir,
"cRM.ServerAppDir", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("cRM.ServerAppDir: " + cRM.ServerAppDir, "cRM.ServerAppDir",
0);
```

TempDir, read-only

Beschreibung:

Rückgabe des Benutzer Temp-Verzeichnisses inkl. Pfad.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("cRM.TempDir: " & cRM.TempDir, "cRM.TempDir", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("cRM.TempDir: " + cRM.TempDir, "cRM.TempDir", 0);
```

TimemanagerType, read-only

Beschreibung:

Rückgabe der eingestellten Terminverwaltung.

Typ:

Long

Wert	Beschreibung
0	Interne Termin- und Aufgabenverwaltung

1	Microsoft Outlook
2	Tobit David
3	IBM Lotus Notes

Beispiel VBScript:

```
Dim ncRMTimeManagerType : ncRMTimeManagerType = cRM.TimemanagerType
If (ncRMTimeManagerType = 0) Then
    Call cRM.DialogMessageBox("Es wird die combit CRM interne Termin- und
Aufgabenverwaltung eingesetzt.", "cRM.TimemanagerType", vbOkOnly)
ElseIf (ncRMTimeManagerType = 1) Then
    Call cRM.DialogMessageBox("Es wird Microsoft Outlook für die Terminverwaltung
eingesetzt.", "cRM.TimemanagerType", vbOkOnly)
ElseIf (ncRMTimeManagerType = 2) Then
    Call cRM.DialogMessageBox("Es wird Tobit David für die Terminverwaltung
eingesetzt.", "cRM.TimemanagerType", vbOkOnly)
ElseIf (ncRMTimeManagerType = 3) Then
    Call cRM.DialogMessageBox("Es wird IBM Lotus Notes für die Terminverwaltung
eingesetzt.", "cRM.TimemanagerType", vbOkOnly)
End If
```

Beispiel C#-Script:

```
long cRMTimeManagerType = cRM.TimeManagerType;

if (cRMTimeManagerType == 0)
    cRM.DialogMessageBox("Es wird die combit CRM interne Termin- und
Aufgabenverwaltung eingesetzt.", "cRM.TimemanagerType", 0);
else if (cRMTimeManagerType == 1)
    cRM.DialogMessageBox("Es wird Microsoft Outlook fr die Terminverwaltung
eingesetzt.", "cRM.TimemanagerType", 0);
else if (cRMTimeManagerType == 2)
    cRM.DialogMessageBox("Es wird Tobit David fr die Terminverwaltung
eingesetzt.", "cRM.TimemanagerType", 0);
else if (cRMTimeManagerType == 3)
    cRM.DialogMessageBox("Es wird IBM Lotus Notes fr die Terminverwaltung
eingesetzt.", "cRM.TimemanagerType", 0);
```

TrialVersion, read-only

Beschreibung:

Abfrage, ob die aktuelle Version eine Demoversion ist.

Typ:

Bool

Beispiel VBScript:

```
Dim bcRMTrialVersion : bcRMTrialVersion = cRM.TrialVersion
If (bcRMTrialVersion = True) Then
    Call cRM.DialogMessageBox("Es wird eine Demoversion des combit CRM
eingesetzt.", "cRM.TrialVersion", vbOkOnly)
ElseIf (bcRMTrialVersion = False) Then
    Call cRM.DialogMessageBox("Es wird lizenzierte Version des combit CRM
eingesetzt.", "cRM.TrialVersion", vbOkOnly)
End If
```

Beispiel C#-Script:

```
bool cRMTrialVersion = cRM.TrialVersion;

if (cRMTrialVersion == true)
    cRM.DialogMessageBox("Es wird eine Demoversion des combit CRM eingesetzt.",
"cRM.TrialVersion", 0);
else if (cRMTrialVersion == false)
    cRM.DialogMessageBox("Es wird lizenzierte Version des combit CRM eingesetzt.",
"cRM.TrialVersion", 0);
```

UILanguageID, read-only

Beschreibung:

Liefert die Sprachkonstante (LANGID gemäß Microsoft Windows SDK) der aktuell eingestellten Sprache des Benutzerinterfaces. Der Wert korrespondiert zu den ".lng"-Unterverzeichnissen der Installation.

Typ:

Long

Wert	Beschreibung
7	Deutsch
9	Englisch

Beispiel VBScript:

```
Dim ncRMUILanguageID : ncRMUILanguageID = cRM.UILanguageID
If (ncRMUILanguageID = 7) Then
    Call cRM.DialogMessageBox("Die Anzeigesprache des combit CRM ist deutsch.",
"cRM.UILanguageID", vbOkOnly)
ElseIf (ncRMUILanguageID = 9) Then
    Call cRM.DialogMessageBox("Die Anzeigesprache des combit CRM ist englisch.",
"cRM.UILanguageID", vbOkOnly)
End If
```

Beispiel C#-Script:

```
int cRMUILanguageID = cRM.UILanguageID;

if (cRMUILanguageID == 7)
    cRM.DialogMessageBox("Die Anzeigesprache des combit CRM ist deutsch.",
"cRM.UILanguageID", 0);
else if (cRMUILanguageID == 9)
    cRM.DialogMessageBox("Die Anzeigesprache des combit CRM ist englisch.",
"cRM.UILanguageID", 0);
```

Version, read-only

Beschreibung:

Rückgabe der Versionsnummer der installierten Anwendung.

Typ:

Long

Wert	Beschreibung
HIWORD	Hauptversion (Dividiert durch 65535), z. B. 2007
LOWORD	Nebenversion (Modulo 0xFFFF0000), z. B. 12

Wichtig: Die Eigenschaft **Version** ist noch aus Gründen der Abwärtskompatibilität enthalten. Für neue Scripte muss die Eigenschaft **FileVersion** verwendet werden.

Beispiel VBScript:

```
Dim scRMVersion
Dim ncRMVersion : ncRMVersion = (cRM.Version Mod &HFFFF0000)
If Len(ncRMVersion) = 1 Then
    scRMVersion = CStr(CInt(cRM.Version / 65535) & ".00" & CStr(ncRMVersion))
ElseIf Len(ncRMVersion) = 2 Then
    scRMVersion = CStr(CInt(cRM.Version / 65535) & ".0" & CStr(ncRMVersion))
ElseIf Len(ncRMVersion) = 3 Then
    scRMVersion = CStr(CInt(cRM.Version / 65535) & "." & CStr(ncRMVersion))
End If
Call cRM.DialogMessageBox("Aktuell installierte Version des combit CRM: " &
ncRMVersion, "cRM.Version", vbOkOnly)
```

Beispiel C#-Script:

```
long version = (cRM.Version % 0xFFFF0000);
int versionNumberCount = version.ToString().Length;
if (versionNumberCount == 1)
    cRM.DialogMessageBox((version / 65535).ToString() + ".00" +
version.ToString(), cRM.AppTitle, 0);
else if (versionNumberCount == 2)
    cRM.DialogMessageBox((version / 65535).ToString() + ".0" + version.ToString(),
cRM.AppTitle, 0);
else if (versionNumberCount == 3)
    cRM.DialogMessageBox((version / 65535).ToString() + "." + version.ToString(),
cRM.AppTitle, 0);
else
    cRM.DialogMessageBox("Information ber Version nicht verfgbar.", cRM.AppTitle,
0);
```

Visible

Beschreibung:

Anzeigezustand der Anwendung. Standardmäßig ist beim Zugriff per COM der Anzeigezustand der Anwendung unsichtbar (**False**).

Typ:

Bool

Beispiel VBScript:

```
Dim bcRMVisible : bcRMVisible = cRM.Visible
If (bcRMVisible = True) Then
    Call cRM.DialogMessageBox("Der combit CRM wird dem Nutzer angezeigt.",
"cRM.Visible", vbOkOnly)
ElseIf (bcRMVisible = False) Then
    ' Anzeigestatus verändern, sodass der combit CRM angezeigt wird
    cRM.Visible = True
    Call cRM.DialogMessageBox("Eine zuvor nicht angezeigte combit CRM-Instanz wird
nun angezeigt.", "cRM.Visible", vbOkOnly)
End If
```

Beispiel C#-Script:

```
bool cRMVisible = cRM.Visible;
if (cRMVisible == true)
    cRM.DialogMessageBox("Der combit CRM wird dem Nutzer angezeigt.",
"cRM.Visible", 0);
else if (cRMVisible == false)
    // Anzeigestatus verändern, sodass der combit CRM angezeigt wird
    cRM.Visible = true;
cRM.DialogMessageBox("Eine zuvor nicht angezeigte combit CRM-Instanz wird nun
angezeigt.", "cRM.Visible", 0);
```

3.3.2 Methoden

CheckAbortedWaitDlg

Beschreibung:

Die Methode muss zwischen **StartWaitDlg** und **EndWaitDlg** aufgerufen werden und gibt zurück, ob der Benutzer die Schaltfläche "Abbrechen" des Wartedialoges betätigt hat, sofern **StartWaitDlg** mit Abbruch-Schaltfläche angezeigt wurde. Die Methode wird vorzugsweise in Ausführungsschleifen verwendet.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Durchlaufen von mehreren Firmen-Datensätzen zur Berechnung der Summe der
Umsatzziele. Für die Laufzeit der Berechnung wird ein Wartedialog aufgerufen,
dessen Inhalt für jeden Durchlauf aktualisiert wird (SetWaitDlgText) und eine
Abbruchbedingung (CheckAbortedWaitDlg) besitzt

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Firmen").CreateRecordSet
Dim oRecord
Dim bSkipped : bSkipped = False
Dim i, nTurnover, nSumOfTurnover
Dim nRecordSetRecCount : nRecordSetRecCount = oRecordSet.RecCount

If (nRecordSetRecCount > 0) Then
    Call cRM.StartWaitDlg("Die Summe der Umsatzziele für alle Datensätze der
Firmen-Ansicht wird berechnet, bitte haben Sie noch etwas Geduld.", True, 0)

    Call oRecordSet.MoveFirst()

    For i = 1 To nRecordSetRecCount
        Call cRM.SetWaitDlgText("Die angeforderten Informationen werden berechnet,
bitte haben Sie noch etwas Geduld." & vbCrLf & vbCrLf & "Fortschritt: Datensatz "
& CStr(i) & " von insgesamt " & oRecordSet.RecCount & " Datensätzen wird derzeit
verarbeitet.")

        Set oRecord = oRecordSet.CurrentRecord
        nTurnover = oRecord.GetContentsValueByName("TurnoverTarget")

        If (IsNull(nTurnover) = False) Then
            nSumOfTurnover = nSumOfTurnover + nTurnover
        End If

        Set oRecord = Nothing

        If (cRM.CheckAbortedWaitDlg = True) Then
            bSkipped = True
            Exit For
        ElseIf (cRM.CheckAbortedWaitDlg = False) Then
            Call oRecordSet.MoveNext()
        End If
    Next

    Call cRM.EndWaitDlg()

    If (bSkipped = True) Then
        Call cRM.DialogMessageBox("Die Summe der Umsatzziele konnte nicht
vollständig berechnet werden. Die Summe vor dem Abbruch der Aktion betrug " &
CStr(nSumOfTurnover) & " EUR.", "Aktion abgebrochen", vbOkOnly)
    ElseIf (bSkipped = False) Then
        Call cRM.DialogMessageBox("Die Summe der Umsatzziele beträgt " &
CStr(nSumOfTurnover) & " EUR.", "Ergebnis der Berechnung", vbOkOnly)
    End If

Else
    Call cRM.DialogMessageBox("Es wurden keine Datensätze für die Berechnung der
Summe der Umsatzziele in der Firmen-Ansicht gefunden.", "Keine Datensätze
gefunden", vbOkOnly)
End If

Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
// Durchlaufen von mehreren Firmen-Datensätzen zur Berechnung der Summe der
Umsatzziele. Für die Laufzeit der Berechnung wird ein Wartedialog aufgerufen,
dessen Inhalt für jeden Durchlauf aktualisiert wird (SetWaitDlgText) und eine
Abbruchbedingung (CheckAbortedWaitDlg) besitzt

RecordSet recordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Firmen").CreateRecordSet();
Record record;
```

```
bool skipped = false;
int sumOfTurnover = 0;

long recordSetRecCount = recordSet.RecCount;

if (recordSetRecCount > 0)
    cRM.StartWaitDlg("Die Summe der Umsatzziele für alle Datensätze der Firmen-
Ansicht wird berechnet, bitte haben Sie noch etwas Geduld.", true, 0);

recordSet.MoveFirst();

for (int i = 1; i <= recordSetRecCount; i++)
{
    cRM.SetWaitDlgText("Die angeforderten Informationen werden berechnet, bitte
haben Sie noch etwas Geduld." + "\r\n" + "\r\n" + "Fortschritt: Datensatz " +
i.ToString() + " von insgesamt " + recordSet.RecCount + " Datensätzen wird derzeit
verarbeitet.");

    record = recordSet.CurrentRecord;
    int turnOver = (int)record.GetContentsValueByName("TurnoverTarget");

    if (turnOver != 0)
        sumOfTurnover += turnOver;

    record.Dispose();

    if (cRM.CheckAbortedWaitDlg() == true)
    {
        skipped = true;
        break;
    }
    else if (cRM.CheckAbortedWaitDlg() == false)
    {
        recordSet.MoveNext();
    }
}

cRM.EndWaitDlg();

if (skipped == true)
    cRM.DialogMessageBox("Die Summe der Umsatzziele konnte nicht vollständig
berechnet werden. Die Summe vor dem Abbruch der Aktion betrug " +
sumOfTurnover.ToString() + " EUR.", "Aktion abgebrochen", 0);
else if (skipped == false)
    cRM.DialogMessageBox("Die Summe der Umsatzziele beträgt " +
sumOfTurnover.ToString() + " EUR.", "Ergebnis der Berechnung", 0);
else
    cRM.DialogMessageBox("Es wurden keine Datensätze für die Berechnung der Summe
der Umsatzziele in der Firmen-Ansicht gefunden.", "Keine Datensätze gefunden", 0);

recordSet.Dispose();
```

ConvertHTML2PlainText

Beschreibung:

Konvertiert einen als HTML übergebenen String nach Klartext.

Parameter:

Parametername	Typ	Beschreibung
sHTML	String	Zu konvertierender HTML-String.

Rückgabewert:

String

Beispiel VBScript:

```
' Eine Benutzereingabe im HTML-Format wird in Klartext umgewandelt

Dim sUserInput : sUserInput = cRM.DialogInputBoxMultiline("Bitte geben Sie HTML-
Quellcode für die Konvertierung in Text an.", "cRM.ConvertHTML2PlainText",
"<html><body><a href=""https://www.combit.net"">Besuchen Sie die combit-
Webseite</a></body></html>")
Call cRM.DialogMessageBox("Der Inhalt des eingegebenen HTMl-Quellcodes wird nach
der Konvertierung wie folgt dargestellt: " & vbCrlf &
cRM.ConvertHTML2PlainText(sUserInput), "cRM.ConvertHTML2PlainText", vbOkOnly)
```

Beispiel C#-Script:

```
// Eine Benutzereingabe im HTML-Format wird in Klartext umgewandelt

string userInput = cRM.DialogInputBoxMultiline("Bitte geben Sie HTML-Quellcode für
die Konvertierung in Text an.", "cRM.ConvertHTML2PlainText", "<html><body><a
href=\"https://www.combit.net\">Besuchen Sie die combit-
Webseite</a></body></html>");
cRM.DialogMessageBox("Der Inhalt des eingegebenen HTMl-Quellcodes wird nach der
Konvertierung wie folgt dargestellt: " + "\r\n" +
cRM.ConvertHTML2PlainText(userInput), "cRM.ConvertHTML2PlainText", 0);
```

ConvertLocalToUTCDateTime

Beschreibung:

Gibt das UTC-Datum und –Zeit zurück. Dies wird für die Konvertierung der Termine und Aufgaben benötigt, da eine Speicherung in die Datenbanktabelle (cmbt_tm_appointments = Termine | cmbt_tm_todos = Aufgabe) im UTC-Format erfolgt. Diese Verwendung finden Sie in der Info-Zentrale des mitgelieferten Beispiel-Projektes.

Parameter:

Parametername	Typ	Beschreibung
LocalDateTime	Date	Lokales Datum und Uhrzeit, z. B. dtmpDate = CStr(Date & " 00:00:00") cRM.ConvertLocalToUTCDateTime(dtmpDate)

Rückgabewert:

Date

Beispiel VBScript:

```
Dim sCurrentUserDate : sCurrentUserDate = FormatDateTime(Date, vbGeneralDate) & "
" & FormatDateTime(Time, vbShortTime)
Call cRM.DialogMessageBox("Der aktuelle Zeitpunkt ist " & sCurrentUserDate &
vbCrlf & "Wert nach der Konvertierung in das UTC-Datum und die UTC-Zeit: " &
cRM.ConvertLocalToUTCDateTime(sCurrentUserDate), "cRM.ConvertLocalToUTCDateTime",
vbOkOnly)
```

Beispiel C#-Script:

```
System.DateTime currentUserDate = System.DateTime.Now;
cRM.DialogMessageBox("Der aktuelle Zeitpunkt ist " +
currentUserDate.ToString("MM/dd/yyyy HH:mm:ss") + "\r\n" + "Wert nach der
Konvertierung in das UTC-Datum und die UTC-Zeit: " +
cRM.ConvertLocalToUTCDateTime(currentUserDate), "cRM.ConvertLocalToUTCDateTime",
0);
```

ConvertPlainText2HTML

Beschreibung:

Konvertiert einen als Klartext übergebenen String nach HTML.

Parameter:

Parametername	Typ	Beschreibung
sPlainText	String	Zu konvertierender Klartext.

Rückgabewert:

String

Beispiel VBScript:

```
' Eine Benutzereingabe im Klartext wird in HTML umgewandelt

Dim sUserInput : sUserInput = cRM.DialogInputBoxMultiline("Eingabe zur
Konvertierung in HTML-Quellcode.", "cRM.ConvertPlainText2HTML", "Der combit CRM
ist eine All-in-one CRM Software mit der Sie alle Kundendaten und Kampagnen im
Blick behalten." & vbCrLf & vbCrLf & "Nutzen Sie die Kommunikationsmöglichkeiten
und das systematische Leadmanagement, um Ihren Erfolg zu steigern.")
Call cRM.DialogMessageBox("Der eingegebene Text wird nach der Konvertierung in
HTML-Quellcode wie folgt dargestellt: " & vbCrlf &
cRM.ConvertPlainText2HTML(sUserInput), "cRM.ConvertPlainText2HTML", vbOkOnly)
```

Beispiel C#-Script:

```
// Eine Benutzereingabe im Klartext wird in HTML umgewandelt

string userInput = cRM.DialogInputBoxMultiline("Eingabe zur Konvertierung in HTML-
Quellcode.", "cRM.ConvertPlainText2HTML", "Der combit CRM ist eine All-in-one CRM
Software mit der Sie alle Kundendaten und Kampagnen im Blick behalten." + "\r\n" +
"\r\n" + "Nutzen Sie die Kommunikationsmöglichkeiten und das systematische
Leadmanagement, um Ihren Erfolg zu steigern.");
cRM.DialogMessageBox("Der eingegebene Text wird nach der Konvertierung in HTML-
Quellcode wie folgt dargestellt: " + "\r\n" +
cRM.ConvertPlainText2HTML(userInput), "cRM.ConvertPlainText2HTML", 0);
```

ConvertUTCToLocalDateTime

Beschreibung:

Gibt das lokale Datum und die Uhrzeit anhand des UTC-Datums zurück.

Parameter:

Parametername	Typ	Beschreibung
UTCDateTime	Date	UTC Datum und Zeit

Rückgabewert:

Date

Beispiel VBScript:

```
Dim sUTCDateTime : sUTCDateTime = CStr(Date) & " " & "09:30:00"
Call cRM.DialogMessageBox("Das vorbelegte Datum ist " & sUTCDateTime & vbCrlf &
"Wert nach der Konvertierung in die lokale Zeitzone: " &
cRM.ConvertUTCToLocalDateTime(sUTCDateTime), "cRM.ConvertUTCToLocalDateTime",
vbOkOnly)
```

Beispiel C#-Script:

```
System.DateTime utcDateTime = System.DateTime.Parse("10.05.2023 09:30:00");
cRM.DialogMessageBox("Das vorbelegte Datum ist " + utcDateTime.ToString() + "\r\n"
+ "Wert nach der Konvertierung in die lokale Zeitzone: " +
cRM.ConvertUTCToLocalDateTime(utcDateTime), "cRM.ConvertUTCToLocalDateTime", 0);
```

CreateGUID

Beschreibung:

Erzeugt einen global eindeutigen Identifier und gibt diesen als unformatierte Zeichenkette zurück.

Rückgabewert:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Es wurde ein global eindeutiger Identifier erzeugt.
Dieser besitzt folgenden Wert: " & vbCrLf & cRM.CreateGUID(), "cRM.CreateGUID",
vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Es wurde ein global eindeutiger Identifier erzeugt. Dieser
besitzt folgenden Wert: " + "\r\n" + cRM.CreateGUID(), "cRM.CreateGUID", 0);
```

CreatePKCEVerifierAndChallenge

Beschreibung:

Erzeugt einen **Proof Key for Code Exchange** gemäß RFC 7636, beispielsweise zur Verwendung in OAuth-Workflows. Der Rückgabewert besteht aus zwei per TAB-Trennzeichen getrennten String-Tokens: Code-Verifier und Code-Challenge. Details: https://www.oauth.com/oauth2-servers/pkce/

Rückgabewert:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Es wurde ein Proof Key for Code Exchange erzeugt.
Dieser besitzt folgenden Wert: " & vbCrLf & cRM.CreatePKCEVerifierAndChallenge(),
"cRM.CreatePKCEVerifierAndChallenge", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Es wurde ein Proof Key for Code Exchange erzeugt. Dieser
besitzt folgenden Wert: " + "\r\n" + cRM.CreatePKCEVerifierAndChallenge (), "
cRM.CreatePKCEVerifierAndChallenge ", 0);
```

CreateTempFile

Beschreibung:

Erzeugt im System-Verzeichnis %TEMP% eine temporäre eindeutige Datei ohne Inhalt und gibt deren Dateiname zurück. In diese Datei kann anschließend bspw. Über die Methode **GetContentsByNameToFile** der Inhalt eines Feldes vom Typ "Eingebettete Datei" oder "Eingebettete Grafik" gespeichert werden ohne dass es Probleme mit einem nicht-eindeutigen Dateinamen gibt.

Parameter:

Parametername	Typ	Beschreibung
sPrefix	String	Präfix des Dateinamens
sFileExtension	String	Dateiendung
bAutoRemove	Bool	True: Die Datei wird beim Beenden der Anwendung wieder weggeräumt. False: Die Datei bleibt beim Beenden der Anwendung bestehen.

Rückgabewert:

String

Hinweis: Der Parameter bAutoRemove hat keine Auswirkung, wenn das Script durch den E-Mail-Autopilot-Dienst ausgeführt wird. In diesem Fall muss die temporäre Datei mittels Scripts gelöscht werden.

Beispiel VBScript:

```
' Ein Foto wird aus der Datenbank ausgelesen und in eine Datei im Dateisystem
geschrieben (Basis: Kontakte-Ansicht einer combit_Large-Solution)
```

```
Dim sFieldName : sFieldName = "Photo"
Dim sFileExtension : sFileExtension = ".jpg"
Dim sTempFileName : sTempFileName = cRM.CreateTempFile("Photo", sFileExtension,
True)
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.GetContentsByNameToFile(sFieldName, sTempFileName)
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Ein Foto wird aus der Datenbank ausgelesen und in eine Datei im Dateisystem
geschrieben (Basis: Kontakte-Ansicht einer combit_Large-Solution)

string fieldName = "Photo";
string fileExtension = ".jpg";
string tempFileName = cRM.CreateTempFile("Photo", fileExtension, true);
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.GetContentsByNameToFile(fieldName, tempFileName);
record.Dispose();
```

CurrentProject

Beschreibung:

Gibt das geladene Projekt als Objekt vom Typ **Project** zurück (leer, wenn kein Projekt geladen).

Rückgabewert:

Project

Beispiel VBScript:

```
Dim oProject : Set oProject = cRM.CurrentProject
'...
Set oProject = Nothing
```

Beispiel C#-Script:

```
Project currentProject = cRM.CurrentProject;
//...
currentProject.Dispose();
```

DebugOutput

Beschreibung:

Erzeugt eine neue Zeile als Debugausgabe in Debwin.

Parameter:

Parametername	Typ	Beschreibung
Output	String	Nachricht für die Debugausgabe. Eine Anpassung des Ereignisprotokollgrades (Information, Warnung, Kritisch) für die Ausgabe in Debwin kann über die folgenden Bezeichner vorgenommen werden: (i) = Information (!) = Warnung (x) = Kritisch Einer dieser Bezeichner kann zu Beginn der Debugausgabe und mit einem Leerzeichen getrennt vom eigentlichen Inhalt verwendet werden.

Hinweis: Beachten Sie bitte, dass Debwin vor der Anwendung gestartet werden muss, damit eine Ausgabe erfolgt.

Beispiel VBScript:

```
Call cRM.DebugOutput("(i) " & FormatDateTime(Date & " " & Time) & " // Der
Datensatz mit der ID " &
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetConten
tsByName("ID") & " befindet sich aktuell in Bearbeitung.")
```

Beispiel C#-Script:

```
cRM.DebugOutput(System.DateTime.Now.ToString() + " // Der Datensatz mit der ID " +
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetConten
tsByName("ID") + " befindet sich aktuell in der Bearbeitung.");
```

DialogAddressAssistant

Beschreibung:

Zeigt den Adresseingabeassistenten für eine übergebene Adresse an oder ergänzt die Adresse mit fehlenden Informationen.

Parameter:

Parametername	Typ	Beschreibung
Mode	Long	Wert 0: Der Adresseingabeassistent wird angezeigt. Wert 1: Der Adresseingabeassistent wird lediglich bei unvollständiger oder mehrdeutiger Adresse angezeigt. Sind alle Bestandteile der Adresse gültig oder kann eine Adresse mit weniger Bestandteilen eindeutig zurückgeliefert werden, wird der Adresseingabeassistent nicht angezeigt.
InputAddress	String	Ermöglicht die TAB-getrennte Übergabe von Adressinformationen im nachfolgenden Format: "Land" & TAB & "Ort" & TAB & "PLZ" & TAB & "Straße" & TAB & "Hausnummer" (je nach Programmiersprache muss TAB durch *vbTab, \t* oder *Chr(9)* ersetzt werden). Nicht vorhandene Adressbestandteile können leer gelassen werden. Wird die Hausnummer nicht separat von der Straße geführt, dann kann dieser Bestandteil leer gelassen und die Hausnummer im Straßenbestandteil zusammen mit der Straße übergeben werden. Wichtig: Die Angabe des Landes gemäß der Länderkonfiguration in den Ansichtseigenschaften ist obligatorisch.

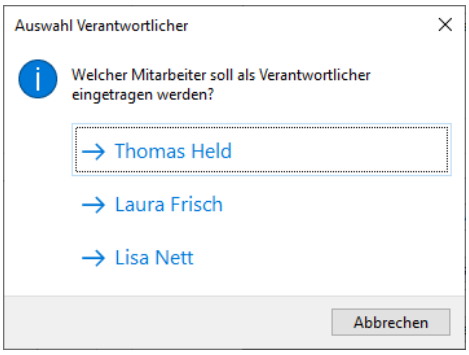
Rückgabewert:

String (TAB-getrennte Liste mit folgenden Bestandteilen: Land, Ort, PLZ, Straße, Hausnummer, Vorwahl, Bundesland/Kanton, Regierungsbezirk, Landkreis, $CANCEL$ bei Abbruch)

DialogChoiceMessageBox

Beschreibung:

Zeigt einen Dialog zur Auswahl von übergebenen Einträgen an.



Hinweis: Weitere Informationen zu den zu verwendenden Konstanten und Rückgabewerten finden sich im Kapitel Script-Konstanten.

Parameter:

Parametername	Typ	Beschreibung
sMessage	String	Nachricht des Dialogs.
sTitle	String	Titel des Dialogs.
sChoices	String	Über diesen Parameter bestimmen Sie tab-getrennt die Auswahlmöglichkeiten. Zum Beispiel: "Thomas Held" & vbTab & "Laura Frisch" & vbTab & "Lisa Nett"
nIcon	Long	Bestimmt das neben der Auswahl dargestellte Symbol: 1: Information 2: Warnung 3: Fehler 4: Schild 5: Frage
bEnableCancel	Bool	Bestimmt, ob eine "Abbrechen"-Schaltfläche angezeigt werden soll.
nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Rückgabewert:

Long bzw. uint (der Rückgabewert entspricht den Konstanten einer MsgBox unter VBScript). Wenn der Nutzer eine Auswahl getroffen hat, dann wird ein OffSet von 100 einberechnet, d.h. bei Klick der zweiten Auswahl ist der Rückgabewert 102.

Beispiel VBScript:

```
Dim nResult : nResult = cRM.DialogChoiceMessageBox("Welcher Mitarbeiter soll als
Verantwortlicher eingetragen werden?", "Auswahl Verantwortlicher", "Thomas Held" &
vbTab & "Laura Frisch" & vbTab & "Lisa Nett", 1, True)
```

Beispiel C#-Script:

```
long result = cRM.DialogChoiceMessageBox("Welcher Mitarbeiter soll als
Verantwortlicher eingetragen werden?", "Auswahl Verantwortlicher", "Thomas Held" +
"\t" + "Laura Frisch" + "\t" + "Lisa Nett", 1, true);
```

DialogInputBox

Beschreibung:

Zeigt einen Eingabedialog an und gibt den eingetragenen Wert zurück.

Parameter:

Parametername	Typ	Beschreibung
sPrompt	String	Information, die im Dialog angezeigt werden soll.
sTitle	String	Optional. Titel des Dialogs.
sDefaultValue	String	Optional. Voreingestellter Wert für die Eingabe.
nInputType	Long	Optional. Reserviert. Der Parameterwert wird derzeit ignoriert.
nMaxInputLength	Long	Optional. Maximale Länge der Benutzereingabe. Voreinstellung: 256
nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Rückgabewert:

String (Benutzereingabe oder $CANCEL$ bei Benutzerabbruch)

Beispiel VBScript:

```
Dim sUserInput : sUserInput = cRM.DialogInputBox("Nach welchem Kontakt-Datensatz
soll gesucht werden?", "Kontakt-Suche...", "Meyer")
```

Beispiel C#-Script:

```
string userInput = cRM.DialogInputBox("Nach welchem Kontakt-Datensatz soll gesucht
werden?", "Kontakt-Suche...", "Meyer");
```

DialogInputBoxMultiline

Beschreibung:

Zeigt einen Eingabedialog mit einem mehrzeiligen Eingabefeld an und gibt den eingetragenen Wert zurück.

Parameter:

Parametername	Typ	Beschreibung
sPrompt	String	Information, die im Dialog angezeigt werden soll. Beachten Sie bitte, dass keine Umbrüche unterstützt werden.
sTitle	String	Optional. Titel des Dialogs.
sDefaultValue	String	Optional. Voreingestellter Wert für die Eingabe.
sSyntaxColoring	String	Optional. Angabe, für welche Syntax ein Highlighting bzw. Coloring erfolgen soll. Mögliche Werte: "sql", "script", "" (wenn leer, wird kein Highlighting bzw. Coloring dargestellt).
nMaxInputLength	Long	Optional. Reserviert. Der Parameterwert wird derzeit ignoriert.
nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Rückgabewert:

String (Benutzereingabe oder $CANCEL$ bei Benutzerabbruch)

Beispiel VBScript:

```
' Eine Benutzereingabe im HTML-Format wird in Klartext umgewandelt
```

```
Dim sUserInput : sUserInput = cRM.DialogInputBoxMultiline("Bitte geben Sie HTML-
Quellcode für die Konvertierung in Text an.", "cRM.ConvertHTML2PlainText",
"<html><body><a href=""https://www.combit.net"">Besuchen Sie die combit-
Webseite</a></body></html>")
Call cRM.DialogMessageBox("Der Inhalt des eingegebenen HTMl-Quellcodes wird nach
der Konvertierung wie folgt dargestellt: " & vbCrlf &
cRM.ConvertHTML2PlainText(sUserInput), "cRM.ConvertHTML2PlainText", vbOkOnly)
```

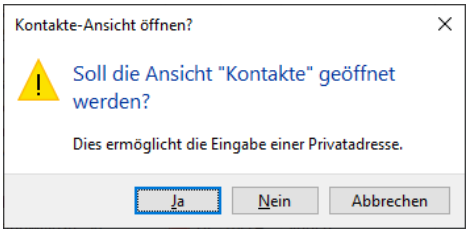
Beispiel C#-Script:

```
// Eine Benutzereingabe im HTML-Format wird in Klartext umgewandelt

string userInput = cRM.DialogInputBoxMultiline("Bitte geben Sie HTML-Quellcode fr
die Konvertierung in Text an.", "cRM.ConvertHTML2PlainText", "<html><body><a
href=\"https://www.combit.net\">Besuchen Sie die combit-
Webseite</a></body></html>");
cRM.DialogMessageBox("Der Inhalt des eingegebenen HTMl-Quellcodes wird nach der
Konvertierung wie folgt dargestellt: " + "\r\n" +
cRM.ConvertHTML2PlainText(userInput), "cRM.ConvertHTML2PlainText", 0);
```

DialogMessageBox

Beschreibung:

Zeigt einen Dialog zur Ausgabe von Informationen im cRM-Stil an.



Hinweis: Weitere Informationen zu den zu verwendenden Konstanten und Rückgabewerten finden sich im Kapitel Script-Konstanten.

Parameter:

Parametername	Typ	Beschreibung
sMessage	String	Nachricht des Dialogs. Bitte beachten Sie, dass bei der Verwendung eines Zeilenumbruchs (\n bzw. vbCrLf) der erste Abschnitt bis zum Zeilenumbruch in einer größeren Schrift dargestellt wird. Der nachfolgende Abschnitt erhält eine kleinere Schriftgröße.
sTitle	String	Titel des Dialogs.
nStyle	Long	Dieser Parameter bestimmt mittels Übergabe von Konstanten das Aussehen des Dialogs. Zum Beispiel: vbYesNoCancel bzw. den Enumerationswert 3.
nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Rückgabewert:

Long bzw. uint (der Rückgabewert entspricht den Konstanten einer MsgBox unter VBScript)

Beispiel VBScript:

```
Dim nResult : nResult = cRM.DialogMessageBox("Soll die Ansicht ""Kontakte""
geöffnet werden?" & vbCrLf & "Dies ermöglicht die Eingabe einer Privatadresse.",
"Kontakte-Ansicht öffnen?", vbExclamation + vbYesNoCancel)
```

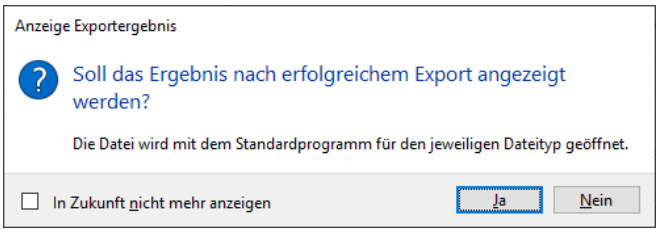
Beispiel C#-Script:

```
long result = cRM.DialogMessageBox("Soll die Ansicht \"Kontakte\" geöffnet
werden?" + "\r\n" + "Dies ermöglicht die Eingabe einer Privatadresse.", "Kontakte-
Ansicht öffnen?", 48 + 3);
```

DialogMessageBoxAuto

Beschreibung:

Zeigt einen Dialog zur Ausgabe von Informationen im cRM-Stil an. Zusätzlich wird eine Option "In Zukunft nicht mehr anzeigen" bereitgestellt.



Hinweis: Weitere Informationen zu den zu verwendenden Konstanten und Rückgabewerten finden sich im Kapitel **Script-Konstanten**.

Parameter:

Parametername	Typ	Beschreibung
sMessage	String	Nachricht des Dialogs. Bitte beachten Sie, dass bei der Verwendung eines Zeilenumbruchs (\n bzw. vbCrLf) der erste Abschnitt bis zum Zeilenumbruch in einer größeren Schrift dargestellt wird. Der nachfolgende Abschnitt erhält eine kleinere Schriftgröße.
sTitle	String	Titel des Dialogs.
nStyle	Long	Dieser Parameter bestimmt mittels Übergabe von Konstanten das Aussehen des Dialogs. Zum Beispiel: vbYesNoCancel bzw. den Enumerationswert 3.
sMessageID	String	Eindeutige ID in Form einer Zeichenkette (String) zur Speicherung der Auswahl für den Wert "In Zukunft nicht mehr anzeigen".
nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Rückgabewert:

Long bzw. uint (der Rückgabewert entspricht den Konstanten einer MsgBox unter VBScript)

Beispiel VBScript:

```
Dim nResult : nResult = cRM.DialogMessageBoxAuto("Soll das Ergebnis nach
erfolgreichem Export angezeigt werden?" & vbCrLf & "Die Datei wird mit dem
Standardprogramm für den jeweiligen Dateityp geöffnet.", "Anzeige Exportergebnis",
vbQuestion + vbYesNo, "ShowExportResult")
```

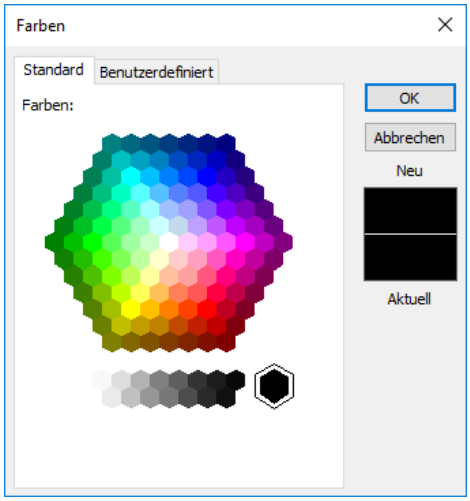
Beispiel C#-Script:

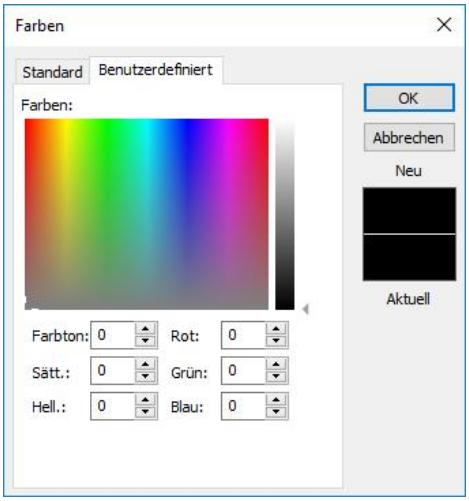
```
long result = cRM.DialogMessageBoxAuto("Soll das Ergebnis nach erfolgreichem
Export angezeigt werden?" + "\r\n" + "Die Datei wird mit dem Standardprogramm für
den jeweiligen Dateityp geöffnet.", "Anzeige Exportergebnis", 0,
"ShowExportResult");
```

DialogSelectColor

Beschreibung:

Zeigt einen Farbauswahldialog an und gibt den ausgewählten Farbwert zurück.





Parameter:

Parametername	Typ	Beschreibung
nCurrentColor	Long	Optional. Aktuelle Farbe. Wenn dieser Parameter nicht übergeben wird, wird Schwarz verwendet.
nDefaultColor	Long	Optional. Neue Farbe, die vorausgewählt werden soll. Wenn dieser Parameter nicht angegeben wird, wird die aktuelle Farbe (vgl. Parameter nCurrentColor) verwendet.
nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Rückgabewert:

Long (ausgewählter Farbwert oder -1 bei Benutzerabbruch)

Beispiel VBScript:

```
Dim nChosenColor : nChosenColor = cRM.DialogSelectColor(33023, 0)
```

Beispiel C#-Script:

```
long chosenColor = cRM.DialogSelectColor(33023, 0);
```

DialogSelectDir

Beschreibung:

Zeigt einen Verzeichnisauswahldialog an und gibt den ausgewählten Verzeichnispfad zurück.

Parameter:

Parametername	Typ	Beschreibung
sTitle	String	Dialogtitel
bOpenFileDlg	Bool	True: Verzeichnis Öffnen False: Verzeichnis Speichern
sInitialPath	String	Initielles Verzeichnis als Voreinstellung für den Dialog oder leer.
dwFlags	Long	Reserviert, muss 0 sein.

nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Rückgabewert:

String (ausgewählter Verzeichnispfad oder leer bei Benutzerabbruch)

Beispiel VBScript:

```
Dim sDir
sDir = cRM.DialogSelectDir("Quell-Verzeichnis auswählen", True, "", 0)
sDir = cRM.DialogSelectDir("Ziel-Verzeichnis auswählen", False, "C:\temp", 0)
```

Beispiel C#-Script:

```
string dir;
dir = cRM.DialogSelectDir("Quell-Verzeichnis auswählen", true, "", 0);
dir = cRM.DialogSelectDir("Ziel-Verzeichnis auswählen", false, "C:\temp", 0);
```

DialogSelectFile

Beschreibung:

Zeigt einen Dateiauswahldialog an und gibt den ausgewählten Dateipfad zurück.

Parameter:

Parametername	Typ	Beschreibung
sTitle	String	Dialogtitel
bOpenFileDlg	Bool	True: Datei öffnen Dialog False: Datei speichern Dialog
sInitialPath	String	Initielles Verzeichnis als Voreinstellung für den Dialog. Kann einen kompletten Dateipfad enthalten oder leer sein.
sFileFilter	String	Kann eine Liste von Dateifiltern enthalten oder leer sein. Format: BeschreibungFilter1|DateiwildcardsFilter1|BeschreibungFilter2|DateiwildcardsFilter2|...||
dwFlags	Long	Kann 0 oder eine Kombination (Addition/verODERung) folgender Zahlenwerte sein (Beschreibung siehe MSDN unter 'OPENFILENAME'): const OFN_FILEMUSTEXIST = 4096 const OFN_CREATEPROMPT = 8192 const OFN_HIDEREADONLY = 4 const OFN_OVERWRITEPROMPT = 2 const OFN_NOREADONLYRETURN = 32768
nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Rückgabewert:

String (ausgewählter Dateipfad oder leer bei Benutzerabbruch)

Beispiel VBScript:

```
' Es werden zwei Dialoge angezeigt: 1. Auswahl einer Datei in einem vordefinierten
Verzeichnis mit einem Dateifilter, 2. Speichern einer Datei mit einem
vordefinierten Pfad im Dateiformat .txt

const OFN_FILEMUSTEXIST = 4096
const OFN_CREATEPROMPT = 8192
const OFN_HIDEREADONLY = 4
const OFN_OVERWRITEPROMPT = 2
const OFN_NOREADONLYRETURN = 32768
sFile = cRM.DialogSelectFile("Druckvorlage auswählen", True, "C:\temp",
"Druckvorlagen (*.lbl;*.crd;*.lst)|*.lbl;*.crd;*.lst|Alle Dateien|*.*||",
OFN_FILEMUSTEXIST)
sFile = cRM.DialogSelectFile("Datei speichern unter", False, "C:\temp\neu.txt",
"Textdateien (*.txt)|*.txt|Alle Dateien|*.*||",
OFN_OVERWRITEPROMPT+OFN_HIDEREADONLY+OFN_NOREADONLYRETURN)
```

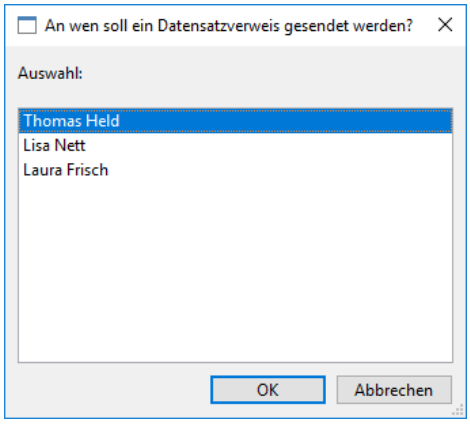
Beispiel C#-Script:

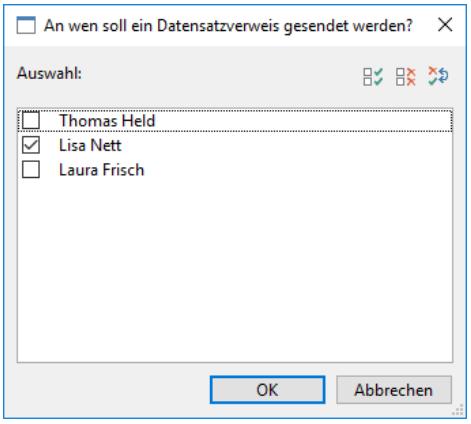
```
// Es werden zwei Dialoge angezeigt: 1. Auswahl einer Datei in einem
vordefinierten Verzeichnis mit einem Dateifilter, 2. Speichern einer Datei mit
einem vordefinierten Pfad im Dateiformat .txt

int OFN_FILEMUSTEXIST = 4096;
int OFN_CREATEPROMPT = 8192;
int OFN_HIDEREADONLY = 4;
int OFN_OVERWRITEPROMPT = 2;
int OFN_NOREADONLYRETURN = 32768;
string file = string.Empty;
file = cRM.DialogSelectFile("Druckvorlage auswählen", true, "C:\temp",
"Druckvorlagen (*.lbl;*.crd;*.lst)|*.lbl;*.crd;*.lst|Alle Dateien|*.*||",
OFN_FILEMUSTEXIST);
file = cRM.DialogSelectFile("Datei speichern unter", false, "C:\temp\neu.txt",
"Textdateien (*.txt)|*.txt|Alle Dateien|*.*||", OFN_OVERWRITEPROMPT +
OFN_HIDEREADONLY + OFN_NOREADONLYRETURN);
```

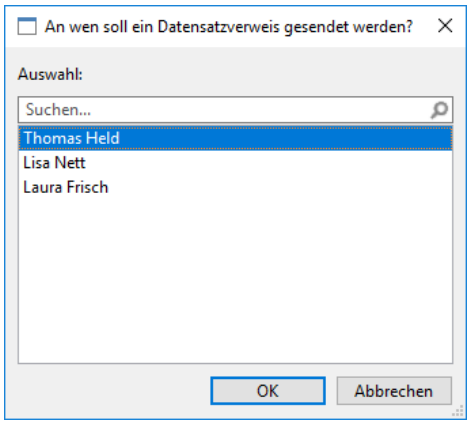
DialogSelectString

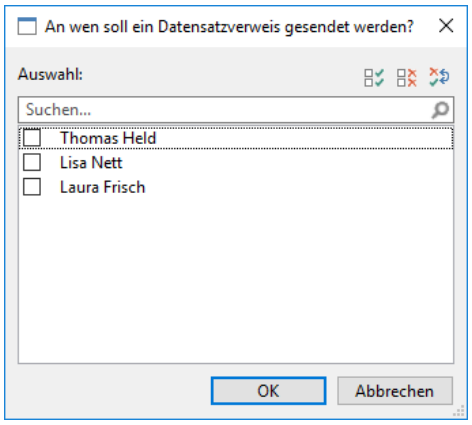
Beschreibung:

Zeigt einen Dialog zur Auswahl eines oder mehrerer Strings an.









Parameter:

Parametername	Typ	Beschreibung
sMessage	String	Information oder Frage, die im Titel des Dialogs angezeigt werden soll.
sChoices	String	TAB-getrennte Liste mit Auswahlmöglichkeiten. Beginnt ein TAB-getrennter Eintrag mit einem *, dann wird dieser Eintrag vorausgewählt.
nSortChoices	Long	0: Reihenfolge der Einträge wird nicht sortiert. 1: Alphabetisch aufsteigende Reihenfolge der Einträge (keine Beachtung von Groß- und Kleinschreibung). -1: Alphabetisch absteigende Reihenfolge der Einträge (keine Beachtung von Groß- und Kleinschreibung).
bAllowMultiSelect	Bool	True: Mehrfachauswahl von Einträgen ist möglich. False: Mehrfachauswahl von Einträgen ist nicht möglich
bAllowFilter	Bool	True: Suche innerhalb der Auflistung der Einträge wird ermöglicht. False: Suche innerhalb der Auflistung der Einträge ist nicht möglich.
sProfileKey	String	Name, unter welchem innerhalb der Registrierung im nachfolgenden Schlüssel die letzte Auswahl gespeichert und beim nächsten Mal automatisch geladen werden soll. Wenn neue Einträge mit einem * übergeben wurden, sodass diese vorausgewählt angezeigt werden sollen, dann überschreiben die neuen Einträge die bestehende Vorauswahl aus der Registrierung. Folgender Registrierungsschlüssel wird verwendet: HKEY_CURRENT_USER\Software\combit\combit Relationship Manager\Projects\<Name des Projektes>\COMDialogSelectString
nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Rückgabewert:

String (ausgewählte Strings (TAB-getrennt bei Mehrfachauswahl) oder $CANCEL$ bei Benutzerabbruch)

Beispiel VBScript:

```
Dim sChoice
sChoice = cRM.DialogSelectString("An wen soll ein Datensatzverweis gesendet
werden?", "Thomas Held" & vbTab & "Lisa Nett" & vbTab & "Laura Frisch", 0, False,
False, "")
sChoice = cRM.DialogSelectString("An wen soll ein Datensatzverweis gesendet
werden?", "Thomas Held" & vbTab & "*Lisa Nett" & vbTab & "Laura Frisch", 0, True,
False, "Datensatzverweis")
sChoice = cRM.DialogSelectString("An wen soll ein Datensatzverweis gesendet
werden?", "*Thomas Held" & vbTab & "Lisa Nett" & vbTab & "Laura Frisch", 0, Talse,
True, "")
sChoice = cRM.DialogSelectString("An wen soll ein Datensatzverweis gesendet
werden?", "Thomas Held" & vbTab & "Lisa Nett" & vbTab & "Laura Frisch", 0, True,
True, "Datensatzverweis")
```

Beispiel C#-Script:

```
string choice;
choice = cRM.DialogSelectString("An wen soll ein Datensatzverweis gesendet
werden?", "Thomas Held" + "\t" + "Lisa Nett" + "\t" + "Laura Frisch", 0, false,
false, "");
choice = cRM.DialogSelectString("An wen soll ein Datensatzverweis gesendet
werden?", "Thomas Held" + "\t" + "*Lisa Nett" + "\t" + "Laura Frisch", 0, true,
false, "Datensatzverweis");
choice = cRM.DialogSelectString("An wen soll ein Datensatzverweis gesendet
werden?", "*Thomas Held" + "\t" + "Lisa Nett" + "\t" + "Laura Frisch", 0, false,
true, "");
choice = cRM.DialogSelectString("An wen soll ein Datensatzverweis gesendet
werden?", "Thomas Held" + "\t" + "Lisa Nett" + "\t" + "Laura Frisch", 0, true,
true, "Datensatzverweis");
```

EndWaitDlg

Beschreibung:

Blendet einen zuvor mit **StartWaitDlg** angezeigten Wartedialog aus.

Beispiel VBScript:

```
' Durchlaufen von mehreren Firmen-Datensätzen zur Berechnung der Summe der
Umsatzziele. Für die Laufzeit der Berechnung wird ein Wartedialog aufgerufen,
dessen Inhalt für jeden Durchlauf aktualisiert wird (SetWaitDlgText) und eine
Abbruchbedingung (CheckAbortedWaitDlg) besitzt.

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Firmen").CreateRecordSet
Dim oRecord
Dim bSkipped : bSkipped = False
Dim i, nTurnover, nSumOfTurnover
Dim nRecordSetRecCount : nRecordSetRecCount = oRecordSet.RecCount

If (nRecordSetRecCount > 0) Then
    Call cRM.StartWaitDlg("Die Summe der Umsatzziele fr alle Datensätze der
Firmen-Ansicht wird berechnet, bitte haben Sie noch etwas Geduld.", True, 0)

    Call oRecordSet.MoveFirst()

    For i = 1 To nRecordSetRecCount
        Call cRM.SetWaitDlgText("Die angeforderten Informationen werden berechnet,
bitte haben Sie noch etwas Geduld." & vbCrLf & vbCrLf & "Fortschritt: Datensatz "
& CStr(i) & " von insgesamt " & oRecordSet.RecCount & " Datensätzen wird derzeit
verarbeitet.")

        Set oRecord = oRecordSet.CurrentRecord
```

```
            nTurnover = oRecord.GetContentsValueByName("TurnoverTarget")

            If (IsNull(nTurnover) = False) Then
                nSumOfTurnover = nSumOfTurnover + nTurnover
            End If

            Set oRecord = Nothing

            If (cRM.CheckAbortedWaitDlg = True) Then
                bSkipped = True
                Exit For
            ElseIf (cRM.CheckAbortedWaitDlg = False) Then
                Call oRecordSet.MoveNext()
            End If
        Next

        Call cRM.EndWaitDlg()

        If (bSkipped = True) Then
            Call cRM.DialogMessageBox("Die Summe der Umsatzziele konnte nicht
    vollständig berechnet werden. Die Summe vor dem Abbruch der Aktion betrug " &
    CStr(nSumOfTurnover) & " EUR.", "Aktion abgebrochen", vbOkOnly)
        ElseIf (bSkipped = False) Then
            Call cRM.DialogMessageBox("Die Summe der Umsatzziele beträgt " &
    CStr(nSumOfTurnover) & " EUR.", "Ergebnis der Berechnung", vbOkOnly)
        End If

    Else
        Call cRM.DialogMessageBox("Es wurden keine Datensätze für die Berechnung der
    Summe der Umsatzziele in der Firmen-Ansicht gefunden.", "Keine Datensätze
    gefunden", vbOkOnly)
    End If

    Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
    // Durchlaufen von mehreren Firmen-Datensätzen zur Berechnung der Summe der
    Umsatzziele. Für die Laufzeit der Berechnung wird ein Wartedialog aufgerufen,
    dessen Inhalt für jeden Durchlauf aktualisiert wird (SetWaitDlgText) und eine
    Abbruchbedingung (CheckAbortedWaitDlg) besitzt.

    RecordSet recordSet =
    cRM.CurrentProject.ViewConfigs.ItemByName("Firmen").CreateRecordSet();
    Record record;
    bool skipped = false;
    int sumOfTurnover = 0;

    long recordSetRecCount = recordSet.RecCount;

    if (recordSetRecCount > 0)
        cRM.StartWaitDlg("Die Summe der Umsatzziele fr alle Datensätze der Firmen-
    Ansicht wird berechnet, bitte haben Sie noch etwas Geduld.", true, 0);

    recordSet.MoveFirst();

    for (int i = 1; i <= recordSetRecCount; i++)
    {
        cRM.SetWaitDlgText("Die angeforderten Informationen werden berechnet, bitte
    haben Sie noch etwas Geduld." + "\r\n" + "\r\n" + "Fortschritt: Datensatz " +
    i.ToString() + " von insgesamt " + recordSet.RecCount + " Datensätzen wird derzeit
    verarbeitet.");

        record = recordSet.CurrentRecord;
        int turnOver = (int)record.GetContentsValueByName("TurnoverTarget");

        if (turnOver != 0)
            sumOfTurnover += turnOver;

        record.Dispose();
```

```
        if (cRM.CheckAbortedWaitDlg() == true)
        {
            skipped = true;
            break;
        }
        else if (cRM.CheckAbortedWaitDlg() == false)
        {
            recordSet.MoveNext();
        }
    }

    cRM.EndWaitDlg();

    if (skipped == true)
        cRM.DialogMessageBox("Die Summe der Umsatzziele konnte nicht vollständig
    berechnet werden. Die Summe vor dem Abbruch der Aktion betrug " +
    sumOfTurnover.ToString() + " EUR.", "Aktion abgebrochen", 0);
    else if (skipped == false)
        cRM.DialogMessageBox("Die Summe der Umsatzziele beträgt " +
    sumOfTurnover.ToString() + " EUR.", "Ergebnis der Berechnung", 0);
    else
        cRM.DialogMessageBox("Es wurden keine Datensätze für die Berechnung der Summe
    der Umsatzziele in der Firmen-Ansicht gefunden.", "Keine Datensätze gefunden", 0);

    recordSet.Dispose();
```

FetchGlobalConfigFile

Beschreibung:

Holt die globalen Konfigurationsdateien ohne Projekt-ID.

Parameter:

Parametername	Typ	Beschreibung
sSrcFilePath	String	Dateiname in der cmbt_Files Tabelle in der System-Datenbank
sDestFilePath	String	Dateiname, unter dem die Datei abgespeichert werden soll.

Rückgabewert:

Bool

Wert	Beschreibung
True	Befehl wurde ausgeführt.
False	Befehl konnte nicht ausgeführt werden, dies deutet darauf hin, dass einer der Parameterwerte ungültig ist (Dateiname in cmbt_Files Tabelle kann nicht gefunden werden, Pfad für Speicherort ist ungültig, ...).

Beispiel VBScript:

```
' Die Konfigurationsdatei global.ini wird aus der Datenbanktabelle cmbt_Files in
das Dateisystem geschrieben

Dim sSourceFileName : sSourceFileName = "global.ini"
Dim sDestinationPath : sDestinationPath = "C:\" & sSourceFileName
Dim bSuccess : bSuccess = cRM.FetchGlobalConfigFile(sSourceFileName,
sDestinationPath)

If (bSuccess = True) Then
    Call cRM.DialogMessageBox("Die Datei """ & sSourceFileName & """ konnte
erfolgreich abgelegt werden: " & vbCrlf & "" & sDestinationPath & "",
"cRM.FetchGlobalConfigFile", vbOkOnly)
ElseIf (bSuccess = False) Then
    Call cRM.DialogMessageBox("Die Datei """ & sSourceFileName & """ konnte nicht
erfolgreich abgelegt werden.", "cRM.FetchGlobalConfigFile", vbOkOnly)
End If
```

Beispiel C#-Script:

```
// Die Konfigurationsdatei global.ini wird aus der Datenbanktabelle cmbt_Files in
das Dateisystem geschrieben

string sourceFileName = "global.ini";
string destinationPath = @"C:\" + sourceFileName;
bool success = cRM.FetchGlobalConfigFile(sourceFileName, destinationPath);

if (success == true)
    cRM.DialogMessageBox("Die Datei \"" + sourceFileName + "\" konnte erfolgreich
abgelegt werden: " + "\r\n" + "" + destinationPath + "",
"cRM.FetchGlobalConfigFile", 0);
else if (success == false)
    cRM.DialogMessageBox("Die Datei \"" + sourceFileName + "\" konnte nicht
erfolgreich abgelegt werden.", "cRM.FetchGlobalConfigFile", 0);
```

GetcRMByProcessID

Beschreibung:

Liefert ein bestimmtes cRM-Objekt anhand seiner ProcessID zurück.

Parameter:

Parametername	Typ	Beschreibung
nProcessID	DWORD	ProcessID

Rückgabewert:

Application, NULL (falls kein Prozess mit dieser ID existiert)

Beispiel VBScript:

```
Dim nProcessID : nProcessID = 3984 ' Dieser Wert ist für jede combit CRM-Instanz
verschieden
' Auf eine der geöffneten combit CRM-Instanzen zugreifen
Dim ocRM : Set ocRM = GetObject(, "cRM.Application")
Set ocRM = ocRM.GetcRMByProcessID(nProcessID)

If (Not ocRM Is Nothing) Then
    Dim oProject : Set oProject = ocRM.Login("", "", "")
    Call ocRM.DialogMessageBox("Das aktuell geladene Projekt hat folgenden Namen:
" & oProject.Name, "cRM.GetcRMByProcessID", vbOkOnly)
    Set oProject = Nothing
End If

Set ocRM = Nothing
```

GetIniProfileString

Beschreibung:

Liest einen Eintrag aus einer INI-Datei aus.

Parameter:

Parametername	Typ	Beschreibung
sSection	String	Name der Sektion in der INI-Datei, in der der zu lesende Eintrag steht.
sProperty	String	Name des zu lesenden Eintrags.
sDefaultValue	String	Der Standardwert, der zurückgegeben wird, wenn kein Eintrag mit dem angegebenen Namen existiert.
sIniFile	String	Pfad zur INI-Datei.

Rückgabewert:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Folgender Wert wurde für den Eintrag
""GDPRLogTriggerActivatedByEvent_combit_Large"" ausgelesen: " &
cRM.GetIniProfileString("Scripting", "GDPRLogTriggerActivatedByEvent_combitLarge",
"Kein Wert gefunden", "global.ini"), "cRM.GetIniProfileString", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Folgender Wert wurde für den Eintrag
\"GDPRLogTriggerActivatedByEvent_combit_Large\" ausgelesen: " +
cRM.GetIniProfileString("Scripting", "GDPRLogTriggerActivatedByEvent_combitLarge",
"Kein Wert gefunden", "global.ini"), "cRM.GetIniProfileString", 0);
```

HTTPDelete

Beschreibung:

Sendet einen HTTP-Request mit einer DELETE-Methode.

Hinweis: Im Unterordner Scripts\ v_JSON Ihrer combit CRM-Installation finden Sie Scripte, die das Handling mit JSON-Strings vereinfachen. Diese können beispielsweise per <!--#include-once file="..."--> eingebunden werden. Im Kapitel **HTTP-Requests ausführen** finden Sie ein Beispiel zur Vorgehensweise bei der Nutzung von HTTP-Requests.

Parameter:

Parametername	Typ	Beschreibung
sUrl	String	URL des HTTP-Servers.

sHeaderData	String	JSON-formatierte Zeichenkette. Muss ein Array aus Objekten mit jeweils einem Key und einem Value enthalten.
		Beispiel:
		`[`
		`{`
		`""key"": ""custom_key""`,
		`""value"": ""custom_value""`
		`},`
		`{`
		`""key"": ""Content-Type""`,
		`""value"": ""application/json""`
		`}`
		`]`

Rückgabewert:

Im Erfolgsfall: String (JSON-formatiert, enthält die Objekte "status" [Statuscode] und "response" [Antwort des Servers])

Im Fehlerfall: String (JSON-formatiert), mit folgendem Aufbau:

{
"status": -1; *// Interner / combit Fehler*
"error": {
"type": number; *// Kann momentan die Werte 0 (UNKNOWN), 1 (INVALID_ARGUMENT), 2 (JSON_ERROR) oder 3 (HTTP_ERROR) enthalten.*
"type_string": string; *// Entspricht dem Namen des Fehlers, welcher in 'type' enthalten ist.*
"error_details": string; *// Nur bei type == 2 vorhanden, enthält weitere Informationen wieso die JSON-Formatierung der Serverantwort fehlgeschlagen ist.*

"response": string; *// Nur bei type == 2 vorhanden, enthält die Serverantwort, allerdings OHNE die Zeichen, welche den aktuellen JSON_ERROR ausgelöst haben. Dient nur zur Information und sollte nicht zur weiteren Datenverarbeitung genutzt werden.*
}
}

HTTPGet

Beschreibung:

Sendet einen HTTP-Request mit einer GET-Methode.

Hinweis: Im Unterordner Scripts\ v_JSON Ihrer combit CRM-Installation finden Sie Scripte, die das Handling mit JSON-Strings vereinfachen. Diese können beispielsweise per <!--#include-once file="..."--> eingebunden werden. Im Kapitel **HTTP-Requests ausführen** finden Sie ein Beispiel zur Vorgehensweise bei der Nutzung von HTTP-Requests.

Parameter:

Parametername	Typ	Beschreibung
sUrl	String	URL des HTTP-Servers.

sHeaderData	String	JSON-formatierte Zeichenkette. Muss ein Array aus Objekten mit jeweils einem Key und einem Value enthalten.
		Beispiel:
		`[`
		`{`
		`""key"": ""custom_key"",`
		`""value"": ""custom_value""`
		`},`
		`{`
		`""key"": ""Content-Type"",`
		`""value"": ""application/json""`
		`}`
		`]`

Rückgabewert:

Im Erfolgsfall: String (JSON-formatiert, enthält die Objekte "status" [Statuscode] und "response" [Antwort des Servers])

Im Fehlerfall: String (JSON-formatiert), mit folgendem Aufbau:

{
"status": -1; *// Interner / combit Fehler*
"error": {
"type": number; *// Kann momentan die Werte 0 (UNKNOWN), 1 (INVALID_ARGUMENT), 2 (JSON_ERROR) oder 3 (HTTP_ERROR) enthalten.*
"type_string": string; *// Entspricht dem Namen des Fehlers, welcher in 'type' enthalten ist.*
"error_details": string; *// Nur bei type == 2 vorhanden, enthält weitere Informationen wieso die JSON-Formatierung der Serverantwort fehlgeschlagen ist.*
"response": string; *// Nur bei type == 2 vorhanden, enthält die Serverantwort, allerdings OHNE die Zeichen, welche den aktuellen JSON_ERROR ausgelöst haben. Dient nur zur Information und sollte nicht zur weiteren Datenverarbeitung genutzt werden.*
}

}

HTTPPatch

Beschreibung:

Sendet einen HTTP-Request mit einer PATCH-Methode.

Hinweis: Im Unterordner Scripts\ v_JSON Ihrer combit CRM-Installation finden Sie Scripte, die das Handling mit JSON-Strings vereinfachen. Diese können beispielsweise per <!--#include-once file="..."--> eingebunden werden. Im Kapitel **HTTP-Requests ausführen** finden Sie ein Beispiel zur Vorgehensweise bei der Nutzung von HTTP-Requests.

Parameter:

Parametername	Typ	Beschreibung
sUrl	String	URL des HTTP-Servers.

sHeaderData	String	JSON-formatierte Zeichenkette. Muss ein Array aus Objekten mit jeweils einem Key und einem Value enthalten.
		Beispiel:
		`[`
		`{`
		`""key"": ""custom_key"",`
		`""value"": ""custom_value""`
		`},`
		`{`
		`""key"": ""Content-Type"",`
		`""value"": ""application/json""`
		`}`
		`]`

sBodyData	String	JSON-formatierte Zeichenkette. Muss ein Array aus Objekten mit jeweils einem Key und einem Value enthalten.
		Beispiel:
		`[`
		`{`
		`""key"": """",`
		`""value"": {""json_object_key\"":""json_object_value""}`
		`}`
		`]`
		Dieses Beispiel verdeutlicht die Vorgehensweise wenn **Content-Type:application/json-Header** genutzt wird. Abweichende Content-Types benötigen mitunter abweichende Inhalte.

Rückgabewert:

Im Erfolgsfall: String (JSON-formatiert, enthält die Objekte "status" [Statuscode] und "response" [Antwort des Servers])

Im Fehlerfall: String (JSON-formatiert), mit folgendem Aufbau:

{

"status": -1; *// Interner / combit Fehler*
"error": {
"type": number; *// Kann momentan die Werte 0 (UNKNOWN), 1 (INVALID_ARGUMENT), 2 (JSON_ERROR) oder 3 (HTTP_ERROR) enthalten.*
"type_string": string; *// Entspricht dem Namen des Fehlers, welcher in 'type' enthalten ist.*
"error_details": string; *// Nur bei type == 2 vorhanden, enthält weitere Informationen wieso die JSON-Formatierung der Serverantwort fehlgeschlagen ist.*
"response": string; *// Nur bei type == 2 vorhanden, enthält die Serverantwort, allerdings OHNE die Zeichen, welche den aktuellen JSON_ERROR ausgelöst haben. Dient nur zur Information und sollte nicht zur weiteren Datenverarbeitung genutzt werden.*
}
}

HTTPPost

Beschreibung:

Sendet einen HTTP-Request mit einer POST-Methode.

Hinweis: Im Unterordner Scripts\ v_JSON Ihrer combit CRM-Installation finden Sie Scripte, die das Handling mit JSON-Strings vereinfachen. Diese können beispielsweise per <!--#include-once file="..."--> eingebunden werden. Im Kapitel HTTP-Requests ausführen finden Sie ein Beispiel zur Vorgehensweise bei der Nutzung von HTTP-Requests.

Parameter:

Parametername	Typ	Beschreibung
sUrl	String	URL des HTTP-Servers.

sHeaderData	String	JSON-formatierte Zeichenkette. Muss ein Array aus Objekten mit jeweils einem Key und einem Value enthalten.
		Beispiel:
		`[`
		`{`
		`""key"": ""custom_key"",`
		`""value"": ""custom_value""`
		`},`
		`{`
		`""key"": ""Content-Type"",`
		`""value"": ""application/json""`
		`}`
		`]`

sBodyData	String	JSON-formatierte Zeichenkette. Muss ein Array aus Objekten mit jeweils einem Key und einem Value enthalten.
		Beispiel:
		`[`
		`{`
		`""key"": """",`
		`""value"": {""json_object_key"":""json_object_value""}`
		`}`
		`]`

		Dieses Beispiel verdeutlicht die Vorgehensweise wenn **Content-Type:application/json-Header** genutzt wird. Abweichende Content-Types benötigen mitunter abweichende Inhalte.

Rückgabewert:

Im Erfolgsfall: String (JSON-formatiert, enthält die Objekte "status" [Statuscode] und "response" [Antwort des Servers])

Im Fehlerfall: String (JSON-formatiert), mit folgendem Aufbau:

{
"status": -1; *// Interner / combit Fehler*
"error": {
"type": number; *// Kann momentan die Werte 0 (UNKNOWN), 1 (INVALID_ARGUMENT), 2 (JSON_ERROR) oder 3 (HTTP_ERROR) enthalten.*
"type_string": string; *// Entspricht dem Namen des Fehlers, welcher in 'type' enthalten ist.*
"error_details": string; *// Nur bei type == 2 vorhanden, enthält weitere Informationen wieso die JSON-Formatierung der Serverantwort fehlgeschlagen ist.*
"response": string; *// Nur bei type == 2 vorhanden, enthält die Serverantwort, allerdings OHNE die Zeichen, welche den aktuellen JSON_ERROR ausgelöst haben. Dient nur zur Information und sollte nicht zur weiteren Datenverarbeitung genutzt werden.*
}
}

HTTPPut

Beschreibung:

Sendet einen HTTP-Request mit einer PUT-Methode.

Hinweis: Im Unterordner Scripts\ v_JSON Ihrer combit CRM-Installation finden Sie Scripte, die das Handling mit JSON-Strings vereinfachen. Diese können beispielsweise per <!--#include-once file="..."--> eingebunden werden. Im Kapitel **HTTP-Requests ausführen** finden Sie ein Beispiel zur Vorgehensweise bei der Nutzung von HTTP-Requests.

Parameter:

Parametername	Typ	Beschreibung
sUrl	String	URL des HTTP-Servers.

sHeaderData	String	JSON-formatierte Zeichenkette. Muss ein Array aus Objekten mit jeweils einem Key und einem Value enthalten.
		Beispiel:
		`[`
		`{`
		`""key"": ""custom_key"",`
		`""value"": ""custom_value""`
		`},`
		`{`
		`""key"": ""Content-Type"",`
		`""value"": ""application/json""`
		`}`
		`]`

sBodyData	String	JSON-formatierte Zeichenkette. Muss ein Array aus Objekten mit jeweils einem Key und einem Value enthalten.
		Beispiel:
		`[`
		`{`
		`""key"": """",`
		`""value"":`
		`{""json_object_key"":""json_object_value""}`
		`}`
		`]`
		Dieses Beispiel verdeutlicht die Vorgehensweise wenn **Content-Type:application/json-Header** genutzt wird. Abweichende Content-Types benötigen mitunter abweichende Inhalte.

Rückgabewert:

Im Erfolgsfall: String (JSON-formatiert, enthält die Objekte "status" [Statuscode] und "response" [Antwort des Servers])

Im Fehlerfall: String (JSON-formatiert), mit folgendem Aufbau:

{
"status": -1; *// Interner / combit Fehler*
"error": {
"type": number; *// Kann momentan die Werte 0 (UNKNOWN), 1 (INVALID_ARGUMENT), 2 (JSON_ERROR) oder 3 (HTTP_ERROR) enthalten.*
"type_string": string; *// Entspricht dem Namen des Fehlers, welcher in 'type' enthalten ist.*
"error_details": string; *// Nur bei type == 2 vorhanden, enthält weitere Informationen wieso die JSON-Formatierung der Serverantwort fehlgeschlagen ist.*
"response": string; *// Nur bei type == 2 vorhanden, enthält die Serverantwort, allerdings OHNE die Zeichen, welche den aktuellen JSON_ERROR ausgelöst haben. Dient nur zur Information und sollte nicht zur weiteren Datenverarbeitung genutzt werden.*
}
}

InvokeMenu

Beschreibung:

Ruft einen Menüeintrag der Anwendung auf. Neben der ID des Menüeintrages wird angegeben, ob das Script solange warten soll, bis der Befehl abgearbeitet wurde (und evtl. Dialoge geschlossen wurden) oder ob das Script direkt weiterlaufen soll. Die Menü-IDs der Anwendung finden Sie im Kapitel **Menü-IDs**.

Hinweis: Es werden nur Menü-IDs von direkt sichtbaren Menüs unterstützt, d.h. Kontextmenüs können nicht verwendet werden. Sollte die Methode in einem asynchron ausgeführten Script ausgeführt werden, so ist der Rückgabewert immer True. Der Rückgabewert beschreibt, ob der Aufruf übermittelt werden konnte, nicht jedoch, ob in der aufzurufenden Funktion ggf. ein Problem festgestellt wurde.

Parameter:

Parametername	Typ	Beschreibung
MenuID	Long	Die ID des Menüeintrages.
Synchron	Bool	True: synchrone Ausführung False: asynchrone Ausführung

Rückgabewert:

Bool

Wert	Beschreibung
True	Befehl zum Aufrufen eines Menüeintrags wurde erfolgreich an combit CRM übermittelt.
False	Befehl zum Aufrufen eines Menüeintrags konnte nicht übermittelt werden. Dies kann z. B. der Fall sein, wenn der aufzurufende Menü-Befehl derzeit nicht zur Verfügung steht.

Beispiel VBScript:

```
Call cRM.InvokeMenu(57664, True) ' Aufrufen des Info-Dialogs (ID: 57664) des
combit CRM
```

Beispiel C#-Script:

```
cRM.InvokeMenu(57664, true); // Aufrufen des Info-Dialogs (ID: 57664) des combit
CRM
```

Login

Beschreibung:

Öffnet ein Projekt.

Parameter:

Parametername	Typ	Beschreibung
ProjectFilePath	String	Verzeichnispfad der Projektdatei. Übergibt man für das zu ladende Projekt leer, so wird zunächst versucht das bereits geladene Projekt zurückzugeben. Wurde Login aber unmittelbar nach CreateObject mit leer aufgerufen, wird versucht das zuletzt geladene Projekt zu öffnen.
UserName	String	Benutzername. Übergibt man diesen und das Passwort leer, wird Windows Authentifizierung versucht. Schlägt diese fehl, wird der Login-Dialog angezeigt.
Password	String	Passwort des Benutzers

Rückgabewert:

Project

Hinweis: Diese Methode sollte nur aufgerufen werden, wenn noch kein anderes Projekt geladen ist. Sollte bereits ein Projekt geladen sein wird eine Ausnahme (C#: InvalidOperationException) ausgelöst, sofern sich das geladene Projekt vom angegebenen Projekt unterscheidet. Sollte aber das angegebene Projekt das gleiche sein, wie das aktuell geladene Projekt, so wird das aktuelle Projekt verwendet und zurück geliefert. Hierbei werden jedoch der angegebene Benutzername und das Passwort ignoriert und der Benutzer des bereits geladenen Projektes verwendet.

Beispiel VBScript:

```
Dim sProjectPath : sProjectPath = "C:\Program Files (x86)\combit\combit
CRM\Solutions\Large\combit_Large.crm"
Dim sUserName : sUserName = "Administrator"
Dim sPassword : sPassword = ""
Dim oProject : Set oProject = cRM.Login(sProjectPath, sUserName, sPassword)
```

```
If (oProject Is Nothing) Then
    Call cRM.DialogMessageBox("Der Login am Projekt " & "" & sProjectPath & "" & "
konnte nicht erfolgreich durchgeführt werden.", "cRM.Login", vbOkOnly)
Else
    Set oProject = Nothing
End If
```

Beispiel C#-Script:

```
cRMApplication cRM = new cRMApplication(EApplicationStartType.GetActiveobject);

string projectPath = @"C:\Program Files (x86)\combit\combit
CRM\Solutions\Large\combit_Large.crm";
string userName = "Administrator";
string password = "";
Project project = cRM.Login(projectPath, userName, password);

if (project == null)
{
    cRM.DialogMessageBox("Der Login am projekt " + projectPath + " konnte nicht
erfolgreich durchgeführt werden.", "cRM.Login", 0);
}
else
{
    project.Dispose();
```

OAuthRedirectDialog

Beschreibung:

Öffnet einen Browserdialog, der die übergebene URL ansteuert. Diese Methode ist explizit für OAuth-Anmeldungen zu nutzen. Wenn sich der Benutzer im Dialog angemeldet hat, gibt diese Methode die URL zurück, zu der die aufgerufene Seite einen Redirect ausführen möchte. Typischerweise befindet sich in dieser Redirect-URL ein Token, welches für weitere API-Aufrufe genutzt werden kann (z. B. um sich bei dem übergebenen Server zu authentifizieren bzw. den Request im Namen des im Dialog angegebenen Nutzers durchzuführen).

Parameter:

Parametername	Typ	Beschreibung
sUrl	String	URL des HTTP-Servers.
sRedirectUrl	String	Redirect-URL des HTTP-Servers Der Parameter wird genutzt, um Redirects richtig zu erkennen, falls die normale URL und die Redirect -URL die gleiche Origin-Komponente besitzen. Ein Beispiel für gleiche Origin-Komponenten bei URL und Redirect-URL wäre: URL (1.Parameter): https://login.microsoftonline.com/{tenant}/oauth2/v2.0/authorize Redirect-URL (2.Parameter): https://login.microsoftonline.com/common/oauth2/nativeclient

Rückgabewert:

String

Beispiel VBScript:

```
Dim sToken : sToken =
cRM.OAuthRedirectDialog("https://login.microsoftonline.com/{tenant}/oauth2/v2.0/au
thorize", "https://login.microsoftonline.com/common/oauth2/nativeclient")
```

Beispiel C#-Script:

```
string token =
cRM.OAuthRedirectDialog("https://login.microsoftonline.com/{tenant}/oauth2/v2.0/au
thorize", "https://login.microsoftonline.com/common/oauth2/nativeclient");
```

PerformanceCounterCreate

Beschreibung:

Legt einen Messpunkt mit einem bestimmten Namen an oder setzt einen bereits existierenden Messpunkt mit dem übergebenen Namen auf 0 zurück. So können Performance-Messungen mit Hilfe des Debug-Tools von combit CRM durchgeführt werden.

Parameter:

Parametername	Typ	Beschreibung
sCounterName	String	Name des Messpunkts.

Beispiel VBScript:

```
Call cRM.PerformanceCounterCreate("Mailversand")
Call cRM.SendMail("E-Mail-Adresse", "Betreff der E-Mail", "Text der E-Mail", "",
1)
Call cRM.PerformanceCounterHit("Mailversand", "Mailversand abgeschlossen")
```

Beispiel C#-Script:

```
cRM.PerformanceCounterCreate("Mailversand");
cRM.SendMail("eMail-Adresse", "Betreff der E-Mail", "Text der E-Mail", "", 1);
cRM.PerformanceCounterHit("Mailversand", "Mailversand abgeschlossen");
```

PerformanceCounterHit

Beschreibung:

Gibt eine Messung der vergangenen Zeit in Millisekunden seit dem Aufruf von **PerformanceCounterCreate** für den angegebenen Namen eines Messpunkts im Debug-Tool von combit CRM aus. Der zweite Parameter ist optional und kann, wenn erforderlich, noch weitere relevante Informationen zur Messung ausgeben.

Parameter:

Parametername	Typ	Beschreibung
sCounterName	String	Name des Messpunkts.
sAdditionalInfo	String	(Optional) Weitere Informationen.

Beispiel VBScript:

```
Call cRM.PerformanceCounterCreate("Mailversand")
Call cRM.SendMail("E-Mail-Adresse", "Betreff der E-Mail", "Text der E-Mail", "",
1)
Call cRM.PerformanceCounterHit("Mailversand", "Mailversand abgeschlossen")
```

Beispiel C#-Script:

```
cRM.PerformanceCounterCreate("Mailversand");
cRM.SendMail("eMail-Adresse", "Betreff der E-Mail ", "Text der E-Mail ", "", 1);
cRM.PerformanceCounterHit("Mailversand", "Mailversand abgeschlossen");
```

SendMail

Beschreibung:

Sendet eine E-Mail über die MAPI-Schnittstelle, unabhängig von den **Record** oder **RecordSet** Objekten.

Parameter:

Parametername	Typ	Beschreibung
EmailAddress	String	E-Mail-Adresse(n) eines oder mehrerer Empfänger(s). Bei mehreren E-Mail-Adressen müssen diese durch Semikolon getrennt werden. Die Empfangsart pro E-Mail-Adresse kann über Präfixe bestimmt werden. Wenn kein Präfix angegeben ist, wird "TO:" angenommen, dieser kann aber optional auch angegeben werden. Für den Versand von E-Mail-Kopien können zusätzlich die Präfixe "CC:" und/oder "BCC:" verwendet werden; beachten Sie hierbei, dass jede E-Mail-Adresse einen eigenen Präfix benötigt. Das E-Mail-Protokoll kann über die Präfixe "MAPI:" (Voreinstellung), "XMAPI" oder "SMTP:" (Voreinstellung bei Versand über Workflow-Server) bestimmt werden. Pro E-Mail-Versand kann nur einer dieser Präfixe verwendet werden, dieser muss am Anfang stehen. Beispiel 1: TO:maier@combit.net;CC:mueller@combit.net;CC:schmidt@combit.net;BCC:fischer@combit.net; Beispiel 2: maier@combit.net;weber@combit.net; Beispiel 3: SMTP:maier@combit.net;weber@combit.net;
Subject	String	Betreff der E-Mail.
Contents	String	Text der E-Mail.
Files	String	Kann eine Liste von E-Mail-Anhängen beinhalten. Diese müssen dann per Semikolon separiert übergeben werden. z. B. C:\MyFiles\Report.pdf;C:\Info.doc
ShowDialog	Long	Anzeige eines Bestätigungsdialoges. 0 = Dialog wird nicht angezeigt. 1 = Dialog wird angezeigt.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call cRM.SendMail("info@relationship-
manager.net;CC:support@combit.net;BCC:webmaster@combit.net;TO:info@combit.net",
"Betreff der E-Mail", "Text der E-Mail", "", 1)
```

Beispiel C#-Script:

```
cRM.SendMail("info@relationship-
manager.net;CC:support@combit.net;BCC:webmaster@combit.net;TO:info@combit.net",
"Betreff der E-Mail", "Text der E-Mail", "", 1);
```

SetIniProfileString

Beschreibung:

Setzt einen Eintrag in einer INI-Datei.

Parameter:

Parametername	Typ	Beschreibung

sSection	String	Name der Sektion in der INI-Datei, in der der zu setzende Eintrag steht.
sProperty	String	Name des zu setzenden Eintrags.
sValue	String	Der zu setzende Wert.
sIniFile	String	Pfad zur INI-Datei.

Beispiel VBScript:

```
Call cRM.SetIniProfileString("Scripting",
"GDPRLogTriggerActivatedByEvent_combitLarge", "False", "global.ini")
```

Beispiel C#-Script:

```
cRM.SetIniProfileString("Scripting", "GDPRLogTriggerActivatedByEvent_combitLarge",
"False", "global.ini");
```

SetWaitDlgText

Beschreibung:

Hiermit kann der Text in einem per StartWaitDlg gestarteten Wartedialog nachträglich geändert werden.

Parameter:

Parametername	Typ	Beschreibung
sText	String	Der anzuzeigende Informationstext.

Beispiel VBScript:

```
' Durchlaufen von mehreren Firmen-Datensätzen zur Berechnung der Summe der
Umsatzziele. Für die Laufzeit der Berechnung wird ein Wartedialog aufgerufen,
dessen Inhalt für jeden Durchlauf aktualisiert wird (SetWaitDlgText) und eine
Abbruchbedingung (CheckAbortedWaitDlg) besitzt.

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Firmen").CreateRecordSet
Dim oRecord
Dim bSkipped : bSkipped = False
Dim i, nTurnover, nSumOfTurnover
Dim nRecordSetRecCount : nRecordSetRecCount = oRecordSet.RecCount

If (nRecordSetRecCount > 0) Then
    Call cRM.StartWaitDlg("Die Summe der Umsatzziele für alle Datensätze der
Firmen-Ansicht wird berechnet, bitte haben Sie noch etwas Geduld.", True, 0)

    Call oRecordSet.MoveFirst()

    For i = 1 To nRecordSetRecCount
        Call cRM.SetWaitDlgText("Die angeforderten Informationen werden berechnet,
bitte haben Sie noch etwas Geduld." & vbCrLf & vbCrLf & "Fortschritt: Datensatz "
& CStr(i) & " von insgesamt " & oRecordSet.RecCount & " Datensätzen wird derzeit
verarbeitet.")

        Set oRecord = oRecordSet.CurrentRecord
        nTurnover = oRecord.GetContentsValueByName("TurnoverTarget")

        If (IsNull(nTurnover) = False) Then
            nSumOfTurnover = nSumOfTurnover + nTurnover
        End If

        Set oRecord = Nothing

        If (cRM.CheckAbortedWaitDlg = True) Then
            bSkipped = True
            Exit For
        ElseIf (cRM.CheckAbortedWaitDlg = False) Then
            Call oRecordSet.MoveNext()
        End If
```

```
    Next

    Call cRM.EndWaitDlg()

    If (bSkipped = True) Then
        Call cRM.DialogMessageBox("Die Summe der Umsatzziele konnte nicht
vollständig berechnet werden. Die Summe vor dem Abbruch der Aktion betrug " &
CStr(nSumOfTurnover) & " EUR.", "Aktion abgebrochen", vbOkOnly)
    ElseIf (bSkipped = False) Then
        Call cRM.DialogMessageBox("Die Summe der Umsatzziele beträgt " &
CStr(nSumOfTurnover) & " EUR.", "Ergebnis der Berechnung", vbOkOnly)
    End If

Else
    Call cRM.DialogMessageBox("Es wurden keine Datensätze für die Berechnung der
Summe der Umsatzziele in der Firmen-Ansicht gefunden.", "Keine Datensätze
gefunden", vbOkOnly)
End If

Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
// Durchlaufen von mehreren Firmen-Datensätzen zur Berechnung der Summe der
Umsatzziele. Für die Laufzeit der Berechnung wird ein Wartedialog aufgerufen,
dessen Inhalt für jeden Durchlauf aktualisiert wird (SetWaitDlgText) und eine
Abbruchbedingung (CheckAbortedWaitDlg) besitzt.

RecordSet recordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Firmen").CreateRecordSet();
Record record;
bool skipped = false;
int sumOfTurnover = 0;

long recordSetRecCount = recordSet.RecCount;

if (recordSetRecCount > 0)
    cRM.StartWaitDlg("Die Summe der Umsatzziele für alle Datensätze der Firmen-
Ansicht wird berechnet, bitte haben Sie noch etwas Geduld.", true, 0);

recordSet.MoveFirst();

for (int i = 1; i <= recordSetRecCount; i++)
{
    cRM.SetWaitDlgText("Die angeforderten Informationen werden berechnet, bitte
haben Sie noch etwas Geduld." + "\r\n" + "\r\n" + "Fortschritt: Datensatz " +
i.ToString() + " von insgesamt " + recordSet.RecCount + " Datensätzen wird derzeit
verarbeitet.");

    record = recordSet.CurrentRecord;
    int turnOver = (int)record.GetContentsValueByName("TurnoverTarget");

    if (turnOver != 0)
        sumOfTurnover += turnOver;

    record.Dispose();

    if (cRM.CheckAbortedWaitDlg() == true)
    {
        skipped = true;
        break;
    }
    else if (cRM.CheckAbortedWaitDlg() == false)
    {
        recordSet.MoveNext();
    }
}

cRM.EndWaitDlg();

if (skipped == true)
```

```
    cRM.DialogMessageBox("Die Summe der Umsatzziele konnte nicht vollständig
berechnet werden. Die Summe vor dem Abbruch der Aktion betrug " +
sumOfTurnover.ToString() + " EUR.", "Aktion abgebrochen", 0);
else if (skipped == false)
    cRM.DialogMessageBox("Die Summe der Umsatzziele beträgt " +
sumOfTurnover.ToString() + " EUR.", "Ergebnis der Berechnung", 0);
else
    cRM.DialogMessageBox("Es wurden keine Datensätze für die Berechnung der Summe
der Umsatzziele in der Firmen-Ansicht gefunden.", "Keine Datensätze gefunden", 0);

recordSet.Dispose();
```

StartWaitDlg

Beschreibung:

Zeigt einen Wartedialog mit einer Fortschrittsanimation sowie einem zu übergebenden Informationstext an. Die Ausführung des Scripts wird während der Anzeige fortgeführt. Die Ausblendung erfolgt mit **EndWaitDlg**. Über den zweiten Parameter kann bestimmt werden, ob der Dialog eine "Abbrechen"-Schaltfläche enthalten soll. Wenn ja, dann kann mit **CheckAbortedWaitDlg** geprüft werden, ob die Schaltfläche betätigt wurde. Reserviert: Über den dritten Parameter kann die Verzögerung bis zur Anzeige des Dialogs eingestellt werden. Standardmäßig wird eine Verzögerung von 3 Sekunden verwendet, um zu verhindern, dass schnell geöffnete und geschlossene Dialoge flackern.

Parameter:

Parametername	Typ	Beschreibung
sText	String	Der anzuzeigende Informationstext.
bCancel	Bool	Anzeige einer Schaltfläche "Abbrechen".
nInitialDelay	Long	Reserviert, derzeit wird immer 0 verwendet. Optional. Verzögerung bis zur Anzeige des Dialogs in Sekunden. 0 = unmittelbare Anzeige. Der vorbelegte Wert beträgt 3 Sekunden.

Beispiel VBScript:

```
' Durchlaufen von mehreren Firmen-Datensätzen zur Berechnung der Summe der
Umsatzziele. Für die Laufzeit der Berechnung wird ein Wartedialog aufgerufen,
dessen Inhalt für jeden Durchlauf aktualisiert wird (SetWaitDlgText) und eine
Abbruchbedingung (CheckAbortedWaitDlg) besitzt.

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Firmen").CreateRecordSet
Dim oRecord
Dim bSkipped : bSkipped = False
Dim i, nTurnover, nSumOfTurnover
Dim nRecordSetRecCount : nRecordSetRecCount = oRecordSet.RecCount

If (nRecordSetRecCount > 0) Then
    Call cRM.StartWaitDlg("Die Summe der Umsatzziele für alle Datensätze der
Firmen-Ansicht wird berechnet, bitte haben Sie noch etwas Geduld.", True, 0)

    Call oRecordSet.MoveFirst()

    For i = 1 To nRecordSetRecCount
        Call cRM.SetWaitDlgText("Die angeforderten Informationen werden berechnet,
bitte haben Sie noch etwas Geduld." & vbCrLf & vbCrLf & "Fortschritt: Datensatz "
& CStr(i) & " von insgesamt " & oRecordSet.RecCount & " Datensätzen wird derzeit
verarbeitet.")

        Set oRecord = oRecordSet.CurrentRecord
        nTurnover = oRecord.GetContentsValueByName("TurnoverTarget")

        If (IsNull(nTurnover) = False) Then
            nSumOfTurnover = nSumOfTurnover + nTurnover
        End If

        Set oRecord = Nothing
```

```
        If (cRM.CheckAbortedWaitDlg = True) Then
            bSkipped = True
            Exit For
        ElseIf (cRM.CheckAbortedWaitDlg = False) Then
            Call oRecordSet.MoveNext()
        End If
    Next

    Call cRM.EndWaitDlg()

    If (bSkipped = True) Then
        Call cRM.DialogMessageBox("Die Summe der Umsatzziele konnte nicht
vollständig berechnet werden. Die Summe vor dem Abbruch der Aktion betrug " &
CStr(nSumOfTurnover) & " EUR.", "Aktion abgebrochen", vbOkOnly)
    ElseIf (bSkipped = False) Then
        Call cRM.DialogMessageBox("Die Summe der Umsatzziele beträgt " &
CStr(nSumOfTurnover) & " EUR.", "Ergebnis der Berechnung", vbOkOnly)
    End If

Else
    Call cRM.DialogMessageBox("Es wurden keine Datensätze für die Berechnung der
Summe der Umsatzziele in der Firmen-Ansicht gefunden.", "Keine Datensätze
gefunden", vbOkOnly)
End If

Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
// Durchlaufen von mehreren Firmen-Datensätzen zur Berechnung der Summe der
Umsatzziele. Für die Laufzeit der Berechnung wird ein Wartedialog aufgerufen,
dessen Inhalt für jeden Durchlauf aktualisiert wird (SetWaitDlgText) und eine
Abbruchbedingung (CheckAbortedWaitDlg) besitzt.

RecordSet recordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Firmen").CreateRecordSet();
Record record;
bool skipped = false;
int sumOfTurnover = 0;

long recordSetRecCount = recordSet.RecCount;

if (recordSetRecCount > 0)
    cRM.StartWaitDlg("Die Summe der Umsatzziele für alle Datensätze der Firmen-
Ansicht wird berechnet, bitte haben Sie noch etwas Geduld.", true, 0);

recordSet.MoveFirst();

for (int i = 1; i <= recordSetRecCount; i++)
{
    cRM.SetWaitDlgText("Die angeforderten Informationen werden berechnet, bitte
haben Sie noch etwas Geduld." + "\r\n" + "\r\n" + "Fortschritt: Datensatz " +
i.ToString() + " von insgesamt " + recordSet.RecCount + " Datensätzen wird derzeit
verarbeitet.");

    record = recordSet.CurrentRecord;
    int turnOver = (int)record.GetContentsValueByName("TurnoverTarget");

    if (turnOver != 0)
        sumOfTurnover += turnOver;

    record.Dispose();

    if (cRM.CheckAbortedWaitDlg() == true)
    {
        skipped = true;
        break;
    }
    else if (cRM.CheckAbortedWaitDlg() == false)
    {
```

```
            recordSet.MoveNext();
        }
    }

    cRM.EndWaitDlg();

    if (skipped == true)
        cRM.DialogMessageBox("Die Summe der Umsatzziele konnte nicht vollständig
    berechnet werden. Der Summe vor dem Abbruch der Aktion betrug " +
    sumOfTurnover.ToString() + " EUR.", "Aktion abgebrochen", 0);
    else if (skipped == false)
        cRM.DialogMessageBox("Die Summe der Umsatzziele beträgt " +
    sumOfTurnover.ToString() + " EUR.", "Ergebnis der Berechnung", 0);
    else
        cRM.DialogMessageBox("Es wurden keine Datensätze für die Berechnung der Summe
    der Umsatzziele in der Firmen-Ansicht gefunden.", "Keine Datensätze gefunden", 0);

    recordSet.Dispose();
```

StoreGlobalConfigFile

Beschreibung:

Speichert die globalen Konfigurationsdateien ohne Projekt-ID.

Parameter:

Parametername	Typ	Beschreibung
sFilePath	String	Dateiname unter dem die Datei in der cmbt_Files Tabelle in der System-Datenbank abgelegt werden soll.
sLocalFilePath	String	Dateiname der lokalen Datei, die verwendet werden soll.

Rückgabewert:

Bool

Wert	Beschreibung
True	Befehl wurde ausgeführt.
False	Befehl konnte nicht ausgeführt werden, dies deutet darauf hin, dass einer der Parameterwerte ungültig ist (Dateiname in cmbt_Files Tabelle kann nicht gefunden werden, Pfad für Speicherort ist ungültig, …).

Beispiel VBScript:

```
' Die Datei global.ini wird vom Dateisystem in die cmbt_Files-Tabelle gespeichert

Dim sConfigFileName : sConfigFileName = "global.ini"
Dim sLocalFilePath : sLocalFilePath = "C:\" & sConfigFileName
Dim bSuccess : bSuccess = cRM.StoreGlobalConfigFile(sConfigFileName,
sLocalFilePath)

If (bSuccess = True) Then
    Call cRM.DialogMessageBox("Die Datei """ & sConfigFileName & """ konnte
erfolgreich gespeichert werden.", "cRM.StoreGlobalConfigFile", vbOkOnly)
ElseIf (bSuccess = False) Then
    Call cRM.DialogMessageBox("Die Datei """ & sConfigFileName & """ konnte nicht
erfolgreich gespeichert werden.", "cRM.StoreGlobalConfigFile", vbOkOnly)
End If
```

Beispiel C#-Script:

```
// Die Datei global.ini wird vom Dateisystem in die cmbt_Files-Tabelle gespeichert

string configFileName = "global.ini";
```

```
string localFilePath = @"C:\" + configFileName;
bool success = cRM.StoreGlobalConfigFile(configFileName, localFilePath);

if (success == true)
    cRM.DialogMessageBox("Die Datei \"" + configFileName + "\" konnte erfolgreich
gespeichert werden.", "cRM.StoreGlobalConfigFile", 0);
else if (success == false)
    cRM.DialogMessageBox("Die Datei \"" + configFileName + "\" konnte nicht
erfolgreich gespeichert werden.", "cRM.StoreGlobalConfigFile", 0);
```

3.4 Appointment Objekt

Beispiel VBScript:

```
Dim oProject : Set oProject = cRM.CurrentProject
Dim oActiveView : Set oActiveView = cRM.CurrentProject.ActiveViews.ActiveView
Dim oViewConfig : Set oViewConfig = oActiveView.Config
Dim oRecord : Set oRecord = oActiveView.CurrentRecordSet.CurrentRecord
Dim sCurrentUserLoginName : sCurrentUserLoginName =
oProject.Users.CurrentUser.LoginName
Dim sPrimaryKeyFieldName : sPrimaryKeyFieldName = oViewConfig.PrimaryKeyFldName
Dim sRecordRefDescription : sRecordRefDescription =
oRecord.GetRecordRefDescription
Dim sRefLink : sRefLink = oProject.ID & "|" & oActiveView.Name & "|" &
oViewConfig.FamilyName & "|" & sPrimaryKeyFieldName & "|" &
oRecord.GetContentsByName(sPrimaryKeyFieldName) & "|" & sRecordRefDescription
Set oRecord = Nothing
Set oViewConfig = Nothing
Set oActiveView = Nothing
Set oProject = Nothing

Dim oAppointment : Set oAppointment =
cRM.CurrentProject.timemanager.Appointments.Add()

oAppointment.ActionData = "info@relationship-manager.net"
oAppointment.ActionID = "TMMAIL"
oAppointment.ActionType = "1"
oAppointment.AllDayEvent = False
oAppointment.Attendees.Add("LFrisch")
oAppointment.Attendees.Add("THeld")
oAppointment.Body("{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1031{\fonttbl{\
f0\fnil\fcharset0 Calibri;}}{\*\generator Riched20 10.0.22000}\viewkind4\uc1
\pard\sa200\sl276\slmult1\i\f0\fs22\lang7 Formatierter \b\i0 Inhalt/Text \b0 des
\ul Termins\ulnone\par}")
oAppointment.Categories.Add("Meeting")
oAppointment.ChangeDate(Now())
oAppointment.ChangeUser(sCurrentUserLoginName)
oAppointment.Contact("Kontakt, mit dem der Termin stattfindet")
oAppointment.CreationDate(Now())
oAppointment.CreationUser(sCurrentUserLoginName)
oAppointment.End(DateAdd("h", 2, Now()))
oAppointment.ExtUserData1 = "Erster Abschnitt von Zusatzinformationen"
oAppointment.ExtUserData2 = "Zweiter Abschnitt von Zusatzinformationen"
oAppointment.HostDataBase = sRefLink
oAppointment.Importance = 3
oAppointment.IsRecurring = True
oAppointment.Location = "Büro"
oAppointment.Private = False
oAppointment.Reminder = True
oAppointment.ReminderMinutesBeforeStart = 30
oAppointment.Start(DateAdd("h", 1, Now()))
oAppointment.Subject = "Priorität" & oAppointment.Importance & ": Meeting im Büro
mit " & sRecordRefDescription
oAppointment.TimeStamp = FormatDateTime(Now(), vbGeneralDate)
oAppointment.User = sCurrentUserLoginName

Call oAppointment.Save()
Call oAppointment.Display()

Set oAppointment = Nothing
```

Beispiel C#-Script:

```
Project currentProject = cRM.CurrentProject;
View activeView = currentProject.ActiveViews.ActiveView;
ViewConfig activeViewConfig = activeView.Config;
Record currentRecord = activeView.CurrentRecordSet.CurrentRecord;
string currenUserLoginName = currentProject.Users.CurrentUser.LoginName;
string primaryKeyFieldName = activeViewConfig.PrimaryKeyFldName;
string recordRefDescription = currentRecord.GetRecordRefDescription();
string refLink = currentProject.ID + "|" + activeView.Name + "|" +
activeViewConfig.FamilyName + "|" + primaryKeyFieldName + "|" +
currentRecord.GetContentsByName(primaryKeyFieldName) + "|" + recordRefDescription;
currentRecord.Dispose();
activeViewConfig.Dispose();
activeView.Dispose();
currentProject.Dispose();

Appointment appointment = currentProject.TimeManager.Appointments.Add();

appointment.ActionData = "info@relationship-manager.net";
appointment.ActionID = "TMMAIL";
appointment.ActionType = 1;
appointment.AllDayEvent = false;
appointment.Attendees.Add("LFrisch");
appointment.Attendees.Add("THeld");
appointment.Body =
@"{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1031{\fonttbl{\f0\fnil\fcharset0
Calibri;}}{\*\generator Riched20 10.0.22000}\viewkind4\uc1
\pard\sa200\sl276\slmult1\i\f0\fs22\lang7 Formatierter \b\i0 Inhalt/Text \b0 des
\ul Termins\ulnone\par}";
appointment.Categories.Add("Meeting");
appointment.ChangeDate = System.DateTime.Now.ToString();
appointment.ChangeUser = currenUserLoginName;
appointment.Contact = "Kontakt, mit dem der Termin stattfindet";

appointment.End = System.DateTime.Now.AddHours(2);
appointment.ExtUserData1 = "Erster Abschnitt von Zusatzinformationen";
appointment.ExtUserData2 = "Zweiter Abschnitt von Zusatzinformationen";
appointment.HostDatabase = refLink;
appointment.Importance = 3;
appointment.Location = "Büro";
appointment.Private = false;
appointment.Reminder = true;
appointment.ReminderMinutesBeforeStart = 30;
appointment.Start = System.DateTime.Now.AddHours(1);
appointment.Subject = "Priorität " + appointment.Importance + ": Meeting im Büro
mit " + recordRefDescription;
appointment.TimeStamp = System.DateTime.Now.ToString();
appointment.User = currenUserLoginName;

RecurrencePattern recurrencePattern = appointment.RecurrencePattern;
recurrencePattern.RecurrenceType = 2; // Monatliche Wiederholung;
recurrencePattern.DayOfMonth = 10;
recurrencePattern.Duration = 60;
recurrencePattern.EndTime = System.DateTime.Now.AddMinutes(30);
recurrencePattern.Interval = 1;
recurrencePattern.NoEndDate = false;
recurrencePattern.StartTime = System.DateTime.Now.AddMinutes(10);
recurrencePattern.PatternStartDate = System.DateTime.Now.AddDays(14);
recurrencePattern.PatternEndDate = System.DateTime.Now.AddYears(1);
recurrencePattern.Dispose();

appointment.Links.NewLink().SetLinkFromString(refLink);
appointment.ClearRecurrencePattern();

appointment.Save();
appointment.Display();

appointment.Dispose();
```

3.4.1 Eigenschaften

ActionData

Beschreibung:

Enthält die Daten einer Aktion.

Typ:

String

Siehe auch:

ActionID

Beispiel VBScript:

```
oAppointment.ActionData = info@relationship-manager.net
```

Beispiel C#-Script:

```
appointment.ActionData = "info@relationship-manager.net";
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

ActionID

Beschreibung:

Eindeutige ID der zugeordneten Aktion.

Typ:

String

Wert	Beschreibung
<leer>	< keine Aktion ausgewählt >
TMMAIL	Senden einer E-Mail-Benachrichtigung
TMSHEX	Datei/Dokument ausführen bzw. öffnen
DIAL_CRM	Telefonanruf ausführen

Beispiel VBScript:

```
oAppointment.ActionID = "TMMAIL"
```

Beispiel C#-Script:

```
appointment.ActionID = "TMMAIL";
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

ActionType

Beschreibung:

Art der Ausführung der Aktion.

Typ:

String

Wert	Beschreibung
0	Automatisch
1	Manuell in Erinnerungsdialog

Beispiel VBScript:

```
oAppointment.ActionType = "1"
```

Beispiel C#-Script:

```
appointment.ActionType = 1;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

AllDayEvent

Beschreibung:

Legt fest, ob es sich um einen ganztägigen Termin handelt.

Typ:

Bool

Beispiel VBScript:

```
oAppointment.AllDayEvent = False
```

Beispiel C#-Script:

```
appointment.AllDayEvent = false;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Attendees

Beschreibung:

Liefert alle Teilnehmer des Termins als Objekt vom Typ **Attendees** zurück.

> **Hinweis**: Das Hinzufügen von Teilnehmern ist mit der Add-Methode des (zurückgelieferten) Attendees-Objekts möglich.

Typ:

Attendees

Beispiel VBScript:

```
oAppointment.Attendees.Add("LFrisch")
oAppointment.Attendees.Add("THeld")
```

Beispiel C#-Script:

```
appointment.Attendees.Add("LFrisch");
appointment.Attendees.Add("THeld");
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Body

Beschreibung:

Inhalt/Text des Termins (mit RTF-Formatierung).

Typ:

String

Beispiel VBScript:

```
oAppointment.Body =
"{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1031{\fonttbl{\f0\fnil\fcharset0
Calibri;}}{\*\generator Riched20 10.0.22000}\viewkind4\uc1
\pard\sa200\sl276\slmult1\i\f0\fs22\lang7 Formatierter \b\i0 Inhalt/Text \b0 des
\ul Termins\ulnone\par}"
```

Beispiel C#-Script:

```
appointment.Body =
@"{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1031{\fonttbl{\f0\fnil\fcharset0
Calibri;}}{\*\generator Riched20 10.0.22000}\viewkind4\uc1
\pard\sa200\sl276\slmult1\i\f0\fs22\lang7 Formatierter \b\i0 Inhalt/Text \b0 des
\ul Termins\ulnone\par}";
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

BodyPlain

Beschreibung:

Inhalt/Text des Termins (ohne RTF-Formatierung).

Typ:

String

Beispiel VBScript:

```
oAppointment.BodyPlain = "Inhalt/Text des Termins ohne RTF-Formatierung"
```

Beispiel C#-Script:

```
appointment.BodyPlain = "Inhalt/Text des Termins ohne RTF-Formatierung";
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Categories, read-only

Beschreibung:

Liefert alle Kategorien des Termins als Objekt vom Typ **Categories** zurück.

Typ:

Categories

Beispiel VBScript:

```
oAppointment.Categories.Add("Meeting")
```

Beispiel C#-Script:

```
appointment.Categories.Add("Meeting");
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

ChangeDate

Beschreibung:

Letztes Änderungsdatum des Termins.

Typ:

Date

Beispiel VBScript:

```
oAppointment.ChangeDate = Now()
```

Beispiel C#-Script:

```
appointment.ChangeDate = System.DateTime.Now.ToString();
```

> Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel Appointment Objekt.

ChangeUser

Beschreibung:

Letzter Änderungsbenutzer des Termins.

Typ:

String

Beispiel VBScript:

```
oAppointment.ChangeUser = sCurrentUserLoginName
```

Beispiel C#-Script:

```
appointment.ChangeUser = currenUserLoginName;
```

> Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel Appointment Objekt.

Contact

Beschreibung:

Kontakt, mit dem der Termin stattfindet ("Mit").

Typ:

String

Beispiel VBScript:

```
oAppointment.Contact = "Kontakt, mit dem der Termin stattfindet"
```

Beispiel C#-Script:

```
appointment.Contact = "Kontakt, mit dem der Termin stattfindet";
```

> Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel Appointment Objekt.

CreationDate

Beschreibung:

Erfassungsdatum des Termins.

Typ:

Date

Beispiel VBScript:

```
oAppointment.CreationDate = Now()
```

Beispiel C#-Script:

```
appointment.CreationDate = System.DateTime.Now.ToString();
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

CreationUser

Beschreibung:

Erfassungsbenutzer des Termins.

Typ:

String

Beispiel VBScript:

```
oAppointment.CreationUser = sCurrentUserLoginName
```

Beispiel C#-Script:

```
appointment.CreationUser = currenUserLoginName;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

End

Beschreibung:

Enddatum und –zeit des Termins.

Typ:

Date

Beispiel VBScript:

```
oAppointment.End = DateAdd("h", 2, Now())
```

Beispiel C#-Script:

```
appointment.End = System.DateTime.Now.AddHours(2);
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

ExtUserData1

Beschreibung:

Ermöglicht das Hinterlegen von Zusatzinformationen in Form von Zeichenketten (Strings).

> **Hinweis**: Die Zeichenketten können maximal jeweils 250 Zeichen lang werden.

Typ:

String

Beispiel VBScript:

```
oAppointment.ExtUserData1 = "Erster Abschnitt von Zusatzinformationen"
```

Beispiel C#-Script:

```
appointment.ExtUserData1 = "Erster Abschnitt von Zusatzinformationen";
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

ExtUserData2

Beschreibung:

Ermöglicht das Hinterlegen von Zusatzinformationen in Form von Zeichenketten (Strings).

> **Hinweis**: Die Zeichenketten können maximal jeweils 250 Zeichen lang werden.

Typ:

String

Beispiel VBScript:

```
oAppointment.ExtUserData2 = "Zweiter Abschnitt von Zusatzinformationen"
```

Beispiel C#-Script:

```
appointment.ExtUserData2 = "Zweiter Abschnitt von Zusatzinformationen";
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

HostDatabase

Beschreibung:

Eindeutige Projekt-ID. Der Aufbau ist folgender:

<ProjektID>|<Ansichtenname>|<Ansichtenfamilienname(optional)>|<Primärschlüsselfeld-name>|<Primärschlüsselwert>|<Bezeichnung Datensatzverknüpfung>

Typ:

String

Beispiel VBScript:

```
Dim oProject : Set oProject = cRM.CurrentProject
Dim oActiveView : Set oActiveView = cRM.CurrentProject.ActiveViews.ActiveView
Dim oViewConfig : Set oViewConfig = oActiveView.Config
Dim oRecord : Set oRecord = oActiveView.CurrentRecordSet.CurrentRecord
Dim sPrimaryKeyFieldName : sPrimaryKeyFieldName = oViewConfig.PrimaryKeyFldName
Dim sRecordRefDescription : sRecordRefDescription =
oRecord.GetRecordRefDescription
Dim sRefLink : sRefLink = oProject.ID & "|" & oActiveView.Name & "|" &
oViewConfig.FamilyName & "|" & sPrimaryKeyFieldName & "|" &
oRecord.GetContentsByName(sPrimaryKeyFieldName) & "|" & sRecordRefDescription
Set oRecord = Nothing
Set oViewConfig = Nothing
Set oActiveView = Nothing
Set oProject = Nothing

...

oAppointment.HostDataBase = sRefLink
```

Beispiel C#-Script:

```
Project currentProject = cRM.CurrentProject;
View activeView = currentProject.ActiveViews.ActiveView;
ViewConfig activeViewConfig = activeView.Config;
Record currentRecord = activeView.CurrentRecordSet.CurrentRecord;
string primaryKeyFieldName = activeViewConfig.PrimaryKeyFldName;
string recordRefDescription = currentRecord.GetRecordRefDescription();
string refLink = currentProject.ID + "|" + activeView.Name + "|" +
activeViewConfig.FamilyName + "|" + primaryKeyFieldName + "|" +
currentRecord.GetContentsByName(primaryKeyFieldName) + "|" + recordRefDescription;
currentRecord.Dispose();
activeViewConfig.Dispose();
activeView.Dispose();
currentProject.Dispose();

…

appointment.HostDatabase = refLink;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel Appointment Objekt.

Importance

Beschreibung:

Diese Eigenschaft ermöglicht es die Priorität eines Termins zu setzen und abzufragen.

Typ:

Long

Beispiel VBScript:

```
oAppointment.Importance = 3
```

Beispiel C#-Script:

```
appointment.Importance = 3;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel Appointment Objekt.

IsRecurring, read-only

Beschreibung:

Gibt an, ob es sich um einen Serientermin handelt.

Typ:

Bool

Beispiel VBScript:

```
oAppointment.IsRecurring = True
```

Beispiel C#-Script:

```
appointment.IsRecurring = true;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel Appointment Objekt.

Links, read-only

Beschreibung:

Liefert alle Verknüpfungen des Termins als Objekt vom Typ **Links** zurück.

Typ:

Links

Beispiel VBScript:

```
oAppointment.Links.NewLink.SetLinkFromString(sReflink)
```

Beispiel C#-Script:

```
appointment.Links.NewLink.SetLinkFromString(sReflink);
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Location

Beschreibung:

Name des Ortes, an dem der Termin stattfindet.

Typ:

String

Beispiel VBScript:

```
oAppointment.Location = "Büro"
```

Beispiel C#-Script:

```
appointment.Location = "Büro";
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Private

Beschreibung:

Legt fest, ob der Termin als "Privat" markiert werden soll.

Typ:

Bool

Beispiel VBScript:

```
oAppointment.Private = False
```

Beispiel C#-Script:

```
appointment.Private = false;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

RecurrencePattern

Beschreibung:

Eigenschaften und Einstellungen eines Serientermins. Gibt ein Objekt vom Typ **RecurrencePattern** zurück.

Typ:

RecurrencePattern

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
Set oRecurrencePattern = Nothing
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Reminder

Beschreibung:

Legt fest, ob an diesen Termin erinnert werden soll.

Typ:

Bool

Beispiel VBScript:

```
oAppointment.Reminder = True
```

Beispiel C#-Script:

```
appointment.Reminder = true;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

ReminderMinutesBeforeStart

Beschreibung:

Gibt die Anzahl der Minuten zurück, die vor der Fälligkeit an einen Termin erinnert werden soll.

Typ:

Long

Beispiel VBScript:

```
oAppointment.ReminderMinutesBeforeStart = 30
```

Beispiel C#-Script:

```
appointment.ReminderMinutesBeforeStart = 30;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Start

Beschreibung:

Startdatum und –zeit des Termins.

Typ:

Date

Beispiel VBScript:

```
oAppointment.Start = DateAdd("h", 1, Now())
```

Beispiel C#-Script:

```
appointment.Start = System.DateTime.Now.AddHours(1);
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Subject

Beschreibung:

Beschreibung des Termins.

Typ:

String

Beispiel VBScript:

```
oAppointment.Subject = "Priorität" & oAppointment.Importance & ": Meeting im Büro
mit " & sRecordRefDescription
```

Beispiel C#-Script:

```
appointment.Subject = "Priorität " + appointment.Importance + ": Meeting im Büro
mit " + recordRefDescription;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

TimeStamp

Beschreibung:

Zeichenkette die aus dem Änderungsdatum und einem fortlaufender Zahlenwert (als String) besteht. Der Zahlenwert wird bei jeder Datensatzänderung verändert. Durch diesen Wert kann man auch feststellen, wenn ein Datensatz sich an einem Tag mehrfach geändert hat.

Format:YYYYMMDD:XXXX, z. B. 20010219:0020

Typ:

String

Beispiel VBScript:

```
oAppointment.TimeStamp = FormatDateTime(Now(), vbGeneralDate)
```

Beispiel C#-Script:

```
appointment.TimeStamp = System.DateTime.Now.ToString();
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

UniqueID, read-only

Beschreibung:

Eindeutige RecordID des Termins.

Typ:

String

Beispiel VBScript:

```
Dim sUniqueID : sUniqueID = oAppointment.UniqueID
```

Beispiel C#-Script:

```
string uniqueID = appointment.UniqueID;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

User

Beschreibung:

Eigentümer des Termins.

Typ:

String

Beispiel VBScript:

```
oAppointment.User = sCurrentUserLoginName
```

Beispiel C#-Script:

```
appointment.User = currenUserLoginName;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

3.4.2 Methoden

ClearRecurrencePattern

Beschreibung:

Kennzeichnet einen Termin als Einzeltermin und setzt **IsRecurring** auf False.

Beispiel VBScript:

```
Call oAppointment.ClearRecurrencePattern()
```

Beispiel C#-Script:

```
appointment.ClearRecurrencePattern();
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Display

Beschreibung:

Zeigt einen Dialog zum Termin an.

Parameter:

Parametername	Typ	Beschreibung
bWait	Bool	True: Das Script wartet auf die Beendigung des Dialoges. False: Das Script läuft wartet nicht auf die Beendigung des Dialoges.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call oAppointment.Display()
```

Beispiel C#-Script:

```
appointment.Display();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

ExportAsICal

Beschreibung:

Exportiert den Termin in das iCal-Format.

Parameter:

Parametername	Typ	Beschreibung
sFileName	String	Pfad und Dateiname der zu exportierenden Datei.

Rückgabewert:

Bool

Wert	Beschreibung
True	Datei wurde exportiert.
False	Beim Exportieren der Datei ist ein Fehler aufgetreten.

Beispiel VBScript:

```
Call oAppointment.ExportAsICal("C:\" & sRecordRefDescription & ".ical")
```

Beispiel C#-Script:

```
appointment.ExportAsiCal(@"C:\" + recordRefDescription + ".ical");
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Remove

Beschreibung:

Löscht den Termin.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call oAppointment.Remove()
```

Beispiel C#-Script:

```
appointment.Remove();
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

Save

Beschreibung:

Speichert die Einstellungen des Termins.

Rückgabewert:

Bool

> **Wichtig**: Um den neu angelegten Termin in der Ansicht gleich angezeigt zu bekommen, muss UpdateViews nach der Speicherung aufgerufen werden.

Beispiel VBScript:

```
Call oAppointment.Save()
```

Beispiel C#-Script:

```
appointment.Save();
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **Appointment Objekt**.

3.5 Appointments Objekt

3.5.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der Einträge in der Sammlung zurück.

Typ:

Long

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Es befinden sich derzeit " &
CStr(cRM.CurrentProject.timemanager.Appointments.Count) & " Termine in der Termin-
und Aufgabenplanung des combit CRM.", "Appointments.Count", vbOKOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Es befinden sich derzeit " +
cRM.CurrentProject.TimeManager.Appointments.Count.ToString() + " Termine in der
Termin- und Aufgabenplanung des combit CRM.", "Appointments.Count", 0);
```

3.5.2 Methoden

Add

Beschreibung:

Legt einen neuen Termin an und liefert diesen als Objekt vom Typ **Appointment** zurück.

Rückgabewert:

Appointment

Beispiel VBScript:

```
Dim oAppointment : Set oAppointment =
cRM.CurrentProject.timemanager.Appointments.Add()
```

Beispiel C#-Script:

```
Appointment appointment = cRM.CurrentProject.TimeManager.Appointments.Add();
```

Item

Beschreibung:

Gibt einen Termin zurück. Es muss die Index-Nummer des Termins übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Appointment

Beispiel VBScript:

```
' Es wird überprüft wie viele ganztägige Termine (im Vergleich zur Gesamtanzahl an
Terminen) vorhanden sind

Dim nCount : nCount = 0
Dim nCountAllAppointments : nCountAllAppointments =
cRM.CurrentProject.timemanager.Appointments.Count
Dim nCountAllDayEvents : nCountAllDayEvents = 0
Dim oAppointment

For nCount = 1 To nCountAllAppointments

  Set oAppointment = cRM.CurrentProject.timemanager.Appointments.Item(nCount)

  If (oAppointment.AllDayEvent = True) Then
    nCountAllDayEvents = nCountAllDayEvents + 1
  End If

  Set oAppointment = Nothing

Next

Call cRM.DialogMessageBox("Bei der Gesamtanzahl von " &
CStr(nCountAllAppointments) & " Terminen, gibt es " & CStr(nCountAllDayEvents) & "
ganztägige Termine.", "Appointments.Item", vbOkonly)
```

Beispiel C#-Script:

```
// Es wird überprüft wie viele ganztägige Termine (im Vergleich zur Gesamtanzahl
an Terminen) vorhanden sind
```

```
int count = 0;
long countAllAppointments = cRM.CurrentProject.TimeManager.Appointments.Count;
int countAllDayEvents = 0;

Appointment appointment;

for (count = 1; count <= countAllAppointments; count++)
{
    appointment = cRM.CurrentProject.TimeManager.Appointments.Item(count);

    if (appointment.AllDayEvent == true)
    {
        countAllDayEvents++;
    }

    appointment.Dispose();
}

cRM.DialogMessageBox("Bei der Gesamtanzahl von " + countAllAppointments.ToString()
+ " Terminen, gibt es " + countAllDayEvents + " ganztägige Termine.",
"Appointments.Item", 0);
```

ItemByUniqueID

Beschreibung:

Gibt einen Termin anhand der eindeutigen RecordID zurück.

Parameter:

Parametername	Typ	Beschreibung
UniqueID	String	Eindeutige RecordID des Termins.

Rückgabewert:

Appointment
NULL (wenn der Benutzer kein Recht hat den Termin zu sehen).

Beispiel VBScript:

```
Dim oAppointment : Set oAppointment =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID("Eindeutige ID des
Termins")
```

Beispiel C#-Script:

```
Appointment appointment =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID("Eindeutige ID des
Termins");
```

Remove

Beschreibung:

Löscht einen Termin. Es muss die Index-Nummer des Termins übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oAppointments : Set oAppointments =
cRM.CurrentProject.timemanager.Appointments
Call oAppointments.Remove(nUniqueAppointmentID)
Set oAppointments = Nothing
```

Beispiel C#-Script:

```
Appointments appointments = cRM.CurrentProject.TimeManager.Appointments;
appointments.Remove(uniqueAppointmentID);
appointments.Dispose();
```

RemoveAll

Beschreibung:

Löscht alle Termine.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oAppointments : Set oAppointments =
cRM.CurrentProject.timemanager.Appointments
Call oAppointments.RemoveAll()
Set oAppointments = Nothing
```

Beispiel C#-Script:

```
Appointments appointments = cRM.CurrentProject.TimeManager.Appointments;
appointments.RemoveAll();
appointments.Dispose();
```

SetFilter

Beschreibung:

Filtert die Anzeige auf Termine, die bestimmte Eigenschaften erfüllen. Die Filterung kann ausgehend von Datenbank oder Datensatz der Host-Applikation erfolgen oder auf Basis des Benutzers.

Parameter:

Parametername	Typ	Beschreibung
FilterType	TMListFilter Constants	Art der Filterung
HostDatabase	String	Datenbank (Pfad+Name)
HostRecordID	Long	Eindeutige Datensatznummer des zugeordneten Datensatzes
UserName	String	Name des Benutzers

Konstanten der TMListFilterConstants:

Konstante	Wert	Beschreibung
TM_FILTER_HOSTDB	1	Filterung aller Termine zu einer bestimmten Datenbank, korrespondierend zu der Eigenschaft HostDatabase des Appointment-Objektes.
TM_FILTER_HOSTRECID	2	Filterung aller Termine zu einem bestimmten Datensatz, korrespondierend zu der Eigenschaft HostRecordID des Appointment-Objektes. Impliziert die Verwendung eines Filters auf Datenbank-Ebene, vgl. TM_FILTER_HOSTDB.

TM_FILTER_USER	4	Filterung aller Termine zu einem bestimmten Benutzer.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oAppointments : Set oAppointments =
cRM.CurrentProject.timemanager.Appointments
Call oAppointments.SetFilter(1, hostDatabase, hostRecordID, userName)
Set oAppointments = Nothing
```

Beispiel C#-Script:

```
Appointments appointments = cRM.CurrentProject.TimeManager.Appointments;
appointments.SetFilter(1, hostDatabase, hostRecordID, userName);
appointments.Dispose();
```

3.6 Attendee Objekt

3.6.1 Eigenschaften

Name, read-only

Beschreibung:

Liefert den Namen des Teilnehmers zurück.

Typ:

String

Beispiel VBScript:

```
' Zeigt die Teilnehmer eines bestimmten Termins an

Dim oAttendees : Set oAttendees =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).A
ttendees
Dim oAttendee
Dim nCount : nCount = 0
Dim nAttendeeType : nCount = 0
Dim sAttendeeType : sAttendeeType = ""

For nCount = 1 To oAttendees.Count

    Set oAttendee = oAttendees.Item(nCount)
    nAttendeeType = oAttendee.Type

    If (nAttendeeType = 0) Then
        sAttendeeType = "Benutzer"
    ElseIf (nAttendeeType = 1) Then
        sAttendeeType = "Gruppe"
    ElseIf (nAttendeeType = 2) Then
        sAttendeeType = "Ressource"
    ElseIf (nAttendeeType = -1) Then
        sAttendeeType = "Unbekannter Typ"
    End If

    Call cRM.DialogMessageBox("Teilnehmer " & CStr(nCount) & ": " & oAttendee.Name
& ", Typ: " & sAttendeeType, "Attendee.Name & Attendee.Type", vbOkOnly)

    Set oAttendee = Nothing

Next
```

```
Set oAttendees = Nothing
```

Beispiel C#-Script:

```
// Zeigt die Teilnehmer eines bestimmten Termins an

string uniqueAppointmentID = "Hier gewnschte Termin-ID eintragen";
Attendees attendees =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).At
tendees;
Attendee attendee;
long attendeeType;
string attendeeTypeTranslated = string.Empty;

for (int i = 1; i <= attendees.Count; i++)
{
    attendee = attendees.Item(i);
    attendeeType = attendee.Type;

    if (attendeeType == 0)
        attendeeTypeTranslated = "Benutzer";
    else if (attendeeType == 1)
        attendeeTypeTranslated = "Gruppe";
    else if (attendeeType == 2)
        attendeeTypeTranslated = "Ressource";
    else if (attendeeType == -1)
        attendeeTypeTranslated = "Unbekannter Typ";

    cRM.DialogMessageBox("Teilnehmer " + i.ToString() + ": " + attendee.Name + ",
Typ: " + attendeeTypeTranslated, "Attendee.Name & Attendee.Type", 0);
    attendee.Dispose();
}

attendees.Dispose();
```

Type, read-only

Beschreibung:

Liefert den Typ des Teilnehmers zurück.

Typ:

Long

Wert	Beschreibung
0	Benutzer.
1	Gruppe.
2	Ressource.
-1	Unbekannter Typ.

Beispiel VBScript:

```
' Zeigt die Teilnehmer eines bestimmten Termins an

Dim oAttendees : Set oAttendees =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).A
ttendees
Dim oAttendee
Dim nCount : nCount = 0
Dim nAttendeeType : nCount = 0
Dim sAttendeeType : sAttendeeType = ""

For nCount = 1 To oAttendees.Count

    Set oAttendee = oAttendees.Item(nCount)
```

```
        nAttendeeType = oAttendee.Type

        If (nAttendeeType = 0) Then
            sAttendeeType = "Benutzer"
        ElseIf (nAttendeeType = 1) Then
            sAttendeeType = "Gruppe"
        ElseIf (nAttendeeType = 2) Then
            sAttendeeType = "Ressource"
        ElseIf (nAttendeeType = -1) Then
            sAttendeeType = "Unbekannter Typ"
        End If

        Call cRM.DialogMessageBox("Teilnehmer " & CStr(nCount) & ": " & oAttendee.Name
    & ", Typ: " & sAttendeeType, "Attendee.Name & Attendee.Type", vbOkOnly)

        Set oAttendee = Nothing

    Next

    Set oAttendees = Nothing
```

Beispiel C#-Script:

```
    // Zeigt die Teilnehmer eines bestimmten Termins an

    string uniqueAppointmentID = "Hier gewnschte Termin-ID eintragen";
    Attendees attendees =
    cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).At
    tendees;
    Attendee attendee;
    long attendeeType;
    string attendeeTypeTranslated = string.Empty;

    for (int i = 1; i <= attendees.Count; i++)
    {
        attendee = attendees.Item(i);
        attendeeType = attendee.Type;

        if (attendeeType == 0)
            attendeeTypeTranslated = "Benutzer";
        else if (attendeeType == 1)
            attendeeTypeTranslated = "Gruppe";
        else if (attendeeType == 2)
            attendeeTypeTranslated = "Ressource";
        else if (attendeeType == -1)
            attendeeTypeTranslated = "Unbekannter Typ";

        cRM.DialogMessageBox("Teilnehmer " + i.ToString() + ": " + attendee.Name + ",
    Typ: " + attendeeTypeTranslated, "Attendee.Name & Attendee.Type", 0);
        attendee.Dispose();
    }

    attendees.Dispose();
```

3.7 Attendees Objekt

3.7.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der Einträge in der Sammlung zurück.

Typ:

Long

Beispiel VBScript:

```
Dim oAttendees : Set oAttendees =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).A
ttendees
Call cRM.DialogMessageBox("Der ausgewählte Termin besitzt " &
CStr(oAttendees.Count) & " Teilnehmer.", "Attendees.Count", vbOkOnly)
Set oAttendees = Nothing
```

Beispiel C#-Script:

```
string uniqueAppointmentID = "Hier gewünschte Termin-ID eintragen";
Attendees attendees =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).At
tendees;
cRM.DialogMessageBox("Der ausgewählte Termin besitzt " +
attendees.Count.ToString() + " Teilnehmer.", "Attendees.Count", 0);
attendees.Dispose();
```

3.7.2 Methoden

Add

Beschreibung:

Fügt einen neuen Teilnehmer zu einer Aufgabe/einem Termin hinzu.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Login Name des Benutzers/Name der Gruppe in der Benutzerverwaltung bzw. Name der Ressource.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim sLoginName : sLoginName = "Administrator"
Call
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).A
ttendees.Add(sLoginName)
```

Beispiel C#-Script:

```
string uniqueAppointmentID = "Hier gewünschte Termin-ID eintragen";
string loginName = "Administrator";
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).At
tendees.Add(loginName);
```

Item

Beschreibung:

Gibt einen Teilnehmer zurück. Es muss die Index-Nummer des Teilnehmers übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Attendee
NULL (wenn der Teilnehmer nicht existiert)

Beispiel VBScript:

```
' Zeigt die Teilnehmer eines bestimmten Termins an

Dim oAttendees : Set oAttendees =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).A
ttendees
Dim oAttendee
Dim nCount : nCount = 0

For nCount = 1 To oAttendees.Count

    Set oAttendee = oAttendees.Item(nCount)
    nAttendeeType = oAttendee.Type

    If (nAttendeeType = 0) Then
        sAttendeeType = "Benutzer"
    ElseIf (nAttendeeType = 1) Then
        sAttendeeType = "Gruppe"
    ElseIf (nAttendeeType = 2) Then
        sAttendeeType = "Ressource"
    ElseIf (nAttendeeType = -1) Then
        sAttendeeType = "Unbekannter Typ"
    End If

    Call cRM.DialogMessageBox("Teilnehmer " & CStr(nCount) & ": " & oAttendee.Name
& ", Typ: " & sAttendeeType, "Attendees.Item", vbOkOnly)

    Set oAttendee = Nothing

Next

Set oAttendees = Nothing
```

Beispiel C#-Script:

```
// Zeigt die Teilnehmer eines bestimmten Termins an

string uniqueAppointmentID = "Hier gewünschte Termin-ID eintragen";
Attendees attendees =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).At
tendees;
Attendee attendee;
long attendeeType;
string attendeeTypeTranslated = string.Empty;

for (int i = 1; i <= attendees.Count; i++)
{
    attendee = attendees.Item(i);
    attendeeType = attendee.Type;

    if (attendeeType == 0)
        attendeeTypeTranslated = "Benutzer";
```

```
        else if (attendeeType == 1)
            attendeeTypeTranslated = "Gruppe";
        else if (attendeeType == 2)
            attendeeTypeTranslated = "Ressource";
        else if (attendeeType == -1)
            attendeeTypeTranslated = "Unbekannter Typ";

        cRM.DialogMessageBox("Teilnehmer " + i.ToString() + ": " + attendee.Name + ",
Typ: " + attendeeTypeTranslated, "Attendee.Name & Attendee.Type", 0);
        attendee.Dispose();
    }

    attendees.Dispose();
```

ItemByName

Beschreibung:

Gibt einen Teilnehmer anhand seines Namens zurück.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Login Name des Benutzers/Name der Gruppe in der Benutzerverwaltung bzw. Name der Ressource.

Rückgabewert:

Attendee
NULL (wenn der Teilnehmer nicht existiert)

Beispiel VBScript:

```
' Zeigt den Typ eines Teilnehmers eines bestimmten Termins an

Dim sLoginName : sLoginName = "Administrator"
Dim oAttendees : Set oAttendees =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).A
ttendees
Dim oAttendee : Set oAttendee = oAttendees.ItemByName(sLoginName)

nAttendeeType = oAttendee.Type

If (nAttendeeType = 0) Then
    sAttendeeType = "Benutzer"
ElseIf (nAttendeeType = 1) Then
    sAttendeeType = "Gruppe"
ElseIf (nAttendeeType = 2) Then
    sAttendeeType = "Ressource"
ElseIf (nAttendeeType = -1) Then
    sAttendeeType = "Unbekannter Typ"
End If

Call cRM.DialogMessageBox("Teilnehmer " & oAttendee.Name & " besitzt folgenden
Typ: " & sAttendeeType, "Attendees.ItemByName", vbOkOnly)

Set oAttendee = Nothing
Set oAttendees = Nothing
```

Beispiel C#-Script:

```
// Zeigt den Typ eines Teilnehmers eines bestimmten Termins an

string uniqueAppointmentID = "Hier gewünschte Termin-ID eintragen";
string loginName = "Administrator";
```

```
Attendees attendees =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).At
tendees;
Attendee attendee = attendees.ItemByName(loginName);

long attendeeType = attendee.Type;
string attendeeTypeTranslated = string.Empty;

if (attendeeType == 0)
    attendeeTypeTranslated = "Benutzer";
else if (attendeeType == 1)
    attendeeTypeTranslated = "Gruppe";
else if (attendeeType == 2)
    attendeeTypeTranslated = "Ressource";
else if (attendeeType == -1)
    attendeeTypeTranslated = "Unbekannter Typ";

cRM.DialogMessageBox("Teilnehmer " + attendee.Name + " besitzt folgenden Typ: " +
attendeeTypeTranslated, "Attendees.ItemByName", 0)

attendee.Dispose();
attendees.Dispose();
```

Remove

Beschreibung:

Löscht einen Teilnehmer. Es muss die Index-Nummer des Teilnehmers übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Entfernt einen Teilnehmer von einem bestimmten Termin. Dabei werden alle
Teilnehmer durchlaufen und nach Prüfung des Namens entschieden, ob der Teilnehmer
entfernt werden soll

Dim sNameToRemoveAsAttendee : sNameToRemoveAsAttendee = "Administrator"
Dim oAttendees : Set oAttendees =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).A
ttendees
Dim nCount : nCount = 0

For nCount = 1 To oAttendees.Count

    Set oAttendee = oAttendees.Item(nCount)

    If (oAttendee.Name = sNameToRemoveAsAttendee) Then

        Call oAttendees.Remove(nCount)

    End If

    Set oAttendee = Nothing

Next

Set oAttendees = Nothing
```

Beispiel C#-Script:

```
// Entfernt einen Teilnehmer von einem bestimmten Termin. Dabei werden alle
Teilnehmer durchlaufen und nach Prüfung des Namens entschieden, ob der Teilnehmer
entfernt werden soll

string uniqueAppointmentID = "Hier gewnüschte Termin-ID eintragen";
string nameToRemoveAsAttendee = "Administrator";
Attendees attendees =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).At
tendees;
Attendee attendee;

for (int i = 1; i <= attendees.Count; i++)
{
    attendee = attendees.Item(i);

    if (attendee.Name == nameToRemoveAsAttendee)
    {
        attendees.Remove(i);
    }

    attendee.Dispose();
}

attendees.Dispose();
```

RemoveByName

Beschreibung:

Löscht einen Teilnehmer anhand seines Names.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Login Name des Benutzers/Name der Gruppe in der Benutzerverwaltung bzw. Name der Ressource.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim sNameToRemoveAsAttendee : sNameToRemoveAsAttendee = "Administrator"
Dim oAttendees : Set oAttendees =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).A
ttendees

Call oAttendees.RemoveByName(sNameToRemoveAsAttendee)

Set oAttendees = Nothing
```

Beispiel C#-Script:

```
string uniqueAppointmentID = "Hier gewünschte Termin-ID eintragen";
string nameToRemoveAsAttendee = "Administrator";
Attendees attendees =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).At
tendees;

attendees.RemoveByName(nameToRemoveAsAttendee);
attendees.Dispose();
```

3.8 CallItem Objekt

3.8.1 Eigenschaften

DialRetriesCount, read-only

Beschreibung:

Die Anzahl der bisherigen Wählversuche.

Typ:

Long

Beispiel VBScript:

```
' Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

Dim oCallList : Set oCallList = cRM.phonemanager.CallList
Dim oCallItem
Dim nCount : nCount = 0
Dim nDialRetriesCount : nDialRetriesCount = 0
Dim sFirstInfo : sFirstInfo = ""
Dim sInfo : sInfo = ""
Dim sLastInfo : sLastInfo = ""
Dim sNumber : sNumber = ""

For nCount = 1 To oCallList.Count

    Set oCallItem = oCallList.Item(nCount)

    If (oCallItem.DialRetriesCount > 10) Then
        Call oCallItem.Remove()
    Else
        nDialRetriesCount = oCallItem.DialRetriesCount
        sFirstInfo = oCallItem.FirstInfo
        sInfo = oCallItem.Info
        sLastInfo = oCallItem.LastInfo
        sNumber = oCallItem.Number

        If (cRM.DialogMessageBox("Die Nummer " & sNumber & " (" & sFirstInfo & " -
" & sInfo & " - " & sLastInfo & ") konnte mit " & CStr(nDialRetriesCount) & "
Anrufversuchen nicht erreicht werden." & vbCrlf & "Soll diese Rufnummer aus der
Anrufliste augetragen werden?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
            Call oCallItem.Remove()
        Else
            If (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehen Datensatz im combit CRM zu
springen?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
                Call oCallItem.GotoRecord()
            End If
        End If
    End If

    Set oCallItem = Nothing

Next

Set oCallList = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

CallList callList = cRM.PhoneManager.CallList;
CallItem callItem;
long dialRetriesCount;
string firstInfo;
string info;
string lastInfo;
string number;

for (int i = 1; i <= callList.Count; i++)
{
    callItem = callList.Item(i);

    if (callItem.DialRetriesCount > 10)
    {
        callItem.Remove();
    }
    else
    {
        dialRetriesCount = callItem.DialRetriesCount;
        firstInfo = callItem.FirstInfo;
        info = callItem.Info;
        lastInfo = callItem.LastInfo;
        number = callItem.Number;

       if (cRM.DialogMessageBox("Die Nummer " + number + "(" + firstInfo + " - " +
info + " - " + lastInfo + ") konnte mit " + dialRetriesCount.ToString() + "
Anrufversuchen nicht erreicht werden." + "\r\n" + "Soll diese Rufnummer aus der
Anrufliste ausgetragen werden?", "CallList.CallItem", 3) == 6) // Es wurde
"Ja/Yes" angeklickt
        {
            callItem.Remove();
        }
        else
        {
            if (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", 3) == 6)
            {
                callItem.GotoRecord();
            }
        }
    }

    callItem.Dispose();
}

callList.Dispose();
```

FirstInfo, read-only

Beschreibung:

Infotext (in der Regel Zeitpunkt) des ersten Wählversuches.

Typ:

String

Beispiel VBScript:

```
' Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen
```

```
Dim oCallList : Set oCallList = cRM.phonemanager.CallList
Dim oCallItem
Dim nCount : nCount = 0
Dim nDialRetriesCount : nDialRetriesCount = 0
Dim sFirstInfo : sFirstInfo = ""
Dim sInfo : sInfo = ""
Dim sLastInfo : sLastInfo = ""
Dim sNumber : sNumber = ""

For nCount = 1 To oCallList.Count

    Set oCallItem = oCallList.Item(nCount)

    If (oCallItem.DialRetriesCount > 10) Then
        Call oCallItem.Remove()
    Else
        nDialRetriesCount = oCallItem.DialRetriesCount
        sFirstInfo = oCallItem.FirstInfo
        sInfo = oCallItem.Info
        sLastInfo = oCallItem.LastInfo
        sNumber = oCallItem.Number

        If (cRM.DialogMessageBox("Die Nummer " & sNumber & " (" & sFirstInfo & " -
" & sInfo & " - " & sLastInfo & ") konnte mit " & CStr(nDialRetriesCount) & "
Anrufversuchen nicht erreicht werden." & vbCrlf & "Soll diese Rufnummer aus der
Anrufliste augetragen werden?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
            Call oCallItem.Remove()
        Else
            If (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehen Datensatz im combit CRM zu
springen?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
                Call oCallItem.GotoRecord()
            End If
        End If
    End If

    Set oCallItem = Nothing

Next

Set oCallList = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

CallList callList = cRM.PhoneManager.CallList;
CallItem callItem;
long dialRetriesCount;
string firstInfo;
string info;
string lastInfo;
string number;

for (int i = 1; i <= callList.Count; i++)
{
    callItem = callList.Item(i);

    if (callItem.DialRetriesCount > 10)
    {
        callItem.Remove();
    }
    else
    {
        dialRetriesCount = callItem.DialRetriesCount;
        firstInfo = callItem.FirstInfo;
        info = callItem.Info;
        lastInfo = callItem.LastInfo;
        number = callItem.Number;
```

```
        if (cRM.DialogMessageBox("Die Nummer " + number + "(" + firstInfo + " - " +
info + " - " + lastInfo + ") konnte mit " + dialRetriesCount.ToString() + "
Anrufversuchen nicht erreicht werden." + "\r\n" + "Soll diese Rufnummer aus der
Anrufliste ausgetragen werden?", "CallList.CallItem", 3) == 6) // Es wurde
"Ja/Yes" angeklickt
        {
            callItem.Remove();
        }
        else
        {
            if (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", 3) == 6)
            {
                callItem.GotoRecord();
            }
        }
    }

    callItem.Dispose();
}

callList.Dispose();
```

Info, read-only

Beschreibung:

Der zugeordnete Beschreibungstext (z. B. Zeitpunkt des eingehenden oder Beschreibung des verknüpften Datensatzes des ausgehenden Anrufes).

Typ:

String

Beispiel VBScript:

```
' Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

Dim oCallList : Set oCallList = cRM.phonemanager.CallList
Dim oCallItem
Dim nCount : nCount = 0
Dim nDialRetriesCount : nDialRetriesCount = 0
Dim sFirstInfo : sFirstInfo = ""
Dim sInfo : sInfo = ""
Dim sLastInfo : sLastInfo = ""
Dim sNumber : sNumber = ""

For nCount = 1 To oCallList.Count

    Set oCallItem = oCallList.Item(nCount)

    If (oCallItem.DialRetriesCount > 10) Then
        Call oCallItem.Remove()
    Else
        nDialRetriesCount = oCallItem.DialRetriesCount
        sFirstInfo = oCallItem.FirstInfo
        sInfo = oCallItem.Info
        sLastInfo = oCallItem.LastInfo
        sNumber = oCallItem.Number

        If (cRM.DialogMessageBox("Die Nummer " & sNumber & " (" & sFirstInfo & " -
" & sInfo & " - " & sLastInfo & ") konnte mit " & CStr(nDialRetriesCount) & "
Anrufversuchen nicht erreicht werden." & vbCrlf & "Soll diese Rufnummer aus der
Anrufliste augetragen werden?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
            Call oCallItem.Remove()
```

```
        Else
            If (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehen Datensatz im combit CRM zu
springen?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
                Call oCallItem.GotoRecord()
            End If
        End If
    End If

    Set oCallItem = Nothing

Next

Set oCallList = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

CallList callList = cRM.PhoneManager.CallList;
CallItem callItem;
long dialRetriesCount;
string firstInfo;
string info;
string lastInfo;
string number;

for (int i = 1; i <= callList.Count; i++)
{
    callItem = callList.Item(i);

    if (callItem.DialRetriesCount > 10)
    {
        callItem.Remove();
    }
    else
    {
        dialRetriesCount = callItem.DialRetriesCount;
        firstInfo = callItem.FirstInfo;
        info = callItem.Info;
        lastInfo = callItem.LastInfo;
        number = callItem.Number;

       if (cRM.DialogMessageBox("Die Nummer " + number + "(" + firstInfo + " - " +
info + " - " + lastInfo + ") konnte mit " + dialRetriesCount.ToString() + "
Anrufversuchen nicht erreicht werden." + "\r\n" + "Soll diese Rufnummer aus der
Anrufliste ausgetragen werden?", "CallList.CallItem", 3) == 6) // Es wurde
"Ja/Yes" angeklickt
        {
            callItem.Remove();
        }
        else
        {
            if (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", 3) == 6)
            {
                callItem.GotoRecord();
            }
        }
    }

    callItem.Dispose();
}

callList.Dispose();
```

LastInfo, read-only

Beschreibung:

Infotext (in der Regel Zeitpunkt) des letzten Wählversuches.

Typ:

String

Beispiel VBScript:

```
' Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

Dim oCallList : Set oCallList = cRM.phonemanager.CallList
Dim oCallItem
Dim nCount : nCount = 0
Dim nDialRetriesCount : nDialRetriesCount = 0
Dim sFirstInfo : sFirstInfo = ""
Dim sInfo : sInfo = ""
Dim sLastInfo : sLastInfo = ""
Dim sNumber : sNumber = ""

For nCount = 1 To oCallList.Count

    Set oCallItem = oCallList.Item(nCount)

    If (oCallItem.DialRetriesCount > 10) Then
        Call oCallItem.Remove()
    Else
        nDialRetriesCount = oCallItem.DialRetriesCount
        sFirstInfo = oCallItem.FirstInfo
        sInfo = oCallItem.Info
        sLastInfo = oCallItem.LastInfo
        sNumber = oCallItem.Number

        If (cRM.DialogMessageBox("Die Nummer " & sNumber & " (" & sFirstInfo & " -
" & sInfo & " - " & sLastInfo & ") konnte mit " & CStr(nDialRetriesCount) & "
Anrufversuchen nicht erreicht werden." & vbCrlf & "Soll diese Rufnummer aus der
Anrufliste augetragen werden?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
            Call oCallItem.Remove()
        Else
            If (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehen Datensatz im combit CRM zu
springen?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
                Call oCallItem.GotoRecord()
            End If
        End If
    End If

    Set oCallItem = Nothing

Next

Set oCallList = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

CallList callList = cRM.PhoneManager.CallList;
CallItem callItem;
long dialRetriesCount;
string firstInfo;
string info;
string lastInfo;
string number;
```

```
for (int i = 1; i <= callList.Count; i++)
{
    callItem = callList.Item(i);

    if (callItem.DialRetriesCount > 10)
    {
        callItem.Remove();
    }
    else
    {
        dialRetriesCount = callItem.DialRetriesCount;
        firstInfo = callItem.FirstInfo;
        info = callItem.Info;
        lastInfo = callItem.LastInfo;
        number = callItem.Number;

       if (cRM.DialogMessageBox("Die Nummer " + number + "(" + firstInfo + " - " +
info + " - " + lastInfo + ") konnte mit " + dialRetriesCount.ToString() + "
Anrufversuchen nicht erreicht werden." + "\r\n" + "Soll diese Rufnummer aus der
Anrufliste ausgetragen werden?", "CallList.CallItem", 3) == 6) // Es wurde
"Ja/Yes" angeklickt
        {
            callItem.Remove();
        }
        else
        {
            if (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", 3) == 6)
            {
                callItem.GotoRecord();
            }
        }
    }

    callItem.Dispose();
}

callList.Dispose();
```

Number, read-only

Beschreibung:

Die hinterlegte Rufnummer.

Typ:

String

Beispiel VBScript:

```
' Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

Dim oCallList : Set oCallList = cRM.phonemanager.CallList
Dim oCallItem
Dim nCount : nCount = 0
Dim nDialRetriesCount : nDialRetriesCount = 0
Dim sFirstInfo : sFirstInfo = ""
Dim sInfo : sInfo = ""
Dim sLastInfo : sLastInfo = ""
Dim sNumber : sNumber = ""

For nCount = 1 To oCallList.Count

    Set oCallItem = oCallList.Item(nCount)
```

```
    If (oCallItem.DialRetriesCount > 10) Then
        Call oCallItem.Remove()
    Else
        nDialRetriesCount = oCallItem.DialRetriesCount
        sFirstInfo = oCallItem.FirstInfo
        sInfo = oCallItem.Info
        sLastInfo = oCallItem.LastInfo
        sNumber = oCallItem.Number

        If (cRM.DialogMessageBox("Die Nummer " & sNumber & " (" & sFirstInfo & " -
" & sInfo & " - " & sLastInfo & ") konnte mit " & CStr(nDialRetriesCount) & "
Anrufversuchen nicht erreicht werden." & vbCrlf & "Soll diese Rufnummer aus der
Anrufliste augetragen werden?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
            Call oCallItem.Remove()
        Else
            If (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehen Datensatz im combit CRM zu
springen?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
                Call oCallItem.GotoRecord()
            End If
        End If
    End If

    Set oCallItem = Nothing

Next

Set oCallList = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

CallList callList = cRM.PhoneManager.CallList;
CallItem callItem;
long dialRetriesCount;
string firstInfo;
string info;
string lastInfo;
string number;

for (int i = 1; i <= callList.Count; i++)
{
    callItem = callList.Item(i);

    if (callItem.DialRetriesCount > 10)
    {
        callItem.Remove();
    }
    else
    {
        dialRetriesCount = callItem.DialRetriesCount;
        firstInfo = callItem.FirstInfo;
        info = callItem.Info;
        lastInfo = callItem.LastInfo;
        number = callItem.Number;

       if (cRM.DialogMessageBox("Die Nummer " + number + "(" + firstInfo + " - " +
info + " - " + lastInfo + ") konnte mit " + dialRetriesCount.ToString() + "
Anrufversuchen nicht erreicht werden." + "\r\n" + "Soll diese Rufnummer aus der
Anrufliste ausgetragen werden?", "CallList.CallItem", 3) == 6) // Es wurde
"Ja/Yes" angeklickt
        {
            callItem.Remove();
        }
        else
        {
```

```
            if (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", 3) == 6)
            {
                callItem.GotoRecord();
            }
        }
    }

    callItem.Dispose();
}

callList.Dispose();
```

3.8.2 Methoden

GotoRecord

Beschreibung:

Springt zum korrespondierenden Datensatz der Anwendung.

Rückgabewert:

Bool

Wert	Beschreibung
True	Datensatz konnte "angesprungen" werden.
False	Keine Verknüpfung zum Datensatz; Datensatz konnte nicht "angesprungen" werden.

Beispiel VBScript:

```
' Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen bzw. gibt dem
Nutzer eine Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

Dim oCallList : Set oCallList = cRM.phonemanager.CallList
Dim oCallItem
Dim nCount : nCount = 0
Dim nDialRetriesCount : nDialRetriesCount = 0
Dim sFirstInfo : sFirstInfo = ""
Dim sInfo : sInfo = ""
Dim sLastInfo : sLastInfo = ""
Dim sNumber : sNumber = ""

For nCount = 1 To oCallList.Count

    Set oCallItem = oCallList.Item(nCount)

    If (oCallItem.DialRetriesCount > 10) Then
        Call oCallItem.Remove()
    Else
        nDialRetriesCount = oCallItem.DialRetriesCount
        sFirstInfo = oCallItem.FirstInfo
        sInfo = oCallItem.Info
        sLastInfo = oCallItem.LastInfo
        sNumber = oCallItem.Number

        If (cRM.DialogMessageBox("Die Nummer " & sNumber & " (" & sFirstInfo & " -
" & sInfo & " - " & sLastInfo & ") konnte mit " & CStr(nDialRetriesCount) & "
Anrufversuchen nicht erreicht werden." & vbCrlf & "Soll diese Rufnummer aus der
Anrufliste augetragen werden?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
            Call oCallItem.Remove()
        Else
```

```
            If (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehen Datensatz im combit CRM zu
springen?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
                Call oCallItem.GotoRecord()
            End If
        End If
    End If

    Set oCallItem = Nothing

Next

Set oCallList = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen bzw. gibt dem
Nutzer eine Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

CallList callList = cRM.PhoneManager.CallList;
CallItem callItem;
long dialRetriesCount;
string firstInfo;
string info;
string lastInfo;
string number;

for (int i = 1; i <= callList.Count; i++)
{
    callItem = callList.Item(i);

    if (callItem.DialRetriesCount > 10)
    {
        callItem.Remove();
    }
    else
    {
        dialRetriesCount = callItem.DialRetriesCount;
        firstInfo = callItem.FirstInfo;
        info = callItem.Info;
        lastInfo = callItem.LastInfo;
        number = callItem.Number;

       if (cRM.DialogMessageBox("Die Nummer " + number + "(" + firstInfo + " - " +
info + " - " + lastInfo + ") konnte mit " + dialRetriesCount.ToString() + "
Anrufversuchen nicht erreicht werden." + "\r\n" + "Soll diese Rufnummer aus der
Anrufliste ausgetragen werden?", "CallList.CallItem", 3) == 6) // Es wurde
"Ja/Yes" angeklickt
        {
            callItem.Remove();
        }
        else
        {
            if (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", 3) == 6)
            {
                callItem.GotoRecord();
            }
        }
    }

    callItem.Dispose();
}

callList.Dispose();
```

Remove

Beschreibung:

Entfernt den Eintrag aus der Anrufliste.

Beispiel VBScript:

```
' Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen bzw. gibt dem
Nutzer eine Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

Dim oCallList : Set oCallList = cRM.phonemanager.CallList
Dim oCallItem
Dim nCount : nCount = 0
Dim nDialRetriesCount : nDialRetriesCount = 0
Dim sFirstInfo : sFirstInfo = ""
Dim sInfo : sInfo = ""
Dim sLastInfo : sLastInfo = ""
Dim sNumber : sNumber = ""

For nCount = 1 To oCallList.Count

    Set oCallItem = oCallList.Item(nCount)

    If (oCallItem.DialRetriesCount > 10) Then
        Call oCallItem.Remove()
    Else
        nDialRetriesCount = oCallItem.DialRetriesCount
        sFirstInfo = oCallItem.FirstInfo
        sInfo = oCallItem.Info
        sLastInfo = oCallItem.LastInfo
        sNumber = oCallItem.Number

        If (cRM.DialogMessageBox("Die Nummer " & sNumber & " (" & sFirstInfo & " -
" & sInfo & " - " & sLastInfo & ") konnte mit " & CStr(nDialRetriesCount) & "
Anrufversuchen nicht erreicht werden." & vbCrlf & "Soll diese Rufnummer aus der
Anrufliste augetragen werden?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
            Call oCallItem.Remove()
        Else
            If (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehen Datensatz im combit CRM zu
springen?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
                Call oCallItem.GotoRecord()
            End If
        End If
    End If

    Set oCallItem = Nothing

Next

Set oCallList = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen bzw. gibt dem
Nutzer eine Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

CallList callList = cRM.PhoneManager.CallList;
CallItem callItem;
long dialRetriesCount;
string firstInfo;
string info;
string lastInfo;
string number;

for (int i = 1; i <= callList.Count; i++)
{
```

```
    callItem = callList.Item(i);

    if (callItem.DialRetriesCount > 10)
    {
        callItem.Remove();
    }
    else
    {
        dialRetriesCount = callItem.DialRetriesCount;
        firstInfo = callItem.FirstInfo;
        info = callItem.Info;
        lastInfo = callItem.LastInfo;
        number = callItem.Number;

       if (cRM.DialogMessageBox("Die Nummer " + number + "(" + firstInfo + " - " +
info + " - " + lastInfo + ") konnte mit " + dialRetriesCount.ToString() + "
Anrufversuchen nicht erreicht werden." + "\r\n" + "Soll diese Rufnummer aus der
Anrufliste ausgetragen werden?", "CallList.CallItem", 3) == 6) // Es wurde
"Ja/Yes" angeklickt
        {
            callItem.Remove();
        }
        else
        {
            if (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", 3) == 6)
            {
                callItem.GotoRecord();
            }
        }
    }

    callItem.Dispose();
}

callList.Dispose();
```

3.9 CallList Objekt

3.9.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der Einträge in der Anrufliste zurück.

Rückgabewert:

Long

Beispiel VBScript:

```
' Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen bzw. gibt dem
Nutzer eine Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

Dim oCallList : Set oCallList = cRM.phonemanager.CallList
Dim oCallItem
Dim nCount : nCount = 0
Dim nDialRetriesCount : nDialRetriesCount = 0
Dim sFirstInfo : sFirstInfo = ""
Dim sInfo : sInfo = ""
Dim sLastInfo : sLastInfo = ""
Dim sNumber : sNumber = ""
```

```
For nCount = 1 To oCallList.Count

    Set oCallItem = oCallList.Item(nCount)

    If (oCallItem.DialRetriesCount > 10) Then
        Call oCallItem.Remove()
    Else
        nDialRetriesCount = oCallItem.DialRetriesCount
        sFirstInfo = oCallItem.FirstInfo
        sInfo = oCallItem.Info
        sLastInfo = oCallItem.LastInfo
        sNumber = oCallItem.Number

        If (cRM.DialogMessageBox("Die Nummer " & sNumber & " (" & sFirstInfo & " -
" & sInfo & " - " & sLastInfo & ") konnte mit " & CStr(nDialRetriesCount) & "
Anrufversuchen nicht erreicht werden." & vbCrlf & "Soll diese Rufnummer aus der
Anrufliste augetragen werden?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
            Call oCallItem.Remove()
        Else
            If (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehen Datensatz im combit CRM zu
springen?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
                Call oCallItem.GotoRecord()
            End If
        End If
    End If

    Set oCallItem = Nothing

Next

Set oCallList = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen bzw. gibt dem
Nutzer eine Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

CallList callList = cRM.PhoneManager.CallList;
CallItem callItem;
long dialRetriesCount;
string firstInfo;
string info;
string lastInfo;
string number;

for (int i = 1; i <= callList.Count; i++)
{
    callItem = callList.Item(i);

    if (callItem.DialRetriesCount > 10)
    {
        callItem.Remove();
    }
    else
    {
        dialRetriesCount = callItem.DialRetriesCount;
        firstInfo = callItem.FirstInfo;
        info = callItem.Info;
        lastInfo = callItem.LastInfo;
        number = callItem.Number;

       if (cRM.DialogMessageBox("Die Nummer " + number + "(" + firstInfo + " - " +
info + " - " + lastInfo + ") konnte mit " + dialRetriesCount.ToString() + "
Anrufversuchen nicht erreicht werden." + "\r\n" + "Soll diese Rufnummer aus der
Anrufliste ausgetragen werden?", "CallList.CallItem", 3) == 6) // Es wurde
"Ja/Yes" angeklickt
        {
```

```
                callItem.Remove();
            }
            else
            {
                if (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", 3) == 6)
                {
                    callItem.GotoRecord();
                }
            }
        }

        callItem.Dispose();
    }

    callList.Dispose();
```

3.9.2 Methoden

Item

Beschreibung:

Gibt einen Eintrag der Anrufliste zurück. Es muss die Index-Nummer des Eintrages übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

CallItem

Beispiel VBScript:

```
' Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen bzw. gibt dem
Nutzer eine Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

Dim oCallList : Set oCallList = cRM.phonemanager.CallList
Dim oCallItem
Dim nCount : nCount = 0
Dim nDialRetriesCount : nDialRetriesCount = 0
Dim sFirstInfo : sFirstInfo = ""
Dim sInfo : sInfo = ""
Dim sLastInfo : sLastInfo = ""
Dim sNumber : sNumber = ""

For nCount = 1 To oCallList.Count

    Set oCallItem = oCallList.Item(nCount)

    If (oCallItem.DialRetriesCount > 10) Then
        Call oCallItem.Remove()
    Else
        nDialRetriesCount = oCallItem.DialRetriesCount
        sFirstInfo = oCallItem.FirstInfo
        sInfo = oCallItem.Info
        sLastInfo = oCallItem.LastInfo
        sNumber = oCallItem.Number

        If (cRM.DialogMessageBox("Die Nummer " & sNumber & " (" & sFirstInfo & " -
" & sInfo & " - " & sLastInfo & ") konnte mit " & CStr(nDialRetriesCount) & "
```

```
Anrufversuchen nicht erreicht werden." & vbCrlf & "Soll diese Rufnummer aus der
Anrufliste augetragen werden?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
            Call oCallItem.Remove()
        Else
            If (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehen Datensatz im combit CRM zu
springen?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
                Call oCallItem.GotoRecord()
            End If
        End If
    End If

    Set oCallItem = Nothing

Next

Set oCallList = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen bzw. gibt dem
Nutzer eine Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

CallList callList = cRM.PhoneManager.CallList;
CallItem callItem;
long dialRetriesCount;
string firstInfo;
string info;
string lastInfo;
string number;

for (int i = 1; i <= callList.Count; i++)
{
    callItem = callList.Item(i);

    if (callItem.DialRetriesCount > 10)
    {
        callItem.Remove();
    }
    else
    {
        dialRetriesCount = callItem.DialRetriesCount;
        firstInfo = callItem.FirstInfo;
        info = callItem.Info;
        lastInfo = callItem.LastInfo;
        number = callItem.Number;

       if (cRM.DialogMessageBox("Die Nummer " + number + "(" + firstInfo + " - " +
info + " - " + lastInfo + ") konnte mit " + dialRetriesCount.ToString() + "
Anrufversuchen nicht erreicht werden." + "\r\n" + "Soll diese Rufnummer aus der
Anrufliste ausgetragen werden?", "CallList.CallItem", 3) == 6) // Es wurde
"Ja/Yes" angeklickt
        {
            callItem.Remove();
        }
        else
        {
            if (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", 3) == 6)
            {
                callItem.GotoRecord();
            }
        }
    }

    callItem.Dispose();
}
```

```
callList.Dispose();
```

3.10 Categories Objekt

3.10.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der Einträge in der Sammlung zurück.

Typ:

Long

Beispiel VBScript:

```
Dim oCategories : Set oCategories =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).C
ategories
Call cRM.DialogMessageBox("Der ausgewählte Termin besitzt " &
CStr(oCategories.Count) & " zugewiesene Kategorien.", "Categories.Count",
vbOkOnly)
Set oCategories = Nothing
```

Beispiel C#-Script:

```
Categories categories =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).Ca
tegories;
cRM.DialogMessageBox("Der ausgewählte Termin besitzt " +
categories.Count.ToString() + " zugewiesene Kategorien.", "Categories.Count", 0);
categories.Dispose();
```

3.10.2 Methoden

Add

Beschreibung:

Fügt eine neue Kategorie zu einer Aufgabe/einem Termin hinzu.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name der Kategorie.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim sCategoryName : sCategoryName = "Meeting"
Call
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).C
ategories.Add(sCategoryName)
```

Beispiel C#-Script:

```
string categoryName = "Meeting";
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).Ca
tegories.Add(categoryName);
```

Item

Beschreibung:

Gibt eine Kategorie zurück. Es muss die Index-Nummer der Kategorie übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Category
NULL (wenn die Kategorie nicht existiert)

Beispiel VBScript:

```
' Prüft, ob ein bestimmter Termin die Kategorie "Meeting" besitzt

Dim oCategories : Set oCategories =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).C
ategories
Dim oCategory
Dim nCount : nCount = 0
Dim bCategoryMeeting : bCategoryMeeting = False

For nCount = 1 To oCategories.Count

    Set oCategory = oCategories.Item(nCount)

    If (oCategory.Name = "Meeting") Then
        bCategoryMeeting = True
    End If

    Set oCategory = Nothing

Next

Call cRM.DialogMessageBox("Mindestens ein Termin besitzt die Kategorie ""Meeting""
(ID: " & oCategory.ID & ", Symbol: " & oCategory.Symbol & ".", "Categories.Item",
vbOkOnly)
Set oCategories = Nothing
```

Beispiel C#-Script:

```
// Prüft, ob ein bestimmter Termin die Kategorie "Meeting" besitzt

Categories categories =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).Ca
tegories;
Category category;
bool isCategoryMeeting = false;

for (int i = 1; i <= categories.Count; i++)
{
    category = categories.Item(i)
    if (category.Name == "Meeting")
    {
        isCategoryMeeting = true;
    }

    category.Dispose();
}

cRM.DialogMessageBox("Mindestens ein Termin besitzt die Kategorie \"Meeting\" (ID:
" + category.ID + ", Symbol: " + category.Symbol + ".", "Categories.Item", 0);
categories.Dispose();
```

ItemByName

Beschreibung:

Gibt eine Kategorie anhand ihres Namens zurück.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name der Kategorie.

Rückgabewert:

Category
NULL (wenn die Kategorie nicht existiert)

Beispiel VBScript:

```
Dim oCategories : Set oCategories =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).C
ategories
Dim oCategory : Set oCategory = oCategories.ItemByName("Meeting")
Call oCategory.Remove()
Set oCategory = Nothing
Set oCategories = Nothing
```

Beispiel C#-Script:

```
Categories categories =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).Ca
tegories;
Category category = categories.ItemByName("Meeting");
categories.Remove();
category.Dispose();
categories.Dispose();
```

Remove

Beschreibung:

Löscht eine Kategorie. Es muss die Index-Nummer der Kategorie übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oCategories : Set oCategories =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).C
ategories
Dim oCategory : Set oCategory = oCategories.ItemByName("Meeting")
Call oCategory.Remove()
Set oCategory = Nothing
Set oCategories = Nothing
```

Beispiel C#-Script:

```
Categories categories =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).Ca
tegories;
Category category = categories.ItemByName("Meeting");
categories.Remove();
category.Dispose();
```

```
categories.Dispose();
```

3.11 Category Objekt

3.11.1 Eigenschaften

ID, read-only

Beschreibung:

Liefert die ID der Kategorie zurück.

Typ:

String

Beispiel VBScript:

```
' Prüft, ob ein bestimmter Termin die Kategorie "Meeting" besitzt

Dim oCategories : Set oCategories =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).C
ategories
Dim oCategory
Dim nCount : nCount = 0
Dim bCategoryMeeting : bCategoryMeeting = False

For nCount = 1 To oCategories.Count

    Set oCategory = oCategories.Item(nCount)

    If (oCategory.Name = "Meeting") Then
        bCategoryMeeting = True
    End If

    Set oCategory = Nothing

Next

Call cRM.DialogMessageBox("Mindestens ein Termin besitzt die Kategorie ""Meeting""
(ID: " & oCategory.ID & ", Symbol: " & oCategory.Symbol & ".", "Categories.Item",
vbOkOnly)
Set oCategories = Nothing
```

Beispiel C#-Script:

```
// Prüft, ob ein bestimmter Termin die Kategorie "Meeting" besitzt

Categories categories =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).Ca
tegories;
Category category;
bool isCategoryMeeting = false;
string id = string.Empty;
long symbol = -1;

for (int i = 1; i <= categories.Count; i++)
{
    category = categories.Item(i);

    if (category.Name == "Meeting")
    {
        id = category.ID;
        symbol = category.Symbol;
        isCategoryMeeting = true;
    }

    category.Dispose();
```

```
}

cRM.DialogMessageBox("Im ausgewählten Termin gibt es die Kategorie \"Meeting\"
(ID: " + id + ", Symbol: " + symbol + ".", "Categories.Item", 0);
categories.Dispose();
```

Name, read-only

Beschreibung:

Liefert den Namen der Kategorie zurück.

Typ:

String

Beispiel VBScript:

```
' Prüft, ob ein bestimmter Termin die Kategorie "Meeting" besitzt

Dim oCategories : Set oCategories =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).C
ategories
Dim oCategory
Dim nCount : nCount = 0
Dim bCategoryMeeting : bCategoryMeeting = False

For nCount = 1 To oCategories.Count

    Set oCategory = oCategories.Item(nCount)

    If (oCategory.Name = "Meeting") Then
        bCategoryMeeting = True
    End If

    Set oCategory = Nothing

Next

Call cRM.DialogMessageBox("Mindestens ein Termin besitzt die Kategorie ""Meeting""
(ID: " & oCategory.ID & ", Symbol: " & oCategory.Symbol & ".", "Categories.Item",
vbOkOnly)
Set oCategories = Nothing
```

Beispiel C#-Script:

```
// Prüft, ob ein bestimmter Termin die Kategorie "Meeting" besitzt

Categories categories =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).Ca
tegories;
Category category;
bool isCategoryMeeting = false;
string id = string.Empty;
long symbol = -1;

for (int i = 1; i <= categories.Count; i++)
{
    category = categories.Item(i);

    if (category.Name == "Meeting")
    {
        id = category.ID;
        symbol = category.Symbol;
        isCategoryMeeting = true;
    }

    category.Dispose();
}
```

```
cRM.DialogMessageBox("Im ausgewählten Termin gibt es die Kategorie \"Meeting\"
(ID: " + id + ", Symbol: " + symbol + ".", "Categories.Item", 0);
categories.Dispose();
```

Symbol, read-only

Beschreibung:

Liefert die ID des Symbols der Kategorie zurück.

Typ:

Long

Beispiel VBScript:

```
' Prüft, ob ein bestimmter Termin die Kategorie "Meeting" besitzt

Dim oCategories : Set oCategories =
cRM.CurrentProject.timemanager.Appointments.ItemByUniqueID(sUniqueAppointmentID).C
ategories
Dim oCategory
Dim nCount : nCount = 0
Dim bCategoryMeeting : bCategoryMeeting = False

For nCount = 1 To oCategories.Count

    Set oCategory = oCategories.Item(nCount)

    If (oCategory.Name = "Meeting") Then
        bCategoryMeeting = True
    End If

    Set oCategory = Nothing

Next

Call cRM.DialogMessageBox("Mindestens ein Termin besitzt die Kategorie ""Meeting""
(ID: " & oCategory.ID & ", Symbol: " & oCategory.Symbol & ".", "Categories.Item",
vbOkOnly)
Set oCategories = Nothing
```

Beispiel C#-Script:

```
// Prüft, ob ein bestimmter Termin die Kategorie "Meeting" besitzt

Categories categories =
cRM.CurrentProject.TimeManager.Appointments.ItemByUniqueID(uniqueAppointmentID).Ca
tegories;
Category category;
bool isCategoryMeeting = false;
string id = string.Empty;
long symbol = -1;

for (int i = 1; i <= categories.Count; i++)
{
    category = categories.Item(i);

    if (category.Name == "Meeting")
    {
        id = category.ID;
        symbol = category.Symbol;
        isCategoryMeeting = true;
    }

    category.Dispose();
}

cRM.DialogMessageBox("Im ausgewählten Termin gibt es die Kategorie \"Meeting\"
(ID: " + id + ", Symbol: " + symbol + ".", "Categories.Item", 0);
categories.Dispose();
```

3.12 CompanyInfo Objekt

Ermöglicht den Zugriff per COM auf die konfigurierten Firmenstammdaten des Projektes (Menüpunkt: **Datei > Information > Firmenstammdaten**).

Beispiel VBScript:

```
' Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("AccountNo", oCompanyInfo.AccountNo)
Call dicCompanyInfo.Add("Bank", oCompanyInfo.Bank)
Call dicCompanyInfo.Add("BankCode", oCompanyInfo.BankCode)
Call dicCompanyInfo.Add("City", oCompanyInfo.City)
Call dicCompanyInfo.Add("Company", oCompanyInfo.Company)
Call dicCompanyInfo.Add("Company2", oCompanyInfo.Company2)
Call dicCompanyInfo.Add("Company3", oCompanyInfo.Company3)
Call dicCompanyInfo.Add("Country", oCompanyInfo.Country)
Call dicCompanyInfo.Add("Email", oCompanyInfo.Email)
Call dicCompanyInfo.Add("Extra1", oCompanyInfo.Extra1)
Call dicCompanyInfo.Add("Extra2", oCompanyInfo.Extra2)
Call dicCompanyInfo.Add("Fax", oCompanyInfo.Fax)
Call dicCompanyInfo.Add("IBAN", oCompanyInfo.IBAN)
Call dicCompanyInfo.Add("Internet", oCompanyInfo.Internet)
Call dicCompanyInfo.Add("Logo", oCompanyInfo.Logo)
Call dicCompanyInfo.Add("Phone", oCompanyInfo.Phone)
Call dicCompanyInfo.Add("Street", oCompanyInfo.Street)
Call dicCompanyInfo.Add("VatID", oCompanyInfo.VatID)
Call dicCompanyInfo.Add("ZIP", oCompanyInfo.ZIP)

Dim oListCompanyInfoUserDefined : Set oListCompanyInfoUserDefined =
oCompanyInfo.UserDefinedFields
Dim oCompanyInfoUserDefinedItem
Dim nCount : nCount = 0

For nCount = 1 To oListCompanyInfoUserDefined.Count

    Set oCompanyInfoUserDefinedItem = oListCompanyInfoUserDefined.Item(nCount)

    If (oCompanyInfoUserDefinedItem.Type = 2) Then ' Wenn es eine Textinformation
ist
        Call dicCompanyInfo.Add(oCompanyInfoUserDefinedItem.Key,
oCompanyInfoUserDefinedItem.Value)
    End If

    Set oCompanyInfoUserDefinedItem = Nothing

Next

Set oListCompanyInfoUserDefined = Nothing

Dim sFullCompanyInfo : sFullCompanyInfo = ""
Dim dicItem

For Each dicItem In dicCompanyInfo
    If (Len(dicCompanyInfo(dicItem)) > 0) Then
        sFullCompanyInfo = sFullCompanyInfo & dicItem & ": " &
dicCompanyInfo(dicItem) & vbCrlf
    End If
Next

WScript.ClipboardText = sFullCompanyInfo

Call cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die
Zwischenablage übernommen.", "CompanyInfo", vbOkOnly)
```

```
Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
// Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("AccountNo", companyInfo.AccountNo);
dictionaryCompanyInfo.Add("Bank", companyInfo.Bank);
dictionaryCompanyInfo.Add("BankCode", companyInfo.BankCode);
dictionaryCompanyInfo.Add("City", companyInfo.City);
dictionaryCompanyInfo.Add("Company", companyInfo.Company);
dictionaryCompanyInfo.Add("Company2", companyInfo.Company2);
dictionaryCompanyInfo.Add("Company3", companyInfo.Company3);
dictionaryCompanyInfo.Add("Country", companyInfo.Country);
dictionaryCompanyInfo.Add("Email", companyInfo.Email);
dictionaryCompanyInfo.Add("Extra1", companyInfo.Extra1);
dictionaryCompanyInfo.Add("Extra2", companyInfo.Extra2);
dictionaryCompanyInfo.Add("Fax", companyInfo.Fax);
dictionaryCompanyInfo.Add("IBAN", companyInfo.IBAN);
dictionaryCompanyInfo.Add("Internet", companyInfo.Internet);
dictionaryCompanyInfo.Add("Logo", companyInfo.Logo);
dictionaryCompanyInfo.Add("Phone", companyInfo.Phone);
dictionaryCompanyInfo.Add("Street", companyInfo.Street);
dictionaryCompanyInfo.Add("VatID", companyInfo.VatID);
dictionaryCompanyInfo.Add("ZIP", companyInfo.ZIP);

ListCompanyInfoUserDefined listCompanyInfoUserDefined =
companyInfo.UserDefinedFields;

foreach (CompanyInfoUserDefinedItem companyInfoUserDefinedItem in
listCompanyInfoUserDefined)
{
    if (companyInfoUserDefinedItem.Type == 2)
    {
        dictionaryCompanyInfo.Add(companyInfoUserDefinedItem.Key,
companyInfoUserDefinedItem.Value);
    }
}

string fullCompanyInfo = string.Empty;

foreach (var dictionaryItem in dictionaryCompanyInfo)
{
    if (dictionaryItem.Value != "")
    {
        fullCompanyInfo = fullCompanyInfo + dictionaryItem.Key + ": " +
dictionaryItem.Value + System.Environment.NewLine;
    }
}

WScript.ClipboardText = fullCompanyInfo;
cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die Zwischenablage
übernommen.", "CompanyInfo", 0);

companyInfo.Dispose();
```

3.12.1 Eigenschaften

AccountNo, read-only

Beschreibung:

Liefert die Konto-Nr. zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Call dicCompanyInfo.Add("AccountNo", oCompanyInfo.AccountNo)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("AccountNo", companyInfo.AccountNo);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Bank, read-only

Beschreibung:

Liefert den Banknamen (Bank) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Bank", oCompanyInfo.Bank)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Bank", companyInfo.Bank);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

BankCode, read-only

Beschreibung:

Liefert die Bankleitzahl (BLZ) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("BankCode", oCompanyInfo.BankCode)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("BankCode", companyInfo.BankCode);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

City, read-only

Beschreibung:

Liefert den Ort zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("City", oCompanyInfo.City)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("City", companyInfo.City);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Company, read-only

Beschreibung:

Liefert den Firmennamen (Firma) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Company", oCompanyInfo.Company)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Company", companyInfo.Company);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Company2, read-only

Beschreibung:

Liefert den Firmennamen (Firma2) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Company2", oCompanyInfo.Company2)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Company2", companyInfo.Company2);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Company3, read-only

Beschreibung:

Liefert den Firmennamen (Firma3) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Company3", oCompanyInfo.Company3)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Company3", companyInfo.Company3);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Country, read-only

Beschreibung:

Liefert das Land zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Country", oCompanyInfo.Country)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Country", companyInfo.Country);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Email, read-only

Beschreibung:

Liefert die E-Mail-Adresse (E-Mail) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Email", oCompanyInfo.Email)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Email", companyInfo.Email);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Extra1, read-only

Beschreibung:

Liefert Zusatz zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Extra1", oCompanyInfo.Extra1)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Extra1", companyInfo.Extra1);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Extra2, read-only

Beschreibung:

Liefert Zusatz2 zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Extra2", oCompanyInfo.Extra2)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Extra2", companyInfo.Extra2);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Fax, read-only

Beschreibung:

Liefert die Telefaxnummer (Telefax) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Fax", oCompanyInfo.Fax)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Fax", companyInfo.Fax);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

IBAN, read-only

Beschreibung:

Liefert den IBAN-Code zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("IBAN", oCompanyInfo.IBAN)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("IBAN", companyInfo.IBAN);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Internet, read-only

Beschreibung:

Liefert die Internetadresse (Internet) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Internet", oCompanyInfo.Internet)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Internet", companyInfo.Internet);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Logo, read-only

Beschreibung:

Liefert das konfigurierte Firmenlogo (Dateiname) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Logo", oCompanyInfo.Logo)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Logo", companyInfo.Logo);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Phone, read-only

Beschreibung:

Liefert die Telefonnummer (Telefon) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Phone", oCompanyInfo.Phone)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Phone", companyInfo.Phone);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

Street, read-only

Beschreibung:

Liefert die Strasse zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("Street", oCompanyInfo.Street)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("Street", companyInfo.Street);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

UserDefinedFields

Beschreibung:

Liefert eine Sammlung aller benutzerdefinierten Felder der Firmenstammdaten als Objekt vom Typ **ListCompanyInfoUserDefined** zurück.

Typ:

ListCompanyInfoUserDefined

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo

Dim oListCompanyInfoUserDefined : Set oListCompanyInfoUserDefined =
oCompanyInfo.UserDefinedFields

Set oListCompanyInfoUserDefined = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;

ListCompanyInfoUserDefined listCompanyInfoUserDefined =
companyInfo.UserDefinedFields;


companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **CompanyInfo Objekt**.

VatID, read-only

Beschreibung:

Liefert die Steuer-Nr. zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("VatID", oCompanyInfo.VatID)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("VatID", companyInfo.VatID);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel CompanyInfo Objekt.

ZIP, read-only

Beschreibung:

Liefert die Postleitzahl (PLZ) zurück.

Typ:

String

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("ZIP", oCompanyInfo.ZIP)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("ZIP", companyInfo.ZIP);

companyInfo.Dispose();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel CompanyInfo Objekt.

3.13 CompanyInfoUserDefinedItem Objekt

Bietet Zugriff auf ein Element der Sammlung der benutzerdefinierten Felder der Firmenstammdaten.

3.13.1 Eigenschaften

Key, read-only

Beschreibung:

Liefert den Namen zurück.

Typ:

String

Beispiel VBScript:

```
' Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("AccountNo", oCompanyInfo.AccountNo)
Call dicCompanyInfo.Add("Bank", oCompanyInfo.Bank)
Call dicCompanyInfo.Add("BankCode", oCompanyInfo.BankCode)
Call dicCompanyInfo.Add("City", oCompanyInfo.City)
Call dicCompanyInfo.Add("Company", oCompanyInfo.Company)
Call dicCompanyInfo.Add("Company2", oCompanyInfo.Company2)
Call dicCompanyInfo.Add("Company3", oCompanyInfo.Company3)
Call dicCompanyInfo.Add("Country", oCompanyInfo.Country)
Call dicCompanyInfo.Add("Email", oCompanyInfo.Email)
Call dicCompanyInfo.Add("Extra1", oCompanyInfo.Extra1)
Call dicCompanyInfo.Add("Extra2", oCompanyInfo.Extra2)
Call dicCompanyInfo.Add("Fax", oCompanyInfo.Fax)
Call dicCompanyInfo.Add("IBAN", oCompanyInfo.IBAN)
Call dicCompanyInfo.Add("Internet", oCompanyInfo.Internet)
Call dicCompanyInfo.Add("Logo", oCompanyInfo.Logo)
Call dicCompanyInfo.Add("Phone", oCompanyInfo.Phone)
Call dicCompanyInfo.Add("Street", oCompanyInfo.Street)
Call dicCompanyInfo.Add("VatID", oCompanyInfo.VatID)
Call dicCompanyInfo.Add("ZIP", oCompanyInfo.ZIP)

Dim oListCompanyInfoUserDefined : Set oListCompanyInfoUserDefined =
oCompanyInfo.UserDefinedFields
Dim oCompanyInfoUserDefinedItem
Dim nCount : nCount = 0

For nCount = 1 To oListCompanyInfoUserDefined.Count

    Set oCompanyInfoUserDefinedItem = oListCompanyInfoUserDefined.Item(nCount)

    If (oCompanyInfoUserDefinedItem.Type = 2) Then ' Wenn es eine Textinformation
ist
        Call dicCompanyInfo.Add(oCompanyInfoUserDefinedItem.Key,
oCompanyInfoUserDefinedItem.Value)
    End If

    Set oCompanyInfoUserDefinedItem = Nothing

Next

Set oListCompanyInfoUserDefined = Nothing

Dim sFullCompanyInfo : sFullCompanyInfo = ""
Dim dicItem

For Each dicItem In dicCompanyInfo
    If (Len(dicCompanyInfo(dicItem)) > 0) Then
        sFullCompanyInfo = sFullCompanyInfo & dicItem & ": " &
dicCompanyInfo(dicItem) & vbCrlf
    End If
```

```
Next

WScript.ClipboardText = sFullCompanyInfo

Call cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die
Zwischenablage übernommen.", "CompanyInfo", vbOkOnly)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
// Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("AccountNo", companyInfo.AccountNo);
dictionaryCompanyInfo.Add("Bank", companyInfo.Bank);
dictionaryCompanyInfo.Add("BankCode", companyInfo.BankCode);
dictionaryCompanyInfo.Add("City", companyInfo.City);
dictionaryCompanyInfo.Add("Company", companyInfo.Company);
dictionaryCompanyInfo.Add("Company2", companyInfo.Company2);
dictionaryCompanyInfo.Add("Company3", companyInfo.Company3);
dictionaryCompanyInfo.Add("Country", companyInfo.Country);
dictionaryCompanyInfo.Add("Email", companyInfo.Email);
dictionaryCompanyInfo.Add("Extra1", companyInfo.Extra1);
dictionaryCompanyInfo.Add("Extra2", companyInfo.Extra2);
dictionaryCompanyInfo.Add("Fax", companyInfo.Fax);
dictionaryCompanyInfo.Add("IBAN", companyInfo.IBAN);
dictionaryCompanyInfo.Add("Internet", companyInfo.Internet);
dictionaryCompanyInfo.Add("Logo", companyInfo.Logo);
dictionaryCompanyInfo.Add("Phone", companyInfo.Phone);
dictionaryCompanyInfo.Add("Street", companyInfo.Street);
dictionaryCompanyInfo.Add("VatID", companyInfo.VatID);
dictionaryCompanyInfo.Add("ZIP", companyInfo.ZIP);

ListCompanyInfoUserDefined listCompanyInfoUserDefined =
companyInfo.UserDefinedFields;

foreach (CompanyInfoUserDefinedItem companyInfoUserDefinedItem in
listCompanyInfoUserDefined)
{
    if (companyInfoUserDefinedItem.Type == 2)
    {
        dictionaryCompanyInfo.Add(companyInfoUserDefinedItem.Key,
companyInfoUserDefinedItem.Value);
    }
}

string fullCompanyInfo = string.Empty;

foreach (var dictionaryItem in dictionaryCompanyInfo)
{
    if (dictionaryItem.Value != "")
    {
        fullCompanyInfo = fullCompanyInfo + dictionaryItem.Key + ": " +
dictionaryItem.Value + System.Environment.NewLine;
    }
}

WScript.ClipboardText = fullCompanyInfo;
cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die Zwischenablage
übernommen.", "CompanyInfo", 0);

companyInfo.Dispose();
```

Value, read-only

Beschreibung:

Liefert den Inhalt zurück.

Typ:

String

Beispiel VBScript:

```
' Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("AccountNo", oCompanyInfo.AccountNo)
Call dicCompanyInfo.Add("Bank", oCompanyInfo.Bank)
Call dicCompanyInfo.Add("BankCode", oCompanyInfo.BankCode)
Call dicCompanyInfo.Add("City", oCompanyInfo.City)
Call dicCompanyInfo.Add("Company", oCompanyInfo.Company)
Call dicCompanyInfo.Add("Company2", oCompanyInfo.Company2)
Call dicCompanyInfo.Add("Company3", oCompanyInfo.Company3)
Call dicCompanyInfo.Add("Country", oCompanyInfo.Country)
Call dicCompanyInfo.Add("Email", oCompanyInfo.Email)
Call dicCompanyInfo.Add("Extra1", oCompanyInfo.Extra1)
Call dicCompanyInfo.Add("Extra2", oCompanyInfo.Extra2)
Call dicCompanyInfo.Add("Fax", oCompanyInfo.Fax)
Call dicCompanyInfo.Add("IBAN", oCompanyInfo.IBAN)
Call dicCompanyInfo.Add("Internet", oCompanyInfo.Internet)
Call dicCompanyInfo.Add("Logo", oCompanyInfo.Logo)
Call dicCompanyInfo.Add("Phone", oCompanyInfo.Phone)
Call dicCompanyInfo.Add("Street", oCompanyInfo.Street)
Call dicCompanyInfo.Add("VatID", oCompanyInfo.VatID)
Call dicCompanyInfo.Add("ZIP", oCompanyInfo.ZIP)

Dim oListCompanyInfoUserDefined : Set oListCompanyInfoUserDefined =
oCompanyInfo.UserDefinedFields
Dim oCompanyInfoUserDefinedItem
Dim nCount : nCount = 0

For nCount = 1 To oListCompanyInfoUserDefined.Count

    Set oCompanyInfoUserDefinedItem = oListCompanyInfoUserDefined.Item(nCount)

    If (oCompanyInfoUserDefinedItem.Type = 2) Then ' Wenn es eine Textinformation
ist
        Call dicCompanyInfo.Add(oCompanyInfoUserDefinedItem.Key,
oCompanyInfoUserDefinedItem.Value)
    End If

    Set oCompanyInfoUserDefinedItem = Nothing

Next

Set oListCompanyInfoUserDefined = Nothing

Dim sFullCompanyInfo : sFullCompanyInfo = ""
Dim dicItem

For Each dicItem In dicCompanyInfo
    If (Len(dicCompanyInfo(dicItem)) > 0) Then
        sFullCompanyInfo = sFullCompanyInfo & dicItem & ": " &
dicCompanyInfo(dicItem) & vbCrlf
    End If
Next

WScript.ClipboardText = sFullCompanyInfo
```

```
Call cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die
Zwischenablage übernommen.", "CompanyInfo", vbOkOnly)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
// Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("AccountNo", companyInfo.AccountNo);
dictionaryCompanyInfo.Add("Bank", companyInfo.Bank);
dictionaryCompanyInfo.Add("BankCode", companyInfo.BankCode);
dictionaryCompanyInfo.Add("City", companyInfo.City);
dictionaryCompanyInfo.Add("Company", companyInfo.Company);
dictionaryCompanyInfo.Add("Company2", companyInfo.Company2);
dictionaryCompanyInfo.Add("Company3", companyInfo.Company3);
dictionaryCompanyInfo.Add("Country", companyInfo.Country);
dictionaryCompanyInfo.Add("Email", companyInfo.Email);
dictionaryCompanyInfo.Add("Extra1", companyInfo.Extra1);
dictionaryCompanyInfo.Add("Extra2", companyInfo.Extra2);
dictionaryCompanyInfo.Add("Fax", companyInfo.Fax);
dictionaryCompanyInfo.Add("IBAN", companyInfo.IBAN);
dictionaryCompanyInfo.Add("Internet", companyInfo.Internet);
dictionaryCompanyInfo.Add("Logo", companyInfo.Logo);
dictionaryCompanyInfo.Add("Phone", companyInfo.Phone);
dictionaryCompanyInfo.Add("Street", companyInfo.Street);
dictionaryCompanyInfo.Add("VatID", companyInfo.VatID);
dictionaryCompanyInfo.Add("ZIP", companyInfo.ZIP);

ListCompanyInfoUserDefined listCompanyInfoUserDefined =
companyInfo.UserDefinedFields;

foreach (CompanyInfoUserDefinedItem companyInfoUserDefinedItem in
listCompanyInfoUserDefined)
{
    if (companyInfoUserDefinedItem.Type == 2)
    {
        dictionaryCompanyInfo.Add(companyInfoUserDefinedItem.Key,
companyInfoUserDefinedItem.Value);
    }
}

string fullCompanyInfo = string.Empty;

foreach (var dictionaryItem in dictionaryCompanyInfo)
{
    if (dictionaryItem.Value != "")
    {
        fullCompanyInfo = fullCompanyInfo + dictionaryItem.Key + ": " +
dictionaryItem.Value + System.Environment.NewLine;
    }
}

WScript.ClipboardText = fullCompanyInfo;
cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die Zwischenablage
übernommen.", "CompanyInfo", 0);

companyInfo.Dispose();
```

Type, read-only

Beschreibung:

Liefert den Feldtyp zurück (LL_Text).

Typ:

Long

Beispiel VBScript:

```
' Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("AccountNo", oCompanyInfo.AccountNo)
Call dicCompanyInfo.Add("Bank", oCompanyInfo.Bank)
Call dicCompanyInfo.Add("BankCode", oCompanyInfo.BankCode)
Call dicCompanyInfo.Add("City", oCompanyInfo.City)
Call dicCompanyInfo.Add("Company", oCompanyInfo.Company)
Call dicCompanyInfo.Add("Company2", oCompanyInfo.Company2)
Call dicCompanyInfo.Add("Company3", oCompanyInfo.Company3)
Call dicCompanyInfo.Add("Country", oCompanyInfo.Country)
Call dicCompanyInfo.Add("Email", oCompanyInfo.Email)
Call dicCompanyInfo.Add("Extra1", oCompanyInfo.Extra1)
Call dicCompanyInfo.Add("Extra2", oCompanyInfo.Extra2)
Call dicCompanyInfo.Add("Fax", oCompanyInfo.Fax)
Call dicCompanyInfo.Add("IBAN", oCompanyInfo.IBAN)
Call dicCompanyInfo.Add("Internet", oCompanyInfo.Internet)
Call dicCompanyInfo.Add("Logo", oCompanyInfo.Logo)
Call dicCompanyInfo.Add("Phone", oCompanyInfo.Phone)
Call dicCompanyInfo.Add("Street", oCompanyInfo.Street)
Call dicCompanyInfo.Add("VatID", oCompanyInfo.VatID)
Call dicCompanyInfo.Add("ZIP", oCompanyInfo.ZIP)

Dim oListCompanyInfoUserDefined : Set oListCompanyInfoUserDefined =
oCompanyInfo.UserDefinedFields
Dim oCompanyInfoUserDefinedItem
Dim nCount : nCount = 0

For nCount = 1 To oListCompanyInfoUserDefined.Count

    Set oCompanyInfoUserDefinedItem = oListCompanyInfoUserDefined.Item(nCount)

    If (oCompanyInfoUserDefinedItem.Type = 2) Then ' Wenn es eine Textinformation
ist
        Call dicCompanyInfo.Add(oCompanyInfoUserDefinedItem.Key,
oCompanyInfoUserDefinedItem.Value)
    End If

    Set oCompanyInfoUserDefinedItem = Nothing

Next

Set oListCompanyInfoUserDefined = Nothing

Dim sFullCompanyInfo : sFullCompanyInfo = ""
Dim dicItem

For Each dicItem In dicCompanyInfo
    If (Len(dicCompanyInfo(dicItem)) > 0) Then
        sFullCompanyInfo = sFullCompanyInfo & dicItem & ": " &
dicCompanyInfo(dicItem) & vbCrlf
    End If
Next

WScript.ClipboardText = sFullCompanyInfo

Call cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die
Zwischenablage übernommen.", "CompanyInfo", vbOkOnly)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
// Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("AccountNo", companyInfo.AccountNo);
dictionaryCompanyInfo.Add("Bank", companyInfo.Bank);
dictionaryCompanyInfo.Add("BankCode", companyInfo.BankCode);
dictionaryCompanyInfo.Add("City", companyInfo.City);
dictionaryCompanyInfo.Add("Company", companyInfo.Company);
dictionaryCompanyInfo.Add("Company2", companyInfo.Company2);
dictionaryCompanyInfo.Add("Company3", companyInfo.Company3);
dictionaryCompanyInfo.Add("Country", companyInfo.Country);
dictionaryCompanyInfo.Add("Email", companyInfo.Email);
dictionaryCompanyInfo.Add("Extra1", companyInfo.Extra1);
dictionaryCompanyInfo.Add("Extra2", companyInfo.Extra2);
dictionaryCompanyInfo.Add("Fax", companyInfo.Fax);
dictionaryCompanyInfo.Add("IBAN", companyInfo.IBAN);
dictionaryCompanyInfo.Add("Internet", companyInfo.Internet);
dictionaryCompanyInfo.Add("Logo", companyInfo.Logo);
dictionaryCompanyInfo.Add("Phone", companyInfo.Phone);
dictionaryCompanyInfo.Add("Street", companyInfo.Street);
dictionaryCompanyInfo.Add("VatID", companyInfo.VatID);
dictionaryCompanyInfo.Add("ZIP", companyInfo.ZIP);

ListCompanyInfoUserDefined listCompanyInfoUserDefined =
companyInfo.UserDefinedFields;

foreach (CompanyInfoUserDefinedItem companyInfoUserDefinedItem in
listCompanyInfoUserDefined)
{
    if (companyInfoUserDefinedItem.Type == 2)
    {
        dictionaryCompanyInfo.Add(companyInfoUserDefinedItem.Key,
companyInfoUserDefinedItem.Value);
    }
}

string fullCompanyInfo = string.Empty;

foreach (var dictionaryItem in dictionaryCompanyInfo)
{
    if (dictionaryItem.Value != "")
    {
        fullCompanyInfo = fullCompanyInfo + dictionaryItem.Key + ": " +
dictionaryItem.Value + System.Environment.NewLine;
    }
}

WScript.ClipboardText = fullCompanyInfo;
cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die Zwischenablage
übernommen.", "CompanyInfo", 0);

companyInfo.Dispose();
```

3.14 Container Objekt

Bietet Zugriff auf einen Container.

3.14.1 Methoden

CurrentRecord

Beschreibung:

Liefert den aktuellen Datensatz als Objekt vom Typ **Record** zurück.

Rückgabewert:

Record

Beispiel VBScript:

```
' Gibt den Kommentar eines Aktivitäten-Datensatzes aus. Basis ist hierbei der
Aktivitäten-Container der Kontakte-Ansicht einer combit_Large-Solution.

Dim oContainers : Set oContainers =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers
Dim oContainer : Set oContainer =
oContainers.ItemByName("ID.Aktivitäten.ContactID#{B3C0768A-5599-44B5-B4F2-
7D31A6C10EC5}")

Dim oCurrentRecord : Set oCurrentRecord = oContainer.CurrentRecord

Dim sComment : sComment = CStr(oCurrentRecord.GetContentsByName("Comment"))
Call cRM.DialogMessageBox("Kommentar: " & sComment, "Container.CurrentRecord",
vbOkOnly)

Set oCurrentRecord = Nothing
Set oContainer = Nothing
Set oContainers = Nothing
```

Beispiel C#-Script:

```
// Gibt den Kommentar eines Aktivitäten-Datensatzes aus. Basis ist hierbei der
Aktivitäten-Container der Kontakte-Ansicht einer combit_Large-Solution.

ListContainers containers =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers;
Container container = containers.ItemByName("ID.Aktivitäten.ContactID#{ADD84570-
956B-4079-8DE4-2B992DB3AEFE}");
Record containerRecord = container.CurrentRecord;
string comment = containerRecord.GetContentsByName("Comment");

cRM.DialogMessageBox("Kommentar: " + comment, "Container.CurrentRecord", 0);

containerRecord.Dispose();
container.Dispose();
containers.Dispose();
```

CurrentRecordSetCopy

Beschreibung:

Liefert das aktuelle **RecordSet** für einen Container zurück. Hierbei werden etwaig angewandte Container-Filter und aktive Sortierungen beachtet. Werden im Container keine Datensätze angezeigt, liefert die Methode Null bzw. Nothing zurück.

Hinweis: Wir empfehlen, nach Erzeugung eines **RecordSet**-Objektes zunächst mittles Aufruf der Methode "MoveFirst" die Existenz mindestens eines **Record**-Objektes zu überprüfen.

Wichtiger Hinweis für die Verwendung des Parameters CursorModel mit dem Wert 2 (forward-only) unter Microsoft SQL Server: Die Datensätze eines forward-only-RecordSets müssen nach dessen Erstellung direkt und unmittelbar über eine "GotoNext"-Schleife ohne Interaktion vollständig durchlaufen werden. Anderenfalls kann es, wenn das RecordSet viele Zeilen enthält, am Datenbankserver zu einem *ASYNC_NETWORK_IO*-Wartezustand kommen, der dann andere Abfragen (vor allem Änderungen) auf dieselbe Tabelle blockiert.

Parameter:

Parametername	Typ	Beschreibung

CursorModel	Long	Optional.
		Ermöglicht die Spezifikation des Datenbankcursormodells, das für den zurückgegebenen **RecordSet** genutzt werden soll.
		
		Werte:
		0 (Standardwert): Erzeugt ein **RecordSet** mit einem Datenbankcursormodell, welches innerhalb der combit CRM-Projektdatei spezifiziert werden kann:
		
		`...`
		`<!-- DATA -->`
		`<profile>`
		`<list name="">`
		`<list name="ExtendedSettings">`
		`<item name="COMRecordSetCursorDefault">2</item>`
		`</list>`
		`...`
		
		Wird in der combit CRM-Projektdatei keine Eigenschaft **COMRecordSetCursorDefault** gefunden, so wird immer ein fully-dynamic **RecordSet** erzeugt. Mögliche Werte für die Eigenschaft sind: 1 – fully-dynamic RecordSet, 2 – forward-only RecordSet.
		
		1: Erzeugt ein **RecordSet** mit fully-dynamic Datenbankcursor.
		
		2: Erzeugt ein **RecordSet** mit forward-only Datenbankcursor. Ermöglicht deutliche Performance-Gewinne, insbesondere bei großen Datenmengen und komplexen Filterausdrücken, erlaubt aber lediglich das einmalige Durchlaufen in Vorwärtsrichtung durch den **RecordSet**.
		
		Die Methoden **RecordSet.DialogSelectRecord**, **RecordSet.DialogSelectRecordMultiple**, **RecordSet.SendBulkMail** (bei anzuzeigendem integrierten Mail-Editor), **RecordSet.MovePrevious**, **RecordSet.MoveLast**, **InputForm.DialogSelectRecordDropDown** werden einen Scriptfehler werfen, wenn diese für einen forward-only **RecordSet** genutzt werden. Für diese Methoden muss der **RecordSet** explizit ohne forward-only (Werte 0 oder 1) erzeugt werden.

Rückgabewert:

RecordSet oder Null/Nothing

Beispiel VBScript:

```
' Gibt die Kommentare mehrerer Aktivitäten-Datensätze aus. Basis ist hierbei der
Aktivitäten-Container der Kontakte-Ansicht einer combit_Large-Solution.

Dim oContainer : Set oContainer =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-2B992DB3AEFE}")
Dim oCurrentRecordSet, oCurrentRecord
Set oCurrentRecordSet = oContainer.CurrentRecordSetCopy

If oCurrentRecordSet.MoveFirst Then

    Set oCurrentRecord = oCurrentRecordSet.CurrentRecord
    Dim sComment : sComment = ""
    Dim nCount : nCount = 0

    Do
        sComment = sComment & vbCrlf & vbCrlf &
CStr(oCurrentRecord.GetContentsByName("Comment"))
        nCount = nCount + 1
    Loop Until Not oCurrentRecordSet.MoveNext

    Set oCurrentRecord = Nothing
    Call cRM.DialogMessageBox("Kommentar: " & sComment & vbCrlf & nCount,
"Container.CurrentRecordSetCopy", vbOkOnly)

End If
Set oCurrentRecordSet = Nothing
Set oContainer = Nothing
```

Beispiel C#-Script:

```
// Gibt die Kommentare mehrerer Aktivitäten-Datensätze aus. Basis ist hierbei der
Aktivitäten-Container der Kontakte-Ansicht einer combit_Large-Solution.

ListContainers containers =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers;
Container container = containers.ItemByName("ID.Aktivitäten.ContactID#{ADD84570-
956B-4079-8DE4-2B992DB3AEFE}");
RecordSet containerRecordSet = container.CurrentRecordSetCopy();
Record containerRecord;

if (containerRecordSet.MoveFirst())
{
    containerRecord = containerRecordSet.CurrentRecord;
    string comment = string.Empty;
    int count = 0;

    do
    {
        comment = comment + "\r\n" + "\r\n" +
containerRecord.GetContentsByName("Comment");
        count++;
    } while (containerRecordSet.MoveNext() == false);

    containerRecord.Dispose();

    cRM.DialogMessageBox("Kommentar: " + comment + "\r\n" + count,
"Container.CurrentRecordSetCopy", 0);
}

containerRecordSet.Dispose();
```

InvokeDataContextMenu

Beschreibung:

Führt den übergebenen Kontextmenübefehl eines Rechtsklick auf den aktuell gewählten Container-Datensatz und für das angegebene Feld (insofern die Menü-ID einen Bezug zu einem Feld erfordert) aus.

Parameter:

Parametername	Typ	Beschreibung
MenuID	Long	Menü-ID des Kontextmenübefehls.
FieldName	String	Name des gewünschten Feldes. Sollte kein Feld verwendet werden, kann der Parameterwert mit einer leeren Zeichenkette übergeben werden.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-
2B992DB3AEFE}").InvokeDataContextMenu(32858, "Responsible")  ' Den Feldinhalt von
"Verantwortlich" des aktuell ausgewählten Datensatzes in die Zwischenablage
kopieren
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-
2B992DB3AEFE}").InvokeDataContextMenu(33034, "Document_Embedded") ' Das aktuell
ausgewählte eingebettete Dokument mit der verknüpften Anwendung öffnen
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-
2B992DB3AEFE}").InvokeDataContextMenu(32903, "") ' Neues Dokument über die
Dokumentenverwaltung anlegen
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-
2B992DB3AEFE}").InvokeDataContextMenu(32858, "Responsible");  // Den Feldinhalt
von "Verantwortlich" des aktuell ausgewählten Datensatzes in die Zwischenablage
kopieren
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID. Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-
2B992DB3AEFE}").InvokeDataContextMenu(33034, "Document_Embedded"); // Das aktuell
ausgewählte eingebettete Dokument mit der verknüpften Anwendung öffnen
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID. Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-
2B992DB3AEFE}").InvokeDataContextMenu(32903, ""); // Neues Dokument über die
Dokumentenverwaltung anlegen
```

InvokeTitleContextMenu

Beschreibung:

Führt den übergebenen Kontextmenübefehl eines Rechtsklick auf die aktuell dargestellte Container-Titelzeile aus.

Parameter:

Parametername	Typ	Beschreibung
MenuID	Long	Menü-ID des Kontextmenübefehls.
FieldName	String	Name des gewünschten Feldes. Sollte kein Feld verwendet werden, kann der Parameterwert mit einer leeren Zeichenkette übergeben werden.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-
2B992DB3AEFE}").InvokeTitleContextMenu(33094, "") ' Profile verwalten
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID. Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-
2B992DB3AEFE}").InvokeTitleContextMenu(33094, ""); // Profile verwalten
```

SetFilterByName

Beschreibung:

Aktiviert einen der verfügbaren Containerfilter, welche in der Containerfilter-Auswahlliste angeboten werden. Voraussetzung: Es gibt eine derartige Containerfilter-Auswahlliste als Element der Eingabemaske. Der Filter muss ein Containerfilter sein. Der zu übergebene Name ist der in der Filtereigenschaft "Name für Scripte/Workflows" hinterlegte Wert.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name für Scripte/Workflows des gewünschten Filteraus-drucks.

Rückgabewert:

Long

Wert	Beschreibung
1	Filter konnte ausgeführt werden. Beachten Sie bitte, dass die Methode auch dann einen Wert 1 zurückliefert, wenn der Filter-ausdruck einen Fehler beinhaltet (dieser wird visuell angezeigt) oder wenn bei einer Benutzereingabe auf „Abbrechen“ geklickt wurde.
-1	Es konnte kein Filterausdruck mit dem übergebenen Namen ge-funden werden.
0	Der übergebene Container ist nicht sichtbar oder konnte nicht gefunden werden.

Beispiel VBScript:

```
<!--#pragma keepeditmode-->
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-
2B992DB3AEFE}").SetFilterByName("Alle Briefe")
```

Beispiel C#-Script:

```
// < !--#pragma keepeditmode-->
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-
2B992DB3AEFE}").SetFilterByName("Alle Briefe");
```

Update

Beschreibung:

Aktualisiert den Container.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-2B992DB3AEFE}").Update()
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.ContactID#{ADD84570-956B-4079-8DE4-2B992DB3AEFE}").Update();
```

3.15 DataCollection Objekt

Das **DataCollection**-Objekt beinhaltet Eigenschaften und Methoden, um durch eine Liste aus Datensätzen navigieren zu können. Jeder Datensatz wird durch ein **DataItem**-Objekt repräsentiert.

3.15.1 Eigenschaften

Count, read-only

Beschreibung:

Gibt die Anzahl der Datensätze zurück.

Typ:

Long

Beispiel VBScript:

```
Dim nCountDataCollection : nCountDataCollection = oDataCollection.Count
```

Beispiel C#-Script:

```
long countDataCollection = dataCollection.Count;
```

CurrentItem

Beschreibung:

Liefert den aktuellen Datensatz als Objekt vom Typ **DataItem** zurück. Wenn z. B. keine Datensätze im **DataCollection** enthalten sind, liefert diese Methode kein **DataItem** Objekt zurück.

> **Wichtig**: Das erzeugte **DataItem** aktualisiert sich immer automatisch, wenn anschließend **Move**...-Befehle für die zugehörige **DataCollection** aufgerufen werden.
>
> Mit **CurrentItem** erzeugte **DataItem**-Objekte können dadurch jedoch nicht als Variablen für unterschiedliche Datensätze benutzt werden!

Rückgabewert:

DataItem

Beispiel VBScript:

```
Dim oDataItem : Set oDataItem = oDataCollection.CurrentItem()
```

Beispiel C#-Script:

```
DataItem dataItem = dataCollection.CurrentItem();
```

3.15.2 Methoden

MoveFirst

Beschreibung:

Bewegt den Datensatz-Zeiger auf den Anfang der **DataCollection**.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim bReturn : bReturn = oDataCollection.MoveFirst()
```

Beispiel C#-Script:

```
bool return = dataCollection.MoveFirst();
```

MoveLast

Beschreibung:

Bewegt den Datensatz-Zeiger auf das Ende der **DataCollection**.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim bReturn : bReturn = oDataCollection.MoveLast()
```

Beispiel C#-Script:

```
bool return = dataCollection.MoveLast();
```

MoveNext

Beschreibung:

Bewegt den Datensatz-Zeiger um einen Datensatz vorwärts.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim bReturn : bReturn = oDataCollection.MoveNext()
```

Beispiel C#-Script:

```
bool return = dataCollection.MoveNext();
```

MovePrevious

Beschreibung:

Bewegt den Datensatz-Zeiger um einen Datensatz rückwärts.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim bReturn : bReturn = oDataCollection.MovePrevious()
```

Beispiel C#-Script:

```
bool return = dataCollection.MovePrevious();
```

3.16 Dataltem Objekt

Das **Dataltem**-Objekt beinhaltet Eigenschaften und Methoden, um auf spezifische Felder des Datensatzes lesend zugreifen zu können. Welche Felder dabei ausgelesen werden können, wird beim jeweiligen Erzeugen des Objektes dokumentiert.

3.16.1 Methoden

GetContentsValueByName

Beschreibung:

Liefert den Inhalt entsprechend des Feldtyps des Feldes zurück, dessen Feldname übergeben wurde.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name des gewünschten Feldes. Wichtig: Sollen Felder aus einer Empfänger-Liste abgefragt werden (siehe **GetAllRecipients, GetUnsubscribers** oder **GetSubscribers** des **EmailToolRecipientList**-Objektes), so müssen hier die originalen Feldnamen aus der Empfänger-Liste des EmailTools verwendet werden. Diese können meist in den Einstellungen der Empfänger-Liste des EmailTools selbst eingesehen werden. Es muss hierbei auch auf die korrekte Schreibweise geachtet werden.

Beispiel VBScript:

```
Dim Value
Value = DataItem.GetContentsValueByName("MyFieldName")
If (VarType(Value) = vbNULL) Then
    MsgBox "Das Feld ist unbekannt!"
End If
```

Beispiel C#-Script:

```
var value = dataItem.GetContentsValueByName("MyFieldName");
```

Rückgabewert:

Variant

> **Hinweis:** Wird ein unbekannter oder nichtexistierender Feldname abgefragt, wird NULL zurückgeliefert.

3.17 DialogForm Objekt

Mit diesem Objekt können Sie eigene Dialoge gestalten und per Script darauf zugreifen.

> **Wichtig**: Dieses Objekt steht ab der Professional-Edition zur Verfügung.

Beispiel VBScript:

```
Option Explicit
Dim oDlgObj
Set oDlgObj = cRM.DialogForm
WScript.ConnectObject oDlgObj,"DIALOG_"

oDlgObj.DialogTitle = "DialogForm"

oDlgObj.DefineVariableStart
oDlgObj.DefineVariable "Var1", 0, "C", 20, "0", ""
```

```
oDlgObj.DefineVariable "Var2", 0, "N", 20, "3", ""

oDlgObj.DefineFctStart
oDlgObj.DefineFct "OK", "Bestätigen", True, False, vbOK
oDlgObj.DefineFct "Abbrechen", "Beenden", True, False, vbCancel
oDlgObj.DefineFct "Event", "Event", False, True, -1

oDlgObj.DLIPath = " %PRJDIR%\Test.dli"
oDlgObj.Sizable = True
oDlgObj.SizeToContent = False
oDlgObj.WidthInPixel = 640
oDlgObj.HeightInPixel = 480
oDlgObj.Show

Dim contents
contents = ""
oDlgObj.GetVariableContentsVariant "Var2", contents
MsgBox "Var2: " & contents
oDlgObj.GetVariableContentsVariant "Var1", contents
MsgBox "Var1: " & contents

' Ereignis, bei dem CauseCallback = true aufgerufen wird
Sub DIALOG_ExecuteFct(sName)
    MsgBox "Event: " & sName
End Sub
MsgBox "Ende"

WScript.DisconnectObject oDlgObj
Set oDlgObj = Nothing
```

Beispiel C#-Script:

```
public static void Main()
{
    long OK = 1;
    long CANCEL = 2;
    long ABORT = 3;
    long RETRY = 4;
    long IGNORE = 5;
    long YES = 6;
    long NO = 7;

    DialogForm dlgObj = cRM.DialogForm;
    dlgObj.DialogTitle = "DialogForm";

    dlgObj.DefineVariableStart();
    dlgObj.DefineVariable("Var1", 0, "C", 20, 0, "");
    dlgObj.DefineVariable("Var2", 0, "N", 20, 3, "");

    dlgObj.DefineFctStart();
    dlgObj.DefineFct("OK", "Bestätigen", true, false, OK);
    dlgObj.BindFctHandler("OK", ButtonOK);
    dlgObj.DefineFct("Abbrechen", "Beenden", true, false, CANCEL);
    dlgObj.BindFctHandler("Abbrechen", ButtonCancel);
    dlgObj.DefineFct("Event", "Event", false, true, -1);
    dlgObj.BindFctHandler("Event", ButtonEvent);

    dlgObj.DLIPath = @"%PRJDIR%\Test.dli";
    dlgObj.Sizable = true;
    dlgObj.SizeToContent = false;
    dlgObj.WidthInPixel = 640;
    dlgObj.HeightInPixel = 480;
    dlgObj.Show();

    object contents = null;
    dlgObj.GetVariableContentsVariant("Var2", ref contents);
    MessageBox.Show("Var2: " + contents.ToString(), cRM.AppTitle,
MessageBoxButtons.OK);
    dlgObj.GetVariableContentsVariant("Var1", ref contents);
    MessageBox.Show("Var1: " + contents.ToString(), cRM.AppTitle,
MessageBoxButtons.OK);
```

```
}
public static void ButtonOK()
{
    MessageBox.Show("Die Schaltfläche \"OK\" wurde betätigt.", cRM.AppTitle,
MessageBoxButtons.OK);
}
public static void ButtonCancel()
{
    MessageBox.Show("Die Schaltfläche \"Abbrechen\" wurde betätigt.",
cRM.AppTitle, MessageBoxButtons.OK);
}
public static void ButtonEvent()
{
    MessageBox.Show("Die Schaltfläche \"Event\" wurde betätigt.", cRM.AppTitle,
MessageBoxButtons.OK);
}
```

3.17.1 Eigenschaften

DialogTitle

Beschreibung:

Setzt den Dialogtitel oder liefert ihn zurück.

Typ:

String

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

DLIPath

Beschreibung:

Setzt den Pfad inkl. Dateiname zur DLI-Eingabemaskendatei oder liefert ihn zurück.

Wichtig: Um die DLI-Eingabemaskendatei erstellen zu können, muss bereits vor dem ersten Start des Scripts ein Pfad inkl. Dateiname angegeben werden.

Für den Pfad der DLI-Eingabemaskendatei können auch die Variablen %APPDIR% bzw. %PRJDIR% verwendet werden, z. B. %PRJDIR&\Scripts\Questionnaire.dli.

Hinweise zum Aufruf des Eingabemaskendesigners finden Sie in der Methode Layout. Die DLI-Eingabemaskendatei muss nach der Gestaltung abschließend gespeichert werden.

Typ:

String

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

IconPath

Beschreibung:

Setzt oder liefert das Icon für den Dialog.

Wichtig: Es kann der Pfad zu einer .ico-, .exe- oder .dll-Datei verwendet werden. Dabei können auch die Variablen %APPDIR% bzw. %PRJDIR% verwendet werden, z. B. %APPDIR%\Scripts\combit.ico. Es werden nur 16x16 Pixel große Icons unterstützt. Bei .exe- und .dll-Dateien wird immer das erste Icon der Sammlung verwendet.

Typ:

String

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

HeightInPixel

Beschreibung:

Setzt die Höhe des Dialogs in Pixel (maximal 32767 Pixel) oder liefert sie zurück.

Typ:

Short

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

ShowResult

Beschreibung:

Gibt den in der Methode DefineFct übergebenen Wert des Parameters RetValOnClose zurück, insofern der Dialog mit Klick auf diese Schaltfläche geschlossen wird, siehe Parameter CauseCloseDialog.

Typ:

Variant

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

Sizable

Beschreibung:

Bestimmt, ob sich die Größe des Dialogs verändern lässt oder nicht bzw. liefert diesen Wert zurück.

Typ:

Bool

Wert	Beschreibung
True	Größe veränderbar
False	Größe nicht veränderbar

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

SizeToContent

Beschreibung:

Bestimmt, ob sich die Größe des Dialogs an die Größe der Eingabemaske anpasst oder nicht bzw. liefert diesen Wert zurück.

Typ:

Bool

Wert	Beschreibung
True	Die Größe des Dialogs wird an die Eingabemaske angepasst. Dabei werden die Eigenschaften HeightInPixel und WidthInPixel ignoriert und die Standardgröße 330x173 Pixel verwendet. Wenn Sie die Eingabemaskengröße ändern möchten, können Sie die Pixel direkt in der dli-Datei angeben: (...) Type=CARD Rect(l,t,w,h)=(330,173,1,1) (...)
False	Größe wird nicht an Eingabemaske angepasst

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

WidthInPixel

Beschreibung:

Setzt die Breite des Dialogs in Pixel (maximal 32767 Pixel) oder liefert sie zurück.

Typ:

Short

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

3.17.2 Methoden

AddTimer

Beschreibung:

Meldet einen Timer-Event an. Dieser meldet sich in den in Parameter **Seconds** angegebenen Abständen über den Event **TimerEvent**.

Parameter:

Parametername	Typ	Beschreibung
TimerName	String	Name des Timers.
Seconds	Long	Intervall in Sekunden, in denen sich der Event meldet.

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **DialogForm Objekt**.

BindFctHandler [C#]

Beschreibung:

Diese Methode steht nur bei C#-Scripten zur Verfügung.

Bindet eine Methode (delegate) an einen Buttonklick. Der erste Parameter muss hierbei dem ersten Paramenter der Funktion **DefineFct** entsprechen.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name des Buttons an den die Methode gebunden werden soll.
FctEventHandler	Delegate	Eine Methode mit der Signatur *Methode()* und keinem Rückgabewert (void).

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **DialogForm Objekt**.

BindTimerHandler [C#]

Beschreibung:

Diese Methode steht nur bei C#-Scripten zur Verfügung.

Bindet eine Methode (delegate) an einen Timer. Der erste Parameter muss hierbei dem ersten Paramenter der Funktion **AddTimer** entsprechen.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name des Timers an den die Methode gebunden werden soll.
TimerEventHandler	Delegate	Eine Methode mit der Signatur *Methode()* und keinem Rückgabewert (void).

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **DialogForm Objekt**.

Close

Beschreibung:

Schließt den Dialog.

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

DefineComboboxItems

Beschreibung:

Fügt zu einer in der Eingabemaske bestehenden Combobox Auswahleinträge hinzu.

Parameter:

Parametername	Typ	Beschreibung
FieldName	String	Feldname.
ComboboxItems	String	Liste mit Einträgen. Mehrere Einträge werden dabei mit dem Separator 'vbCr' (Char(13)) getrennt.

Rückgabewert:

Long

Wert	Beschreibung
0	Erfolgreich
1	Fehler

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

DefineFct

Beschreibung:

Meldet einen Button an.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Anzeigename.
Tooltip	String	Tooltip.
CauseCloseDialog	Bool	True: Der Dialog wird beim Drücken geschlossen (z. B. bei OK oder Abbrechen).
CauseCallback	Bool	True: Der Event ExecuteFct wird beim Drücken aufgerufen.
RetValOnClose	Long	Rückgabewert, den die ShowResult Eigenschaft später liefern soll, wenn über die Schaltfläche der Dialog geschlossen wird (wenn Parameter CauseCloseDialog =True).

Hinweis: Wird der Dialog durch einen Button beendet (siehe Parameter CauseCloseDialog), werden, falls die Button-ID nicht vbCancel (2) ist, vor dem Schließen alle Eingaberegeln geprüft und, falls diese erzwungen werden, das Schließen des Dialogs verhindert.

Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

DefineFctStart

Beschreibung:

Initialisiert die Funktionen.

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **DialogForm Objekt**.

DefineVariable

Beschreibung:

Meldet ein Eingabefeld an.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Anzeigename.
ID	Long	Eindeutige Nummer für die anzulegende Variable, wird derzeit nur intern verwendet. Empfohlener Wert: 0
Type	String	Feldtyp. Mögliche Werte: "C" für Zeichenfelder "N" für Numerische Felder "D" für Datumsfelder "T" für Datum+Zeit Felder "L" für Boolesche Felder "M" für Bemerkungsfelder
Length	Long	Feldlänge für numerische Felder. Werden andere Feldtypen verwendet, sollte der Wert auf 0 gesetzt werden.
Precision	Long	Präzision bei Nachkommastellen für numerische Felder. Werden andere Feldtypen verwendet, sollte der Wert auf 0 gesetzt werden.
DefaultValue	String	Initialer Wert, verwenden Sie die folgenden Werte/Formate: Bei Datumsfeldern: YYYYMMDD Bei Datumsfeldern mit Zeitanteil: YYYYMMDDHHMMSS Bei Booleschen Feldern: "T" (true/wahr) oder "F" (false/falsch)

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **DialogForm Objekt**.

DefineVariableStart

Beschreibung:

Initialisiert die Variablen.

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **DialogForm Objekt**.

GetVariableContents

Beschreibung:

Liefert den aktuellen Inhalt einer Variablen.

Hinweis: Diese Methode sollte in VBScript/Jscript nicht verwendet werden, da diese keine Stringreferenzen unterstützen und der zurückgegebene Wert daher entweder fehlerhaft ist oder ein Typenkonflikt entsteht. Verwenden Sie stattdessen die Methode 'GetVariableContentsVariant'.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Anzeigename.
Contents	String (Referenz)	Rückgabewert (Inhalt der Variable).

Rückgabewert:

Long

Wert	Beschreibung
0	Erfolgreich
1	Fehler

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

GetVariableContentsVariant

Beschreibung:

Liefert den aktuellen Inhalt einer Variablen.

Hinweis: Diese Methode hat die gleiche Funktion wie die Methode 'GetVariableContents', sollte dieser aber bei Verwendung in VBScript/Jscript vorgezogen werden, um fehlerhafte Ergebnisse oder Typenkonflikte zu vermeiden.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Anzeigename.
Contents	Variant (Referenz)	Rückgabewert (Inhalt der Variable).

Rückgabewert:

Long

Wert	Beschreibung
0	Erfolgreich
1	Fehler

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

Layout

Beschreibung:

Öffnen des Dialoges im Eingabemaskendesigner.

Dieser kann auch durch Drücken der Tastenkombination STRG+J direkt nach dem Öffnen des Dialogs erreicht werden (nur wenn die Methode Show verwendet wird, nicht aber bei Methode ShowModeless).

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

SetFocusToField

Beschreibung:

Setzt den Eingabecursor in ein bestimmtes Feld und wechselt in den Bearbeitungsmodus.

Der typische Anwendungsfall ist die Benutzung im Rahmen eines Events. Falls mit dieser Methode der Fokus für die initielle Anzeige gesetzt werden soll, dann muss dies über einen einmalig ausgeführten Timer (siehe Event TimerEvent) realisiert werden, da ein Methodenaufruf vor Show zu früh ist.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Anzeigename.
SelectText	Bool	True: Der bereits bestehende Text des Feldes wird selektiert.

Rückgabewert:

Long

Wert	Beschreibung
0	Erfolgreich
1	Fehler

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

SetForeground

Beschreibung:

Bringt den Dialog in den Vordergrund.

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

SetVariableContents

Beschreibung:

Setzt den aktuellen Inhalt einer Variablen.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Anzeigename.
Contents	String	Neuer Inhalt der Variable.

		Verwenden Sie folgende Formate: Bei Datumsfeldern: YYYYMMDD Bei Datumsfeldern mit Zeitanteil: YYYYMMDDHHMMSS

Rückgabewert:

Long

Wert	Beschreibung
0	Erfolgreich
1	Fehler

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

Show

Beschreibung:

Stellt den Dialog im modalen Anzeigemodus dar. Dies verhindert Eingaben in combit CRM außerhalb des Dialogs.

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

ShowModeless

Beschreibung:

Stellt den Dialog im nicht modalen Anzeigemodus dar. Dies erlaubt die Benutzung der Anwendung (z. B. der Ansichten) während der Dialog geöffnet ist.

Hinweis: Die Methode kehrt sofort zurück, Anschließend MUSS das Script auf das Schließen des Dialoges durch den Benutzer aktiv warten:

```
oDlgObj.ShowModeless
While not WScript.Terminate and oDlgObj.IsShowing = 1
      WScript.Sleep(200)
Wend
```

Im Falle von WScript.Terminate muss sich das Script sofort beenden.

Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

Update

Beschreibung:

Aktualisiert die Ansicht der geladenen Eingabemaske. Dies kann nötig werden um bspw. Mit der Methode DefineComboboxItems neu hinzugefügte Comboboxeinträge zu aktualisieren.

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel DialogForm Objekt.

3.17.3 Events

ExecuteFct

Beschreibung:

Wird aufgerufen, wenn in einem Button-Click der Parameter **CauseCallback** auf True gesetzt wurde.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Anzeigename des Buttons.

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **DialogForm Objekt**.

ExecuteFctRightClick

Beschreibung:

Wird aufgerufen, wenn in einem Button-Click mit der rechten Maustaste der Parameter **CauseCallback** auf True gesetzt wurde.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Anzeigename des Buttons.

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **DialogForm Objekt**.

TimerEvent

Beschreibung:

Wird aufgerufen, wenn ein mit der Methode **AddTimer** gemeldeter Event abläuft.

Parameter:

Parametername	Typ	Beschreibung
TimerName	String	Name des Events.

Beispiel VBScript:

```
Option Explicit
Dim oDlgObj
Set oDlgObj = cRM.DialogForm
WScript.ConnectObject oDlgObj,"DIALOG_"

oDlgObj.DialogTitle = "DialogForm"

oDlgObj.DefineVariableStart
oDlgObj.DefineVariable "Var1", 0, "C", 20, "0", ""
oDlgObj.DefineVariable "Var2", 0, "N", 20, "3", ""

oDlgObj.DefineFctStart
oDlgObj.DefineFct "OK", "Bestätigen", True, False, vbOK
oDlgObj.DefineFct "Abbrechen", "Beenden", True, False, vbCancel
oDlgObj.DefineFct "Event", "Event", False, True, -1

oDlgObj.DLIPath = " %PRJDIR%\Test.dli"
oDlgObj.Sizable = True
oDlgObj.SizeToContent = False
oDlgObj.WidthInPixel = 640
oDlgObj.HeightInPixel = 480
oDlgObj.Show
```

```
Dim contents
contents = ""
oDlgObj.GetVariableContentsVariant "Var2", contents
MsgBox "Var2: " & contents
oDlgObj.GetVariableContentsVariant "Var1", contents
MsgBox "Var1: " & contents

' Ereignis, bei dem CauseCallback = true aufgerufen wird
Sub DIALOG_ExecuteFct(sName)
    MsgBox "Event: " & sName
End Sub
MsgBox "Ende"

WScript.DisconnectObject oDlgObj
Set oDlgObj = Nothing
```

Beispiel C#-Script:

```
public static void Main()
{
    long OK = 1;
    long CANCEL = 2;
    long ABORT = 3;
    long RETRY = 4;
    long IGNORE = 5;
    long YES = 6;
    long NO = 7;

    DialogForm dlgObj = cRM.DialogForm;
    dlgObj.DialogTitle = "DialogForm";

    dlgObj.DefineVariableStart();
    dlgObj.DefineVariable("Var1", 0, "C", 20, 0, "");
    dlgObj.DefineVariable("Var2", 0, "N", 20, 3, "");

    dlgObj.DefineFctStart();
    dlgObj.DefineFct("OK", "Bestätigen", true, false, OK);
    dlgObj.BindFctHandler("OK", ButtonOK);
    dlgObj.DefineFct("Abbrechen", "Beenden", true, false, CANCEL);
    dlgObj.BindFctHandler("Abbrechen", ButtonCancel);
    dlgObj.DefineFct("Event", "Event", false, true, -1);
    dlgObj.BindFctHandler("Event", ButtonEvent);

    dlgObj.DLIPath = @"%PRJDIR%\Test.dli";
    dlgObj.Sizable = true;
    dlgObj.SizeToContent = false;
    dlgObj.WidthInPixel = 640;
    dlgObj.HeightInPixel = 480;
    dlgObj.Show();

    object contents = null;
    dlgObj.GetVariableContentsVariant("Var2", ref contents);
    MessageBox.Show("Var2: " + contents.ToString(), cRM.AppTitle,
MessageBoxButtons.OK);
    dlgObj.GetVariableContentsVariant("Var1", ref contents);
    MessageBox.Show("Var1: " + contents.ToString(), cRM.AppTitle,
MessageBoxButtons.OK);
}
public static void ButtonOK()
{
    MessageBox.Show("Die Schaltfläche \"OK\" wurde betätigt.", cRM.AppTitle,
MessageBoxButtons.OK);
}
public static void ButtonCancel()
{
    MessageBox.Show("Die Schaltfläche \"Abbrechen\" wurde betätigt.",
cRM.AppTitle, MessageBoxButtons.OK);
}
public static void ButtonEvent()
{
```

```
    MessageBox.Show("Die Schaltfläche \"Event\" wurde betätigt.", cRM.AppTitle,
MessageBoxButtons.OK);
}
```

3.18 DocMngr Objekt

3.18.1 Eigenschaften

LastError, read-only

Beschreibung:

Liefert ein OLEError-Objekt zurück.

Typ:

OLEError

Wert	Beschreibung
32	Schreibzugriff verweigert.
33	Kopieren der Datei fehlgeschlagen.

Beispiel VBScript:

```
' Erstellt einen neuen Aktivitäten-Datensatz und fügt ein Dokument hinzu. Basis
ist hierbei der Aktivitäten-Container der Firmen-Ansicht einer combit_Large-
Solution

Dim oDocMngr : Set oDocMngr = cRM.CurrentProject.DocMngr
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Dim oRelation : Set oRelation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID")
Dim sFileToAppend : sFileToAppend = "C:\Firmen-Dossier.docx"
Dim sDocDescription : sDocDescription = "docx"

If (oDocMngr.AppendFile(oRecord, oRelation, sFileToAppend, sDocDescription) =
True) Then
    Call cRM.DialogMessageBox("Die Datei """ & sFileToAppend & """ konnte
erfolgreich hinzugefügt werden.", "DocMngr.AppendFile", vbOkOnly)
Else
    If (oDocMngr.LastError.ErrorCode = 32) Then
        Call cRM.DialogMessageBox("Das Hinzufgen der Datei """ & sFileToAppend &
""" konnte nicht erfolgreich durchgeführt werden. Der Schreibzugriff wurde
verweigert.", "DocMngr.AppendFile", vbOkOnly)
    ElseIf (oDocMngr.LastError.ErrorCode = 33) Then
        Call cRM.DialogMessageBox("Das Hinzufgen der Datei """ & sFileToAppend &
""" konnte nicht erfolgreich durchgeführt werden. Das Kopieren der Datei ist
fehlgeschlagen.", "DocMngr.AppendFile", vbOkOnly)
    End If
End If

Set oRelation = Nothing
Set oRecord = Nothing
Set oDocMngr = Nothing
```

Beispiel C#-Script:

```
// Erstellt einen neuen Aktivitäten-Datensatz und fügt ein Dokument hinzu. Basis
ist hierbei der Aktivitäten-Container der Firmen-Ansicht einer combit_Large-
Solution

DocMngr docMngr = cRM.CurrentProject.DocMngr;
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
```

```
Relation relation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID");
string fileToAppend = @"C:\Firmen-Dossier.docx";
string docDescription = "docx";

if (docMngr.AppendFile(record, relation, fileToAppend, docDescription) == true)
{
    cRM.DialogMessageBox("Die Datei \"" + fileToAppend + "\" konnte erfolgreich
hinzugefügt werden.", "DocMngr.AppendFile", 0);
}
else
{
    if (docMngr.LastError.ErrorCode == 32)
    {
        cRM.DialogMessageBox("Das Hinzufügen der Datei \"" + fileToAppend + "\"
konnte nicht erfolgreich durchgefhrt werden. Der Schreibzugriff wurde
verweigert.", "DocMngr.AppendFile", 0);
    }
    else if (docMngr.LastError.ErrorCode == 33)
    {
        cRM.DialogMessageBox("Das Hinzufügen der Datei \"" + fileToAppend + "\"
konnte nicht erfolgreich durchgefhrt werden. Das Kopieren der Datei ist
fehlgeschlagen.", "DocMngr.AppendFile", 0);
    }
}

relation.Dispose();
record.Dispose();
docMngr.Dispose();
```

3.18.2 Methoden

AppendFile

Beschreibung:

Fügt einen neuen relationalen Datensatz mit einer Datei hinzu. Die Datei wird in das erste gefundene Dokumentenfeld der relationalen Ansicht geschrieben. Alle weiteren Felder werden ignoriert.

Dokumentenfelder: (eingebettete) Datei, (eingebettete) Grafik oder DMS Dokument.

Parameter:

Parametername	Typ	Beschreibung
oRecord	Objekt	Datensatz-Objekt, zu dem ein relationaler Datensatz angehängt werden soll, z. B. *Firmen*
oRelation	Objekt	Relations-Objekt, in dem der neue relationale Datensatz mit der angehängten Datei eingefügt wird.
sFile	String	Datei, welche hinzugefügt werden soll (gesamter Pfad + Dateiname).
sDocDescriptionCfg	String	Dateiendung der zu verwendeten Dokumentenkonfiguration oder mit | getrennte Dokumentenkonfigurationsbezeichnung, welche in der Konfiguration der Dokumentenverwaltung angeben wurde, z. B. *msg* oder *msg|E-Mails*

Rückgabewert:

Bool

Wert	Beschreibung

True	Neuer relationaler Datensatz mit der eingefügten Datei konnte geschrieben werden.
False	Neuer relationaler Datensatz konnte nicht erstellt werden.

Beispiel VBScript:

```
' Erstellt einen neuen Aktivitäten-Datensatz und fügt ein Dokument hinzu. Basis
ist hierbei der Aktivitäten-Container der Firmen-Ansicht einer combit_Large-
Solution

Dim oDocMngr : Set oDocMngr = cRM.CurrentProject.DocMngr
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Dim oRelation : Set oRelation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID")
Dim sFileToAppend : sFileToAppend = "C:\Firmen-Dossier.docx"
Dim sDocDescription : sDocDescription = "docx"

If (oDocMngr.AppendFile(oRecord, oRelation, sFileToAppend, sDocDescription) =
True) Then
    Call cRM.DialogMessageBox("Die Datei """ & sFileToAppend & """ konnte
erfolgreich hinzugefügt werden.", "DocMngr.AppendFile", vbOkOnly)
Else
    If (oDocMngr.LastError.ErrorCode = 32) Then
        Call cRM.DialogMessageBox("Das Hinzufgen der Datei """ & sFileToAppend &
""" konnte nicht erfolgreich durchgeführt werden. Der Schreibzugriff wurde
verweigert.", "DocMngr.AppendFile", vbOkOnly)
    ElseIf (oDocMngr.LastError.ErrorCode = 33) Then
        Call cRM.DialogMessageBox("Das Hinzufgen der Datei """ & sFileToAppend &
""" konnte nicht erfolgreich durchgeführt werden. Das Kopieren der Datei ist
fehlgeschlagen.", "DocMngr.AppendFile", vbOkOnly)
    End If
End If

Set oRelation = Nothing
Set oRecord = Nothing
Set oDocMngr = Nothing
```

Beispiel C#-Script:

```
// Erstellt einen neuen Aktivitäten-Datensatz und fügt ein Dokument hinzu. Basis
ist hierbei der Aktivitäten-Container der Firmen-Ansicht einer combit_Large-
Solution

DocMngr docMngr = cRM.CurrentProject.DocMngr;
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
Relation relation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID");
string fileToAppend = @"C:\Firmen-Dossier.docx";
string docDescription = "docx";

if (docMngr.AppendFile(record, relation, fileToAppend, docDescription) == true)
{
    cRM.DialogMessageBox("Die Datei \"" + fileToAppend + "\" konnte erfolgreich
hinzugefügt werden.", "DocMngr.AppendFile", 0);
}
else
{
    if (docMngr.LastError.ErrorCode == 32)
    {
        cRM.DialogMessageBox("Das Hinzufügen der Datei \"" + fileToAppend + "\"
konnte nicht erfolgreich durchgefhrt werden. Der Schreibzugriff wurde
verweigert.", "DocMngr.AppendFile", 0);
    }
    else if (docMngr.LastError.ErrorCode == 33)
    {
```

```
        cRM.DialogMessageBox("Das Hinzufügen der Datei \"" + fileToAppend + "\"
konnte nicht erfolgreich durchgefhrt werden. Das Kopieren der Datei ist
fehlgeschlagen.", "DocMngr.AppendFile", 0);
    }
}

relation.Dispose();
record.Dispose();
docMngr.Dispose();
```

AppendFileExt

Beschreibung:

Fügt einen neuen relationalen Datensatz mit einer Datei hinzu. Die Datei wird in das erste gefundene Dokumentenfeld der relationalen Ansicht geschrieben. Alle weiteren Felder werden ignoriert.

Dokumentenfelder: (eingebettete) Datei, (eingebettete) Grafik oder DMS Dokument.

Parameter:

Parametername	Typ	Beschreibung
oRecord	Objekt	Datensatz-Objekt, zu dem ein relationaler Datensatz angehängt werden soll, z. B. *Firmen*
oRelation	Objekt	Relations-Objekt, in dem der neue relationale Datensatz mit der angehängten Datei eingefügt wird.
sFile	String	Datei, welche hinzugefügt werden soll (gesamter Pfad + Dateiname).
sDocDescriptionCfg	String	Dateiendung der zu verwendeten Dokumentenkonfiguration oder mit | getrennte Dokumentenkonfigurationsbezeichnung, welche in der Konfiguration der Dokumentenverwaltung angeben wurde, z. B. *msg* oder *msg|E-Mails*

Rückgabewert:

Record

Wert	Beschreibung
Record	Gültiges Record-Objekt. Kann mit `If oObject Is Nothing Then` überprüft werden.
NULL	Ungültiges Record-Objekt. Der relationale Datensatz mit der angefügten Datei konnte nicht erstellt werden.

Wichtig: Das zurückgelieferte **Record**-Objekt muss unbedingt wieder freigeben werden (`Set oObject = Nothing setzen`), um den Speicher freizugeben!

Beispiel VBScript:

```
' Erstellt einen neuen Aktivitäten-Datensatz und fügt ein Dokument hinzu. Basis
ist hierbei der Aktivitäten-Container der Firmen-Ansicht einer combit_Large-
Solution

Dim oDocMngr : Set oDocMngr = cRM.CurrentProject.DocMngr
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Dim oRelation : Set oRelation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID")
Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName(oRelation.ForeignViewName)
Dim sFileToAppend : sFileToAppend = "C:\Firmen-Dossier.docx"
```

```
Dim sDocDescription : sDocDescription = "docx"

Dim oRecordAppended : Set oRecordAppended = oDocMngr.AppendFileExt(oRecord,
oRelation, sFileToAppend, sDocDescription)

If (Not oRecordAppended Is Nothing) Then
  If (cRM.DialogMessageBox("Die Datei """ & sFileToAppend & """ konnte erfolgreich
hinzugefügt werden. Soll dieser Datensatz jetzt angezeigt werden?",
"DocMngr.AppendFileExt", vbYesNoCancel) = vbYes) Then
      Call cRM.CurrentProject.OpenNewViewByName(oRelation.ForeignViewName,
"SetFilterDirectSQL:SELECT """ & oViewConfig.PrimaryKeyFldName &  """ FROM """ &
oViewConfig.DBTableName & """ WHERE """ & oViewConfig.DBTableName & """."""" &
oViewConfig.PrimaryKeyFldName & """ = 0x" &
oRecordAppended.GetContentsByName(oViewConfig.PrimaryKeyFldName))
  Else
      Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
  End If
Else
  If (oDocMngr.LastError.ErrorCode = 32) Then
      Call cRM.DialogMessageBox("Das Hinzufgen der Datei """ & sFileToAppend & """
konnte nicht erfolgreich durchgeführt werden. Der Schreibzugriff wurde
verweigert.", "DocMngr.AppendFileExt", vbOkOnly)
  ElseIf (oDocMngr.LastError.ErrorCode = 33) Then
      Call cRM.DialogMessageBox("Das Hinzufgen der Datei """ & sFileToAppend & """
konnte nicht erfolgreich durchgeführt werden. Das Kopieren der Datei ist
fehlgeschlagen.", "DocMngr.AppendFileExt", vbOkOnly)
  End If
End If

Set oRecordAppended = Nothing
Set oViewConfig = Nothing
Set oRelation = Nothing
Set oRecord = Nothing
Set oDocMngr = Nothing
```

Beispiel C#-Script:

```
// Erstellt einen neuen Aktivitäten-Datensatz und fügt ein Dokument hinzu. Basis
ist hierbei der Aktivitäten-Container der Firmen-Ansicht einer combit_Large-
Solution

DocMngr docMngr = cRM.CurrentProject.DocMngr;
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
Relation relation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID");
ViewConfig viewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName(relation.ForeignViewName);
string fileToAppend = @"C:\Firmen-Dossier.docx";
string docDescription = "docx";

Record recordAppended = docMngr.AppendFileExt(record, relation, fileToAppend,
docDescription);

if (recordAppended != null)
{
    if (cRM.DialogMessageBox("Die Datei \"" + fileToAppend + "\" konnte
erfolgreich hinzugefügt werden. Soll dieser Datensatz jetzt angezeigt werden?",
"DocMngr.AppendFileExt", 3) == 6)
    {
        cRM.CurrentProject.OpenNewViewByName(relation.ForeignViewName,
"SetFilterDirectSQL:SELECT \"" + viewConfig.PrimaryKeyFldName + "\" FROM \"" +
viewConfig.DBTableName + "\" WHERE \"" + viewConfig.DBTableName + "\".\"" +
viewConfig.PrimaryKeyFldName + "\" = 0x" +
recordAppended.GetContentsByName(viewConfig.PrimaryKeyFldName));
    }
    else
    {
        cRM.CurrentProject.ActiveViews.ActiveView.Update();
    }
```

```
}
else
{
    if (docMngr.LastError.ErrorCode == 32)
    {
        cRM.DialogMessageBox("Das Hinzufügen der Datei \"" + fileToAppend + "\"
konnte nicht erfolgreich durchgeführt werden. Der Schreibzugriff wurde
verweigert.", "DocMngr.AppendFile", 0);
    }
    else if (docMngr.LastError.ErrorCode == 33)
    {
        cRM.DialogMessageBox("Das Hinzufügen der Datei \"" + fileToAppend + "\"
konnte nicht erfolgreich durchgeführt werden. Das Kopieren der Datei ist
fehlgeschlagen.", "DocMngr.AppendFile", 0);
    }

}

recordAppended.Dispose();
viewConfig.Dispose();
relation.Dispose();
record.Dispose();
docMngr.Dispose();
```

AppendFileExt2

Beschreibung:

Fügt einen neuen relationalen Datensatz mit einer Datei hinzu. Die Datei wird in das erste gefundene Dokumentenfeld der relationalen Ansicht geschrieben. Alle weiteren Felder werden ignoriert.

Dokumentenfelder: (eingebettete) Datei, (eingebettete) Grafik oder DMS Dokument.

Parameter:

Parametername	Typ	Beschreibung
sViewName	String	Name der Ansicht, zu der ein relationaler Datensatz angehängt werden soll, z. B. *Firmen*
sRecordID	String	Datensatz-ID, zu der ein relationaler Datensatz angehängt werden soll.
sRelationToken	String	Relation, zu der ein neuer relationaler Datensatz mit der angehängten Datei eingefügt werden soll, z. B. *ID.Aktivitäten.ContactID*
sFile	String	Datei, welche hinzugefügt werden soll (gesamter Pfad + Dateiname).
sDocDescriptionCfg	String	Dateiendung der zu verwendeten Dokumentenkonfiguration oder mit | getrennte Dokumentenkonfigurationsbezeichnung, welche in der Konfiguration der Dokumentenverwaltung angeben wurde, z. B. *msg* oder *msg|E-Mails*

Rückgabewert:

Record

Wert	Beschreibung
Record	Gültiges Record-Objekt. Kann mit `If oObject Is Nothing Then` überprüft werden.
NULL	Ungültiges Record-Objekt. Der relationale Datensatz mit der angefügten Datei konnte nicht erstellt werden.

Wichtig: Das zurückgelieferte **Record**-Objekt muss unbedingt wieder freigeben werden (`Set oObject = Nothing setzen`), um den Speicher freizugeben!

Beispiel VBScript:

```
' Erstellt einen neuen Aktivitäten-Datensatz und fügt ein Dokument hinzu. Basis
ist hierbei der Aktivitäten-Container der Firmen-Ansicht einer combit_Large-
Solution

Dim oDocMngr : Set oDocMngr = cRM.CurrentProject.DocMngr
Dim sViewName : sViewName = cRM.CurrentProject.ActiveViews.ActiveView.Config.Name
Dim sRecordID : sRecordID =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetConten
tsByName("ID")
Dim sRelationToken : sRelationToken = "ID.Aktivitäten.CompanyID"
Dim sFileToAppend : sFileToAppend = "C:\Firmen-Dossier.docx"
Dim sDocDescription : sDocDescription = "docx"

Dim oRecordAppended : Set oRecordAppended = oDocMngr.AppendFileExt2(sViewname,
sRecordID, sRelationToken, sFileToAppend, sDocDescription)

If (Not oRecordAppended Is Nothing) Then
    If (cRM.DialogMessageBox("Die Datei """ & sFileToAppend & """ konnte
erfolgreich hinzugefügt werden. Soll dieser Datensatz jetzt angezeigt werden?",
"DocMngr.AppendFileExt2", vbYesNoCancel) = vbYes) Then
        Call cRM.CurrentProject.OpenNewViewByName("Aktivitäten",
"SetFilterDirectSQL:SELECT ""ID"" FROM ""Activities"" WHERE ""Activities"".""ID""
= 0x" & oRecordAppended.GetContentsByName("ID"))
    Else
        Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
    End If
Else
    If (oDocMngr.LastError.ErrorCode = 32) Then
        Call cRM.DialogMessageBox("Das Hinzufügen der Datei """ & sFileToAppend &
""" konnte nicht erfolgreich durchgeführt werden. Der Schreibzugriff wurde
verweigert.", "DocMngr.AppendFileExt2", vbOkOnly)
    ElseIf (oDocMngr.LastError.ErrorCode = 33) Then
        Call cRM.DialogMessageBox("Das Hinzufgen der Datei """ & sFileToAppend &
""" konnte nicht erfolgreich durchgeführt werden. Das Kopieren der Datei ist
fehlgeschlagen.", "DocMngr.AppendFileExt2", vbOkOnly)
    End If
End If

Set oRecordAppended = Nothing
Set oDocMngr = Nothing
```

Beispiel C#-Script:

```
// Erstellt einen neuen Aktivitäten-Datensatz und fügt ein Dokument hinzu. Basis
ist hierbei der Aktivitäten-Container der Firmen-Ansicht einer combit_Large-
Solution

DocMngr docMngr = cRM.CurrentProject.DocMngr;
string viewName = cRM.CurrentProject.ActiveViews.ActiveView.Config.Name;
string recordID =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetConten
tsByName("ID");
string relationToken = "ID.Aktivitäten.CompanyID";
string fileToAppend = @"C:\Firmen-Dossier.docx";
string docDescription = "docx";

Record recordAppended = docMngr.AppendFileExt2(viewName, recordID, relationToken,
fileToAppend, docDescription);

if (recordAppended != null)
{
    if (cRM.DialogMessageBox("Die Datei \"" + fileToAppend + "\" konnte
erfolgreich hinzugefügt werden. Soll dieser Datensatz jetzt angezeigt werden?",
"DocMngr.AppendFileExt", 3) == 6)
    {
        cRM.CurrentProject.OpenNewViewByName("Aktivitäten",
"SetFilterDirectSQL:SELECT \"ID\" FROM \"Activities\" WHERE \"Activities\".\"ID\"
= 0x" + recordAppended.GetContentsByName("ID"));
    }
    else
```

```
        {
            cRM.CurrentProject.ActiveViews.ActiveView.Update();
        }
    }
    else
    {
        if (docMngr.LastError.ErrorCode == 32)
        {
            cRM.DialogMessageBox("Das Hinzufügen der Datei \"" + fileToAppend + "\"
    konnte nicht erfolgreich durchgefhrt werden. Der Schreibzugriff wurde
    verweigert.", "DocMngr.AppendFile", 0);
        }
        else if (docMngr.LastError.ErrorCode == 33)
        {
            cRM.DialogMessageBox("Das Hinzufügen der Datei \"" + fileToAppend + "\"
    konnte nicht erfolgreich durchgeführt werden. Das Kopieren der Datei ist
    fehlgeschlagen.", "DocMngr.AppendFile", 0);
        }

    }

    recordAppended.Dispose();
    docMngr.Dispose();
```

3.19 EmailTool Objekt

Das EmailTool-Objekt beinhaltet Eigenschaften und Methoden, um ein in combit CRM integriertes externes Tool für professionelles und leistungsfähiges Kampagnenmanagement zu verwenden. Aktuell stehen dafür die folgenden Anbieter zur Verfügung:

- Brevo
- CleverReach
- Inxmail
- Mailchimp

Damit die jeweiligen Tools verwendet werden können, sind in combit CRM entsprechende Anmeldedaten für das ausgewählte Tool zu hinterlegen. Diese werden in den Projekt-Einstellungen auf der Registerkarte Einstellungen vorgenommen. Details dazu können dem Handbuch in Kapitel "Anbindung an E-Mail-Tools" entnommen werden.

Bekannte Einschränkungen und wichtige Hinweise

Im Folgenden werden bisher bekannte Einschränkungen und Hinweise der einzelnen E-Mail-Tools aufgelistet. Doch aufgrund des agilen Marktes und der Freigabe eines combit CRM Servicepacks kann es unter Umständen zu zeitlichen Überschneidungen kommen.

Mailchimp

- Es wird bei der Anlage eines neuen Accounts - unabhängig vom verwendeten Preismodell (auch 'Plans' genannt) - immer direkt auch eine neue Audience/Liste erstellt. Hierbei gilt es zu beachten, dass beim sogenannten 'Free-Plan' lediglich nur eine einzige Audience/Liste erlaubt ist und somit das Speichern einer neuen Liste nicht erlaubt ist.

CleverReach

- Wichtiges zum kostenlosen Tarif:
  - Es ist soweit eingeschränkt, dass nur eine Empfängerliste verwendet werden kann. Darüber hinaus, können beim Erstellen der einen Empfängerliste lediglich nur noch 3 Felder verwendet werden. Daher wird das Erstellen neuer Listen aus combit CRM nur eingeschränkt funktionieren können.

  - Es ist immer die sogenannte Klickmessung aktiv, die auch nicht deaktiviert werden kann.

  - Personalisierte Informationen wie bspw. Öffnungen, Klicks etc. können mit Hilfe von EmailToolMailingResults.[GetStatistics/GetReports] nicht abgefragt werden.

Hinweise zum Fehlerhandling:

Jedes der Objekte, die für das EmailTool verwendet werden können, besitzt auch jeweils die LastError-Eigenschaft, um erweiterte Informationen über unerwartete Ereignisse mit Hilfe des OLEError-Objektes abfragen zu können. In der NativeErrorCode-Eigenschaft ist der allgemeine Fehlercode des EmailTools hinterlegt und in der ErrorText-Eigenschaft wird der dazu passende Text für die Anzeige mit Hinweis auf die Ursache (nativer Fehler-Text des EmailTools) gespeichert.

3.19.1 Eigenschaften

LastError, read-only

Beschreibung:

Liefert ein OLEError-Objekt zurück.

Typ:

OLEError

Name, read-only

Beschreibung:

Liefert den Namen des eingestellten E-Mail-Tools aus den Projekteigenschaften zurück.

Typ:

String

3.19.2 Methoden

CreateMailing

Beschreibung:

Liefert ein EmailToolMailing-Objekt zurück, um eine neue Kampagne zu erstellen.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der zu erstellenden Kampagne.
Beschreibung	String	Eine Kurzbeschreibung für die zu erstellende Kampagne.

Rückgabewert:

EmailToolMailing

CreateRecipientList

Beschreibung:

Liefert ein EmailToolRecipientList-Objekt zurück, um eine neue Empfänger-Liste erstellen zu können.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der zu erstellenden Empfänger-Liste.

Beschreibung	String	Eine Kurzbeschreibung für du zu erstellende Empfänger-Liste.
Attributszuord-nungsdatei	String	Dateipfad zur Attributszuordnungsdatei. Die dort eingestellten Datensatz-Felder werden vom übergebenen Record-Objekt abgefragt und beim Speichern der Empfänger-Liste übertragen.

Rückgabewert:

EmailToolRecipientList

GetMailing

Beschreibung:

Liefert ein EmailToolMailing-Objekt zurück, um eine bereits vorhandene Kampagne zu bearbeiten.

Parameter:

Parametername	Typ	Beschreibung
Kampagnen ID	String	Eindeutige ID der Kampagne aus dem Zielsystem. Die ID erhält man beim Erstellen einer neuen Kampagne, wenn Save() erfolgreich aufgerufen wurde. Um auf eine bereits vorhandene Kampagne des Zielsystems zugreifen zu können, kann auch dessen ID verwendet werden. Folgende Beschreibung kann sich aufgrund externer Anwendungen ändern und muss ggf. angepasst werden. Brevo: Im Online-Portal wird die ID der Kampagne innerhalb der E-Mail-Kampagnen-Übersicht unterhalb des Kampagnennamens mit einem vorangestellten #-Zeichen angezeigt. CleverReach: Markieren Sie im Online-Portal unter "Emails" oder "Reports" die gewünschte Kampagne und die unten im Browser angezeigte URL enthält im letzten Part "…&id=" die ID. Inxmail: Wählen Sie in Inxmail Professional unter "Cockpit" den "Schnelleinstieg" und markieren Sie das gewünschte Mailing. Hier kann dann im unteren Bereich "Details" die ID eingesehen werden. Mailchimp: Für noch nicht versendete Kampagnen: Wählen Sie im Online-Portal unter "Campaigns" die gewünschte Kampagne und markieren Sie dort im Drop-Down "View email". Jetzt wird die URL dazu unten im Browser angezeigt und der letzte Part "…&id=" enthält die ID. Für bereits versendete Kampagnen: Wählen Sie im Online-Portal unter "Campaigns" die gewünschte Kampagne und führen Sie dort im Drop-Down "View email" aus. Der Link hinter dem Button "Past Issues" enthält die URL dazu und wird unten im Browser angezeigt. Der letzte Part "…&id=" enthält die ID.

Rückgabewert:

EmailToolMailing

GetMailingLists

Beschreibung:

Liefert ein DataCollection-Objekt zurück, um eine Liste der aktuellen Mailings zu erhalten.

Hierbei stehen dann pro DataItem-Objekt die folgenden Felder zur Verfügung, die Case-Sensitiv sind:

Feldname	Beschreibung
id	Enthält die ID des Mailings.
name	Enthält den Namen des Mailings.
subject	Enthält den Betreff des Mailings.
create_time	Enthälte Datum/Zeit, wann das Mailing ursprünglich erstellt wurde.

Rückgabewert:

DataCollection

GetRecipientFolders

Beschreibung:

Liefert ein DataCollection-Objekt zurück, um eine Liste des aktuellen Ordners zu erhalten.

Hinweis: Diese Methode steht nur für Brevo zur Verfügung.

Hierbei stehen dann pro DataItem-Objekt die folgenden Felder zur Verfügung, die Case-Sensitiv sind:

Feldname	Beschreibung
id	Enthält die ID des Ordners.
name	Enthält den Namen des Ordners.
create_time	Enthält Datum/Zeit, wann der Ordner ursprünglich erstellt wurde. Hinweis: Brevo gibt diese Information nicht aus, weswegen ein Platzhalter-Datum zurückgegeben wird.

Rückgabewert:

DataCollection

GetRecipientList

Beschreibung:

Liefert ein EmailToolRecipientList-Objekt zurück, um eine bereits vorhandene Empfänger-Liste zu bearbeiten.

Parameter:

Parametername	Typ	Beschreibung
Empfänger-Listen ID	String	Eindeutige ID der Empfänger-Liste aus dem Zielsystem. Die ID erhält man beim Erstellen einer neuen Empfänger-Liste, wenn Save() erfolgreich aufgerufen wurde. Um auf eine bereits vorhandene Empfänger-Liste des Zielsystems zugreifen zu können, kann auch dessen ID verwendet werden. Folgende Beschreibung kann sich aufgrund externer Anwendungen ändern und muss ggf. angepasst werden. Brevo: Wählen Sie im Online-Portal die Listenübersicht aus. Die ID der Liste wird nun in der ID-Spalte mit einem vorangestellten #-Zeichen angezeigt. Befindet man sich bereits in einer Liste, dann wird die ID vor dem Namen der Liste ausgegeben. CleverReach: Wählen Sie im Online-Portal unter "Recipients" die gewünschte Liste aus und wählen Sie dort "Settings > "General".

		Inxmail: Wählen Sie in Inxmail Professional unter "Cockpit" in die Listenübersicht und markieren Sie die gewünschte Liste aus. Hier kann dann im unteren Bereich "Details" die ID eingesehen werden. Mailchimp: Wählen Sie im Online-Portal unter "Audience" die gewünschte Liste aus und wählen Sie dort in dessen Settings "Settings > "Audience name and defaults".
Attributszuordnungs-datei	String	Dateipfad zur Attributszuordnungsdatei. Wird benötigt, um bspw. beim Hinzufügen (AddRecord/AddRecordSet) von neuen Datensätzen die Zuordnung der Felder aus combit CRM in die Empfänger-Liste zu steuern.

Rückgabewert:

EmailToolRecipientList

GetRecipientLists

Beschreibung:

Liefert ein DataCollection-Objekt zurück, um eine Liste der aktuellen Empfänger-Listen zu erhalten.

Hierbei stehen dann pro DataItem-Objekt die folgenden Felder zur Verfügung, die Case-Sensitiv sind:

Feldname	Beschreibung
id	Enthält die ID der Empfänger-Liste.
name	Enthält den Namen der Empfänger-Liste.
create_time	Enthälte Datum/Zeit, wann die Empfänger-Liste ursprünglich erstellt wurde.

Rückgabewert:

DataCollection

SetOption

Beschreibung:

Ermöglicht das Setzen von Optionen.

Optionen, die unterstützt werden (+), sind abhängig vom verwendeten E-Mail-Tool. Ein Eintrag (-) bedeutet, dass die Option nicht unterstützt wird.

Name	Beschreibung	Typ	Brevo	CleverReach	Inxmail	Mailchimp
Support-MailStatus Pending	Wenn die Option auf True gesetzt wurde, dann wird der Status "pending" nicht als "unsubscribed" zurückgegeben, sondern als "pending"	Bool	-	-	-	+

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name der Option.
Wert	Variant	Wert der Option.

Rückgabewert:

Bool

3.20 EmailToolMailing Objekt

Das EmailToolMailing-Objekt beinhaltet Eigenschaften und Methoden, um eine Kampagne zu erstellen, anzuzeigen und dessen Auswertungen abzufragen.

3.20.1 Eigenschaften

ID, read-only

Beschreibung:

Liefert die ID der aktuellen Kampagnen zurück.

Typ:

String

LastError, read-only

Beschreibung:

Liefert ein OLEError-Objekt zurück.

Typ:

OLEError

3.20.2 Methoden

AddRecipientList

Beschreibung:

Weist der aktuellen Kampagne eine Empfänger-Liste über dessen ID zu.

Parameter:

Parametername	Typ	Beschreibung
Empfänger-Listen ID	String	Eindeutige ID der Empfänger-Liste, die für die Kampagnen verwendet werden soll. Hinweis: Aktuell wird nur eine Empfänger-Liste pro Kampagne unterstützt!

Rückgabewert:

Bool

Display

Beschreibung:

Versucht die Kampagne direkt im Zielsystem zu öffnen, sofern das eingestelle System dies unterstützt.

Wert	Beschreibung
Brevo	Startet das Web-Portal von Brevo im Standard-Browser und öffnet das Mailing. Ist noch kein Login im Web-Portal aktiv, muss dies manuell durchgeführt werden.
CleverReach	Startet das Web-Portal von CleverReach im Standard-Browser und öffnet das Mailing. Ist noch kein Login im Web-Portal aktiv, muss dies manuell durchgeführt werden.

Inxmail	Startet die lokal installierte Anwendung von Inxmail. Die Navigation zum Mailing muss manuell durchgeführt werden.
Mailchimp	Startet das Web-Portal von Mailchimp im Standard-Browser. Ist noch kein Login im Web-Portal aktiv, muss dies manuell durchgeführt werden. Die Navigation zum Mailing muss manuell durchgeführt werden.

Rückgabewert:

Bool

GetResults

Beschreibung:

Liefert ein EmailToolMailingResults-Objekt zurück, um diverse Statistiken und Rückläufer der Kampagne abfragen zu können.

Rückgabewert:

EmailToolMailingResults

Save

Beschreibung:

Speichert die neu erstellte Kampagne mit den Werten aus SetProperty im Zielsystem ab und liefert die neue ID dafür zurück.

Rückgabewert:

String

SetProperty, write-only

Beschreibung:

Darüber können für eine ausschließlich neue Kampagne spezifische Eigenschaften gesammelt werden, die dann später bei der Speicherung berücksichtigt werden. Hier gibt es Eigenschaften, die immer für eine neue Kampagne gesetzt sein müssen (+) und ist abhängig vom verwendeten EmailTool. (-) bedeutet, dass die Eigenschaft nicht verwendet werden kann:

Name	Beschreibung	Brevo	CleverReach	Inxmail	Mailchimp
Subject	Betreff des Mailings	+	+	-	-
SenderName	Name des Absenders	+	+	-	+
SenderEmailAddress	E-Mail-Adresse des Absenders	+	+	-	-
ReplyToEmailAddress	Antwort-E-Mail-Adresse	-	-	-	+

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name der Eigenschaft.
Wert	String	Wert für die Eigenschaft.

Rückgabewert:

Bool

3.21 EmailToolRecipientList Objekt

Das EmailToolRecipientList-Objekt beinhaltet Eigenschaften und Methoden, um eine Empfänger-Liste zu verwalten.

Damit auch individuelle Informationen aus dem Datensatz eines Empfängers in der Empfänger-Liste verwendet werden können, bedarf es einer sogenannten Attributszuordnungsdatei. Hier wird eine Zuordnung zwischen dem Datensatz-Feld sowie dem Feld der Empfänger-Liste angeben. Hierbei können nicht nur Felder, sondern auch feste Texte oder auch Formeln verwendet werden.

Aufbau für Datensatz-Feld:

Hier muss der Feldname lediglich in spitzen Klappern angegeben werden. Jedoch aufgrund des XML-Formats muss die Spitze Klammern entsprechend maskiert werden - <Feldname>

Aufbau für festen Text:

Damit einfacher Text verwendet werden kann, muss der Inhalt mittels doppelter Hochkommata eingeklammert sein - "Fester Text"

Aufbau für Formel:

Soll sich der Inhalt eines Feldes dynamisch über eine Formel ergeben können, muss diese Formel in sogenannten Chevrons eingeklammert werden - «Formel»

Der Aufbau der Datei im XML-Format muss dabei diesen Aufbau aufweisen:

```
<?xml version="1.0" encoding="UTF-16" standalone="yes"?>
<!-- DTD -->
<!DOCTYPE profile [
 <!ENTITY cr "&#x000a;<!--CR-->">
 <!ENTITY tab "&#x0009;<!--TAB-->">
 <!ELEMENT profile (list*, item*)*>
  <!ELEMENT list (list?, item?)*>
  <!ATTLIST list name CDATA #REQUIRED>
   <!ELEMENT item (#PCDATA)>
   <!ATTLIST item name CDATA #REQUIRED>
   <!ATTLIST item xml:space (default|preserve) "preserve"> ]>
<!-- DATA -->
<profile>
 <list name="MailProviderAttributesStructure">
    <item name="<Feldname1 in der Empfänger-Liste>"><<Feldname1 in combit
CRM Datensatz>></item>
      <item name="<Feldname2 in der Empfänger-Liste>"><«Formel»></item>
    <item name="<Feldname3 in der Empfänger-Liste>"><"Fester Text"></item>
 </list>
 </profile>
```

Hier ein Beispiel für eine Empfänger-Liste in Mailchimp:

```
<?xml version="1.0" encoding="UTF-16" standalone="yes"?>
<!-- DTD -->
<!DOCTYPE profile [
 <!ENTITY cr "&#x000a;<!--CR-->">
 <!ENTITY tab "&#x0009;<!--TAB-->">
 <!ELEMENT profile (list*, item*)*>
  <!ELEMENT list (list?, item?)*>
  <!ATTLIST list name CDATA #REQUIRED>
   <!ELEMENT item (#PCDATA)>
   <!ATTLIST item name CDATA #REQUIRED>
   <!ATTLIST item xml:space (default|preserve) "preserve"> ]>
<!-- DATA -->
<profile>
 <list name="MailProviderAttributesStructure">
    <item name="EMAIL">&lt;ContactID.Kontakte.ID.Email&gt;</item>
    <item name="FNAME">&lt;ContactID.Kontakte.ID.Firstname&gt;</item>
    <item name="LNAME">&lt;ContactID.Kontakte.ID.Name&gt;</item>
      <item name="DATE">«Date(Date$(Now(),"%02d.%02m.%y"))»</item>
      <item name="TEXT">"Das ist ein fester Text."</item>
 </list>
 </profile>
```

3.21.1 Eigenschaften

ID, read-only

Beschreibung:

Liefert die ID der aktuellen Empfänger-Liste zurück.

Typ:

String

LastError, read-only

Beschreibung:

Liefert ein OLEError-Objekt zurück.

Typ:

OLEError

OperationStatus, read-only

Beschreibung:

Hiermit kann überprüft werden, ob die Operationen beim Speichern (Save()) bereits abgeschlossen sind. Es wird ein DataCollection-Objekt zurückgeliefert, welches ein DataItem-Objekt enthält. Dabei können die folgenden Werte abgefragt werden:

Feldname	Beschreibung
status	Liefert den aktuellen Status der Operation. Mögliche Werte dabei sind: 12: Operation wird noch ausgeführt und ist noch nicht abgeschlossen. 13: Operation ist erfolgreich abgeschlossen. 14: Operation ist fehlgeschlagen. 15: Operation wurde abgebrochen. 16: Operation wurde abgeschlossen; jedoch sind Fehler aufgetreten. In den weiteren Feldern (siehe unten) können weitere Informationen über die Fehler abgefragt werden. 17: Es gibt aktuell keine Operation. Dies kann bspw. eintreten, wenn lediglich eine neue Empfänger-Liste erstellt wird und keine Operation wie AddRecord/AddRecordSet, UpdateRecord/UpdateRecordSet oder RemoveRecord/RemoveRecordSet aufgerufen wurde.
error_count_add	Liefert die Anzahl der bei AddRecord/AddRecordSet fehlerhaften Empfänger.
error_count_update	Liefert die Anzahl der bei UpdateRecord/UpdateRecordSet fehlerhaften Empfänger.
error_count_remove	Liefert die Anzahl der bei RemoveRecord/RemoveRecordSet fehlerhaften Empfänger.
success_count_add	Liefert die Anzahl der bei AddRecord/AddRecordSet erfolgreichen Empfänger.
success_count_update	Liefert die Anzahl der bei UpdateRecord/UpdateRecordSet erfolgreichen Empfänger.
success_count_remove	Liefert die Anzahl der bei RemoveRecord/RemoveRecordSet erfolgreichen Empfänger.

Typ:

DataCollection

3.21.2 Methoden

AddRecord

Beschreibung:

Sammelt die Informationen aus dem Datensatz für den neuen Empfänger, die später bei Save übertragen wird. Hierbei werden auch E-Mail-Adressen in der Sperrliste berücksichtigt und der Vorgang mit dem Rückgabewert 2 abgeschlossen.

Hinweis: Wenn bei Mailchimp ein Empfänger über die Online-Oberfläche zuvor gelöscht wurde, der über combit CRM erneut hinzugefügt werden soll, so ist dies technisch über combit CRM nicht mehr möglich. Hierzu muss zwingend der Empfänger selber sich in die betreffende Liste erneut anmelden. Weitere Details dazu finden sich unter Delete Contacts.

Parameter:

Parametername	Typ	Beschreibung
Record-Objekt	Record	Fügt den aktuellen Datensatz in die Empfänger-Liste hinzu. Welche Felder dazu berücksichtigt werden, ergibt sich aus dem Parameter Attributszuordnungsdatei, mit dem das Objekt EmailToolRecipientList erstellt wurde.

Rückgabewert:

Integer

Wert	Beschreibung
0	Es ist ein Fehler aufgetreten.
1	Das Sammeln der Informationen für den Datensatz war erfolgreich.
2	Es ist eine Warnung beim Sammeln der Informationen für den Datensatz aufgetreten - E-Mail-Adresse befindet sich auf der Sperrliste. Im LastError-Objekt mit dem Fehler-Code 66 wird dann in dessen Eigenschaft ErrorAppendix die gesperrte E-Mail-Adresse aufgelistet.

AddRecordSet

Beschreibung:

Sammelt die Informationen aus jedem einzelnen Datensatz im RecordSet für die neuen Empfänger, die später bei Save übertragen werden. Hierbei werden auch E-Mail-Adressen in der Sperrliste berücksichtigt. Befindet sich eine E-Mail-Adresse in der Sperrliste, so wird der Vorgang vollständig ausgeführt und liefert jedoch 2 zurück (siehe Rückgabewert).

Hinweis: Wenn bei Mailchimp ein Empfänger über die Online-Oberfläche zuvor gelöscht wurde, der über combit CRM erneut hinzugefügt werden soll, so ist dies technisch über combit CRM nicht mehr möglich. Hierzu muss zwingend der Empfänger selber sich in die betreffende Liste erneut anmelden. Weitere Details dazu finden sich unter Delete Contacts.

Parameter:

Parametername	Typ	Beschreibung
RecordSet-Objekt	RecordSet	Fügt die einzelnen Datensätze in die Empfänger-Liste hinzu.

		Welche Felder pro Datensatz dazu berücksichtigt werden, ergibt sich aus dem Parameter Attributszuordnungsdatei, mit dem das Objekt **EmailToolRecipientList** erstellt wurde.

Rückgabewert:

Integer

Wert	Beschreibung
0	Es ist ein Fehler aufgetreten.
1	Das Sammeln der Informationen für die Datensätze war erfolgreich.
2	Es ist eine Warnung beim Sammeln der Informationen für die Datensätze aufgetreten - E-Mail-Adresse(n) befindet sich auf der Sperrliste. Im **LastError**-Objekt mit dem Fehler-Code 66 wird dann in dessen Eigenschaft **ErrorAppendix** die gesperrten E-Mail-Adressen semikolon-separiert aufgelistet.

ChangeEmailAddress

Beschreibung:

Ändert die E-Mail-Adresse eines Teilnehmers.

Parameter:

Parametername	Typ	Beschreibung
current_mail	String	Die E-Mail-Adresse, die geändert werden soll.
new_mail	String	Der neue Wert der zu ändernden E-Mail-Adresse.

Rückgabewert:

Bool

GetAllRecipients

Beschreibung:

Liefert ein **DataCollection**-Objekt zurück, um alle Empfänger der Empfänger-Liste abfragen zu können. Hierbei werden auch pro Empfänger die Felder aus der Empfänger-Liste berücksichtigt.

Für jeden Empfänger wird darüber hinaus auch die beiden Felder "status" und "last_change" geliefert, die im **DataItem**-Objekt abgefragt werden können:

Parametername	Typ	Beschreibung
status	String	Enthält die Information über den aktuellen Status des Empfängers: subscribed, unsubscribed, bounced oder pending, je nach gesetztem Wert der Methode **SetOption** im **EmailTool** Objekt.

last_change	DateTime	Liefert Datum/Uhrzeit, wann bei dem Empfänger zuletzt eine Änderung vorgenommen wurde.
		Hierbei handelt es sich um das UTC-Format und kann ggf. mit Hilfe von ConvertUTCToLocalDateTime auf das lokale System-Datum umgerechnet werden – Beispiel:
		`Dim dateTime`
		`dateTime = recipients.GetContentsValueByName("last_change")`
		`cRM.ConvertUTCToLocalDateTime(CDate(dateTime))`

Parameter:

Parametername	Typ	Beschreibung
Attributszuordnungsdatei	String	Dieser Parameter ist optional.

		Wenn dieser nicht angegeben wird, leer ist oder ein "*" enthält, so werden immer alle Felder der Empfänger-Liste zurückgemeldet. Man kann hier auch eine kommaseparierte Liste an Feldnamen übergeben, um gezielt nur eine Auswahl an Felder zu erhalten - bspw. "Feld1, Feld8" Alternativ kann aber auch der Dateipfad zur Attributszuordnungsdatei angegeben werden. Die dort eingestellten Felder werden pro Empfänger aus der Empfänger-Liste abgefragt.

Rückgabewert:

DataCollection

GetOperationResults

Beschreibung:

Hiermit kann überprüft werden, wenn die Operationen beim Speichern (Save()) abgeschlossen ist, ob und wenn ja welche Fehler dabei entstanden sind. Liefert ein DataCollection-Objekt zurück, und liefert alle Empfänger, die für die Speichern-Operation relevant sind, zurück.

Hierbei stehen dann pro DataItem-Objekt die folgenden Felder zur Verfügung:

Feldname	Typ	Beschreibung
status	String	Bestimmt, ob die Operation erfolgreich oder fehlerhaft war. ok: Operation war erfolgreich failed: Es ist ein Fehler für die Operation aufgetreten
type	String	Bestimmt den Typen der Operation. add: Der Empfänger sollte hinzugefügt werden update: Der Empfänger sollte aktualisiert werden remove: Der Empfänger sollte entfernt werden
address	String	Enthält die E-Mail-Adresse des betreffenden Empfängers.
errortext	String	Enthält den Fehlertext, wieso die Operation für den betreffenden Empfänger nicht duchgeführt werden konnte.

Rückgabewert:

DataCollection

GetRecipientStatus

Beschreibung:

Liefert ein DataCollection-Objekt zurück, um den Empfänger der Empfänger-Liste abfragen zu können.

Hinweis: Diese Methode steht nur für Brevo zur Verfügung.

Für den Empfänger werden die beiden Felder "status" und "last_change" geliefert, die im DataItem-Objekt abgefragt werden können:

Parametername	Typ	Beschreibung
status	String	Enthält die Information über den aktuellen Status des Empfängers: subscribed, unsubscribed oder bounced.
last_change	DateTime	Liefert Datum/Uhrzeit, wann bei dem Empfänger zuletzt eine Änderung vorgenommen wurde. Hierbei handelt es sich um das UTC-Format und kann ggf. mit Hilfe von ConvertUTCToLocalDateTime auf das lokale System-Datum umgerechnet werden – Beispiel: `Dim dateTime`

		`dateTime = recipients.GetContentsValueByName("last_change")` `cRM.ConvertUTCToLocalDateTime(CDate(dateTime))`

Parameter:

Parametername	Typ	Beschreibung
Record-Objekt	Record	Datensatz, der abgefragt werden soll.
EmailFeldname	String	Feldname, des Feldes, in dem die E-Mail-Adresse des Datensatzes steht.

Rückgabewert:

DataCollection

GetSubscribers

Beschreibung:

Liefert ein DataCollection-Objekt zurück, um nur alle angemeldeten Empfänger ("Subscribers") der Empfänger-Liste abfragen zu können. Hierbei werden auch pro Empfänger die Felder aus der Empfänger-Liste berücksichtigt.

Hierbei wird das Feld "last_change" pro Empfänger zusätzlich zurückgeliefert, welches im DataItem-Objekt abgefragt werden kann:

Parametername	Typ	Beschreibung

last_change	DateTime	Liefert Datum/Uhrzeit, wann bei dem Empfänger zuletzt eine Änderung vorgenommen wurde.
		Hierbei handelt es sich um das UTC-Format und kann ggf. mit Hilfe von ConvertUTCToLocalDateTime auf das lokale System-Datum umgerechnet werden – Beispiel:
		`Dim dateTime`
		`dateTime = subscriber.GetContentsValueByName("last_change")`
		`cRM.ConvertUTCToLocalDateTime(CDate(dateTime))`

Parameter:

Parametername	Typ	Beschreibung
Attributszuordnungsdatei	String	Dieser Parameter ist optional. Wenn dieser nicht angegeben wird, leer ist oder ein "*" enthält, so werden immer alle Felder der Empfänger-Liste zurückgemeldet. Man kann hier auch eine kommaseparierte Liste an Feldnamen übergeben, um gezielt nur eine Auswahl an Felder zu erhalten - bspw. "Feld1, Feld8" Alternativ kann aber auch der Dateipfad zur Attributszuordnungsdatei angegeben werden. Die dort eingestellten Felder werden pro Empfänger aus der Empfänger-Liste abgefragt.

Rückgabewert:

DataCollection

GetUnsubscribers

Beschreibung:

Liefert ein DataCollection-Objekt zurück, um nur alle abgemeldeten Empfänger ("Unsubscribers") der Empfänger-Liste abfragen zu können.

Hierbei werden die beiden Felder "email", "last_change" pro Empfänger zurückgeliefert, die im **DataItem**-Objekt abgefragt werden können:

Parametername	Typ	Beschreibung
email	String	Enthält die E-Mail-Adresse des abgemeldeten Empfängers.

last_change	DateTime	Liefert Datum/Uhrzeit, wann bei dem Empfänger zuletzt eine Änderung vorgenommen wurde.
		Hierbei handelt es sich um das UTC-Format und kann ggf. mit Hilfe von ConvertUTCToLocalDateTime auf das lokale System-Datum umgerechnet werden – Beispiel:
		`Dim dateTime`
		`dateTime = unsubscriber.GetContentsValueByName("last_change")`
		`cRM.ConvertUTCToLocalDateTime(CDate(dateTime))`

Rückgabewert:

DataCollection

RemoveRecord

Beschreibung:

Sammelt die Informationen aus dem Datensatz, um diesen später bei **Save** in der Empfänger-Liste zu entfernen.

Hinweis: Bei der Verwendung von Inxmail wird der Empfänger vollständig global aus dem System entfernt.

Parameter:

Parametername	Typ	Beschreibung
Record-Objekt	Record	Datensatz, der aus der Empfänger-Liste entfernt werden soll.
Email Feldname	String	Feldname, indem die E-Mail-Adresse des zu entfernenden Datensatzes steht.

Rückgabewert:

Bool

RemoveRecordSet

Beschreibung:

Sammelt die Informationen aus jedem einzelnen Datensatz im RecordSet, um diese bei **Save** später aus der Empfänger-Liste zu entfernen.

Hinweis: Bei der Verwendung von Inxmail werden die Empfänger vollständig global aus dem System entfernt.

Parameter:

Parametername	Typ	Beschreibung
RecordSet-Objekt	RecordSet	Datensätze, die aus der Empfänger-Liste entfernt werden sollen.
Email Feldname	String	Feldname, indem die E-Mail-Adresse der zu entfernenden Datensätze steht.

Rückgabewert:

Bool

Save

Beschreibung:

Wurde das EmailToolRecipientList-Objekt mit Hilfe von GetRecipientList abgefragt, wird hierbei eine vorhandene Empfänger-Liste abgefragt. Bei CreateRecipientList wird eine neue Empfänger-Liste mit den Werten aus SetProperty erstellt. Im Anschluss werden die Datensätze aus AddRecord/AddRecordSet, UpdateRecord/UpdateRecordSet sowie RemoveRecord/RemoveRecordSet auf die Empfänger-Liste angewendet.

Es wird als Ergebnis ein DataCollection-Objekt zurückgeliefert, welches bei Erfolg genau ein DataItem-Objekt enthält, mit dem Feld "id" (Case-Sensitiv!), welches der eindeutigen ID der Empfänger-Liste entspricht. Wird eine neue Empfänger-Liste erstellt und gespeichert, sollte man sich die ID unbedingt merken. Wurde eine bereits bekannte Empfänger-Liste gespeichert, so entspricht die ID der bekannten ID der Empfänger-Liste.

Wichtig: Mit Hilfe der Eigenschaft OperationStatus kann geprüft werden, ob das Hinzufügen, Aktualisieren und/oder Entfernen der Empfänger-Liste bereits vollständig abgeschlossen ist.

Rückgabewert:

DataCollection

SetProperty, write-only

Beschreibung:

Darüber können für eine neue Empfänger-Liste spezifische Eigenschaften gesammelt werden, die dann später bei der Speicherung (siehe Save) berücksichtigt werden. Hier gibt es jedoch je nach Email Tool individuelle Eigenschaften, die immer für eine Empfänger-Liste gesetzt sein müssen (+). (-) bedeutet, dass die Eigenschaft nicht verwendet werden kann:

Name	Beschreibung	Brevo	CleverReach	Inxmail	Mailchimp
Subject	Standard-Betreff eines Mailings	-	-	-	+
Language	Standard-Sprache eines Mailings im Format ISO 639-1 (2 Buchstaben) wie bspw. "DE" für Deutsch	-	-	-	+
Company	Name der Firma	-	-	-	+
Address	Straße der Firma	-	-	-	+
City	Stadt der Firma	-	-	-	+
State	Bundesland der Firma	-	-	-	+
Zip	Postleitzahl der Firma	-	-	-	+
Country	Land der Firma im 2-Zeichen Format ISO3166	-	-	-	+
Permission-Reminder	Erinnerungstext, dass man die Berechtigung hat den Empfänger anzuschreiben – Beispiel: Sie erhalten diese E-Mail, weil Sie sich für Produkt-Aktualisierungen angemeldet haben.	-	-	-	+
SenderName	Name des Absenders	-	+	+	+
SenderEmailAddress	E-Mail-Adresse des Absenders	-	-	+	+
ReplyToName	Antwort-Name	-	-	+	-
ReplyToEmailAddress	Antwort-E-Mail-Adresse	-	-	+	-

FolderID	ID des Ordners, in welchem sich die Liste befinden soll	+	-	-	-

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name der Eigenschaft.
Wert	String	Wert für die Eigenschaft.

Rückgabewert:

Bool

SubscribeRecord

Beschreibung:

Versucht den Empfänger für die angegebene E-Mail-Adresse in der aktuellen bereits existierenden Empfänger-Liste einzutragen bzw. als angemeldet zu markieren. Dies wird direkt und ohne Save ausgeführt. Hinweis: Wenn die Empfänger-Liste über ein Double-Opt-In-Verfahren verfügt, wird unter Umständen zunächst nur eine Bestätigungs-E-Mail ausgelöst, über die dann der Empfänger die neue Anmeldung zunächst bestätigen muss.

Parameter:

Parametername	Typ	Beschreibung
Record-Objekt	Record	Datensatz, der in der Empfänger-Liste erneut angemeldet werden soll.
Email Feldname	String	Feldname, indem die E-Mail-Adresse des erneut anzumeldenden Datensatzes steht.

Rückgabewert:

Bool

UnsubscribeRecord

Beschreibung:

Versucht den Empfänger für die angegebene E-Mail-Adresse aus der aktuellen bereits existierenden Empfänger-Liste auszutragen bzw. als abgemeldet zu markieren. Dies wird direkt und ohne Save ausgeführt.

Parameter:

Parametername	Typ	Beschreibung
Record-Objekt	Record	Datensatz, der aus der Empfänger-Liste abgemeldet werden soll.
Email Feldname	String	Feldname, indem die E-Mail-Adresse des abzumeldenden Datensatzes steht.

Rückgabewert:

Bool

UpdateRecord

Beschreibung:

Sammelt die Informationen aus dem Datensatz für das Aktualisieren eines bestehenden Empfängers, die später bei Save übertragen wird. Hierbei werden auch E-Mail-Adressen in der Sperrliste berücksichtigt und der Vorgang mit dem Rückgabewert 2 abgeschlossen.

Parameter:

Parametername	Typ	Beschreibung
Record-Objekt	Record	Aktualisiert den aktuellen Datensatz in der Empfänger-Liste. Welche Felder dazu berücksichtigt werden, ergibt sich aus dem Parameter Attributszuordnungsdatei, mit dem das Objekt EmailToolRecipientList erstellt wurde.

Rückgabewert:

Integer

Wert	Beschreibung
0	Es ist ein Fehler aufgetreten.
1	Das Sammeln der Informationen für den Datensatz war erfolgreich.
2	Es ist eine Warnung beim Sammeln der Informationen für den Datensatz aufgetreten - E-Mail-Adresse befindet sich auf der Sperrliste. Im LastError-Objekt mit dem Fehler-Code 66 wird dann in dessen Eigenschaft ErrorAppendix die gesperrte E-Mail-Adresse aufgelistet.

UpdateRecordSet

Beschreibung:

Sammelt die Informationen aus jedem einzelnen Datensatz im RecordSet für das Aktualisieren bestehender Empfänger, die später bei Save übertragen werden. Hierbei werden auch E-Mail-Adressen in der Sperrliste berücksichtigt. Befindet sich eine E-Mail-Adresse in der Sperrliste, so wird der Vorgang vollständig ausgeführt und liefert jedoch 2 zurück (siehe Rückgabewert).

Parameter:

Parametername	Typ	Beschreibung
RecordSet-Objekt	RecordSet	Aktualisiert die einzelnen Datensätze in der Empfänger-Liste. Welche Felder pro Datensatz dazu berücksichtigt werden, ergibt sich aus dem Parameter Attributszuordnungsdatei, mit dem das Objekt EmailToolRecipientList erstellt wurde.

Rückgabewert:

Integer

Wert	Beschreibung
0	Es ist ein Fehler aufgetreten.
1	Das Sammeln der Informationen für die Datensätze war erfolgreich.
2	Es ist eine Warnung beim Sammeln der Informationen für die Datensätze aufgetreten - E-Mail-Adresse(n) befindet sich auf der Sperrliste. Im LastError-Objekt mit dem Fehler-Code 66 wird dann in dessen Eigenschaft ErrorAppendix die gesperrten E-Mail-Adressen semikolon-separiert aufgelistet.

3.22 EmailToolMailingResults Objekt

Das EmailToolMailingResults-Objekt beinhaltet Eigenschaften und Methoden, um Rückläufer und statistische Werte für eine Kampagne abfragen zu können.

3.22.1 Eigenschaften

LastError, read-only

Beschreibung:

Liefert ein OLEError-Objekt zurück.

Typ:

OLEError

3.22.2 Methoden

GetBounces

Beschreibung:

Liefert ein DataCollection-Objekt zurück, welches die Rückläufer der Kampagne enthält.

Hierbei stehen dann pro DataItem-Objekt die folgenden Felder zur Verfügung:

Feldname	Beschreibung
Email	Enthält die E-Mail-Adresse des Rückläufers.
Category	Bestimmt den Grund für den Rückläufer: SOFT: Typisch, wenn die E-Mail-Adresse temporär nicht beschickt werden kann, wenn bspw. das Postfach überfüllt ist, der E-Mail-Server des Empfängers offline ist, die E-Mail-Nachricht zu groß ist, ein Auto-Responder eingerichtet ist u.v.m. HARD: Typisch, wenn die E-Mail-Adresse nicht existiert, die verwendete Domain unbekannt ist, der E-Mail-Server des Empfängers die Zustellung vollsändig blockiert u.v.m. UNKNOWN: Es kann nicht eindeutig bestimmt werden, ob es sich um einen SOFT- oder HARD-Bounce handelt. Hinweis: Bei CleverReach als eingestelltes EmailTool wird als Kategorie immer UNKNOWN zurückgeliefert.

TimeStamp	Liefert Datum/Uhrzeit, wann der Rückläufer entstanden ist.
	Hierbei handelt es sich um das UTC-Format und kann ggf. mit Hilfe von ConvertUTCToLocalDateTime auf das lokale System-Datum umgerechnet werden – Beispiel:
	`Dim oDate`
	`oDate = oBounce.GetContentsValueByName("TimeStamp")`
	`cRM.ConvertUTCToLocalDateTime(CDate(oDate))`

Rückgabewert:

DataCollection

GetMailingRecipients

Beschreibung:

Liefert ein DataCollection-Objekt zurück, welches die Empfänger der Kampagne enthält.

Hierbei stehen dann pro DataItem-Objekt die folgenden Felder zur Verfügung:

Feldname	Beschreibung
Email	Enthält die E-Mail-Adresse des Empfängers.

Rückgabewert:

DataCollection

Hinweis: Die Methode wird in Verbindung mit Inxmail nicht unterstützt und liefert in diesem Fall Nothing bzw. null zurück.

GetStatistics

Beschreibung:

Liefert ein DataItem-Objekt zurück, welches statistische Informationen zur Kampagne enthält.

Hierbei stehen die folgenden Felder zur Verfügung:

Feldname	Beschreibung
bounceCount	Liefert die Anzahl der Rückläufer der Kampagne. Hinweis: Über GetBounces kann pro Empfänger/E-Mail-Adresse die Art des Rückläufers ermittelt werden.
totalClicks	Liefert die Anzahl der Klicks in der Kampagne.
unsubscribeClicks	Liefert die Anzahl der Empfänger, die auf den Abmelde-Link für die Kampagne geklickt haben.
openingRecipients	Liefert die Anzahl der Empfänger zurück, die die Kampagnen geöffnet haben.
notOpeningRecipients	Liefert die Anzahl der Empfänger zurück, die die Kampagnen nicht geöffnet haben.
sentRecipients	Liefert die Anzahl der versendeten E-Mails.

Rückgabewert:

DataItem

GetReports

Beschreibung:

Liefert ein DataCollection-Objekt zurück, welches empfängerspezifische statistische Werte des Mailings enthält.

Hierbei stehen dann pro DataItem-Objekt die folgenden Felder zur Verfügung:

Feldname	Beschreibung
Email	Enthält die E-Mail-Adresse des Empfängers für dessen statistischen Werte (siehe weitere Felder).

TimeStamp	Liefert Datum/Uhrzeit, wann der Empfänger die Aussendung bzw. die E-Mail-Nachricht dazu das erste Mal geöffnet hat.
	Hierbei handelt es sich um das UTC-Format und kann ggf. mit Hilfe von ConvertUTCToLocalDateTime auf das lokale System-Datum umgerechnet werden – Beispiel:
	`Dim oDate`
	`oDate = oBounce.GetContentsValueByName("TimeStamp")`
	`cRM.ConvertUTCToLocalDateTime(CDate(oDate))`

Clicked	Signalisiert, ob der Empfänger innerhalb der Aussendung bzw. innerhalb der E-Mail-Nachricht einen beliebigen Klick ausgeführt hat. Dabei sind die folgenden Werte möglich: true - Es wurde in der E-Mail geklickt false - Es wurde in der E-Mail nicht geklickt

Rückgabewert:

DataCollection

3.23 InputForm Objekt

Dieses Objekt bietet verschiedene Manipulationsmöglichkeiten direkt in der Eingabemaske in einem Datensatz.

Soll ein Script bei Klicken eines Buttons in der Eingabemaske ohne Speichern-Frage gestartet werden, muss folgende Anweisung enthalten sein:

```
<!--#pragma keepeditmode-->
```

Auf diese Weise wird die Speichern-Frage unterdrückt und Sie bleiben im Edit-Modus.

3.23.1 Eigenschaften

Containers

Beschreibung:

Liefert alle Container der Eingabemaske als Objekt vom Typ **ListContainers** zurück.

Typ:

ListContainers

Beispiel VBScript:

```
' Basis für dieses Beispiel ist der Aktivitäten-Container der Kontakte-Ansicht
einer combit_Large-Solution

Dim oListContainers : Set oListContainers =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers
Dim oContainer : Set oContainer =
oListContainers.ItemByName("ID.Aktivitäten.KontaktID#{B3C0768A-5599-44B5-B4F2-
7D31A6C10EC5}")
Call oContainer.Update()
Set oContainer = Nothing
Set oListContainers = Nothing
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist der Aktivitäten-Container der Kontakte-Ansicht
einer combit_Large-Solution

//<!--#pragma keepeditmode-->
ListContainers listContainers =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers;
Container container =
listContainers.ItemByName("ID.Aktivitäten.KontaktID#{B3C0768A-5599-44B5-B4F2-
7D31A6C10EC5}");
container.Update();
container.Dispose();
listContainers.Dispose();
```

WebElements

Beschreibung:

Liefert alle Web-Elemente der Eingabemaske als Objekt vom Typ **ListWebElements** zurück.

Typ:

ListWebElements

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die WebElemente-Ansicht einer combit_Large-
Solution

<!--#pragma keepeditmode-->
Dim oListWebElements : Set oListWebElements =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).WebElements
Dim oWebElement : Set oWebElement = oListWebElements.ItemByName("{64BBFFD7-EA32-
4358-BBFC-744D6A94291D}")
Dim oInternetExplorer : Set oInternetExplorer = oWebElement.IE()
Call oInternetExplorer.Navigate("https://www.combit.net")
Set oInternetExplorer = Nothing
Set oWebElement = Nothing
Set oListWebElements = Nothing
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist die WebElemente-Ansicht einer combit_Large-
Solution

//<!--#pragma keepeditmode-->
ListWebElements listWebElements =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).WebElements;
WebElement webElement = listWebElements.ItemByName("{64BBFFD7-EA32-4358-BBFC-
744D6A94291D}");
webElement.IE().Navigate("https://www.combit.net");
webElement.Dispose();
listWebElements.Dispose();
```

3.23.2 Methoden

ActivateCardPage

Beschreibung:

Aktiviert die erste Registerkarte mit dem übergebenen Namen.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Registerkartentitel.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die Firmen-Ansicht einer combit_Large-Solution

<!--#pragma keepeditmode-->
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).ActivateCardPage("An
sprechpartner")
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist die Firmen-Ansicht einer combit_Large-Solution

//<!--#pragma keepeditmode-->
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).ActivateCardPage("An
sprechpartner");
```

ConvertToLatestVersion

Beschreibung:

Konvertiert die Eingabemasken-Dialogdefinition (<Ansichtenname>.dli) sowie die etwaige zugehörige WebAccess Eingabemaske (<Ansichtenname>.web.dli) automatisch in das aktuelle Format. Für die Bereitstellung der Eingabemasken für combit.WebAccess ist dies zwingend notwendig, da diese mindestens in Version > 5.000 vorliegen müssen. Desweiteren werden dabei ggf. bestehende Inkonsistenzen korrigiert.

Hinweis: Der Anwender muss hierfür Schreibrechte für die DLI-Dateien besitzen.

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).ConvertToLatestVersi
on()
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).ConvertToLatestVersi
on();
```

DialogSelectRecordDropDown

Beschreibung:

Es wird eine Auswahlliste für einen Datensatz aus dem übergebenen (nicht-visuellen!) RecordSet angezeigt und als Record Objekt zurückgegeben. Die Auswahlliste wird als Drop-Down-Auswahlliste analog zur 1:1-relationalen Datensatzauswahl dargestellt. Es wird ein fully-dynamic RecordSet als Basis benötigt, weitere Informationen finden Sie unter **Änderungen und Neuerungen.**

Wichtig: Nach dem Aufruf von **DialogSelectRecordDropDown** für einen per **ViewConfig.CreateRecordSet** erzeugten RecordSet darf für den betreffenden RecordSet keine **Move**-Methode aufgerufen werden, da sonst der zurückgegebene Record u.U. seine Werte verändert!

Parameter:

Parametername	Typ	Beschreibung
RecordSet	RecordSet	Das RecordSet, auf dem die Auswahlliste basieren soll.
ParentControl-Handle	Long	Optional. Handle eines Fensters. Die dargestellte Auswahlliste wird an diesem Fenster ausgerichtet, andernfalls an das Fenster unterhalb der aktuellen Mauszeigerposition
ProfileName	String	Optional. Unter diesem Profilnamen werden die Einstellungen für den nächsten Aufruf gespeichert. Wird hier ein Feldname angegeben, von welchem aus eine 1:1-Relation wegführt, so teilen sich beide Auswahllisten automatisch ein identisches Erscheinungsbild.
AllowUserDefined-SortOrder	Bool	Optional. Bestimmt, ob eine vom Benutzer in diesem Dialog zuletzt eingestellte Sortierung verwendet werden soll (True) oder die Sortierung des RecordSets verwendet wird (False). Wird kein Wert übergeben wird die erste Sortierung aktiviert, die in der zum RecordSet gehörenden Ansicht definiert wurde.

Rückgabewert:

Record

Beispiel VBScript:

```
' Bietet dem Nutzer eine Datensatzauswahl an und überträgt Informationen aus dem
ausgewählten Datensatz in die aktuell dargestellte Eingabemaske. Basis für dieses
Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

<!--#pragma keepeditmode-->
Dim oProject, oRecordSet, oCurrentInputForm, oRecordSelected

Set oProject = cRM.CurrentProject
Set oCurrentInputForm = oProject.ActiveViews.ActiveView.CurrentInputForm(0)

Set oRecordSet =
oProject.ViewConfigs.ItemByName("Firmen").CreateRecordSet("SetSortOrder:1
SetFilter:upper(""ZIP"") >= upper('70000') AND upper(""ZIP"") <= upper('80000')")
If Not (oRecordSet is Nothing) and oRecordSet.MoveFirst() Then

        ' Auswahl des Datensatzes durch den Benutzer:
        Set oRecordSelected =
oCurrentInputForm.DialogSelectRecordDropDown(oRecordSet,
oCurrentInputForm.GetHwndByName("UserDefined1"), "CompanyID")

        If Not (oRecordSelected Is Nothing) Then
            oCurrentInputForm.SetContentsValueByName "UserDefined1",
oRecordSelected.GetContentsValueByName("Company")
        Else
            ' Abbruch
        End If
Else
    Msgbox "Filter fehlgeschlagen"
End If

' Objekte freigeben:
Set oRecordSelected = Nothing
set oCurrentInputForm = Nothing
Set oRecordSet = Nothing
Set oProject = Nothing
```

Beispiel C#-Script:

```
// Bietet dem Nutzer eine Datensatzauswahl an und überträgt Informationen aus dem
ausgewählten Datensatz in die aktuell dargestellte Eingabemaske. Basis für dieses
Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

// <!--#pragma keepeditmode-->
Project project = cRM.CurrentProject;
RecordSet recordSet =
project.ViewConfigs.ItemByName("Firmen").CreateRecordSet(@"SetSortOrder:1
SetFilter:upper(""ZIP"") >= upper('70000') AND upper(""ZIP"") <= upper('80000')");
InputForm currentInputForm = project.ActiveViews.ActiveView.CurrentInputForm(0);
Record recordSelected = null;

if (recordSet != null && recordSet.MoveFirst())
{
    // Auswahl des Datensatzes durch den Benutzer
    recordSelected = currentInputForm.DialogSelectRecordDropDown(recordSet,
currentInputForm.GetHwndByName("UserDefined1"), "CompanyID");

    if (recordSelected != null)
    {
        currentInputForm.SetContentsValueByName("UserDefined1",
recordSelected.GetContentsValueByName("Company").ToString());
    }
    else
    {
        // Abbruch
```

```
    }
}
else
{
    cRM.DialogMessageBox("Filter fehlgeschlagen!",
"InputForm.DialogSelectRecordDropDown", 0);
}

recordSelected.Dispose();
currentInputForm.Dispose();
recordSet.Dispose();
project.Dispose();
```

GetContentsByName

Beschreibung:

Liefert den Inhalt des Feldes als Zeichenkette zurück, dessen physikalischer Feldname übergeben wurde.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes in der verwendeten/aktiven Eingabemaske.

Rückgabewert:

String

> **Hinweis**: Der Inhalt von Datumsfeldern wird im Format YYYYMMDD zurückgeben, Datumsfelder mit Zeitanteil werden mit YYYYMMDDHHMMSS formatiert.

Beispiel VBScript:

```
' Prüft, ob die Informationen im Feld Name innerhalb der Eingabemaske verändert
wurden. Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-
Solution

<!--#pragma keepeditmode-->
Dim oInputForm : Set oInputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2)
Dim sContactName : sContactName = oInputForm.GetContentsByName("Name")
Dim sContactFirstName : sContactFirstName =
oInputForm.GetContentsByName("Firstname")
Set oInputForm = Nothing

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Dim sContactNameSaved : sContactNameSaved = oRecord.GetContentsByName("Name")
Dim sContactFirstNameSaved : sContactFirstNameSaved =
oRecord.GetContentsByName("Firstname")
Set oRecord = Nothing

If ((sContactName & sContactFirstName) <> (sContactNameSaved &
sContactFirstNameSaved)) Then
    Call cRM.DialogMessageBox("Die aktuellen Informationen über den Namen des
Kontakte-Datensatz unterscheiden sich von den zuvor gespeicherten Informationen.",
"InputForm.GetContentsByName", vbOkOnly)
End If
```

Beispiel C#-Script:

```
// Prüft, ob die Informationen im Feld Name innerhalb der Eingabemaske verändert
wurden. Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-
Solution

// <!--#pragma keepeditmode-->
```

```
InputForm inputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2);
string contactName = inputForm.GetContentsByName("Name");
string contactFirstName = inputForm.GetContentsByName("Firstname");
inputForm.Dispose();

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
string contactNameSaved = record.GetContentsByName("Name");
string contactFirstNameSaved = record.GetContentsByName("Firstname");
record.Dispose();

if (contactFirstName + contactName != contactFirstNameSaved + contactNameSaved)
{
    cRM.DialogMessageBox("Die aktuellen Informationen über den Namen des Kontakte-
Datensatz unterscheiden sich von den zuvor gespeicherten Informationen.",
"InputForm.GetContentsByName", 0);
}
```

GetContentsValueByName

Beschreibung:

Liefert den Inhalt entsprechend des Feldtyps des Feldes zurück, dessen Feldname übergeben wurde, z. B. Datumzeit-Typen als Datumsvariable, numerische Typen als numerische Variable etc. Somit werden bspw. Lokalisierungsprobleme (Komma oder Punkt als Dezimalzeichen? Datumsformatierung?) bei der Weiterverarbeitung des Wertes vermieden.

Hinweis: Wird ein formatiertes Notizenfeld abgerufen, so wird immer der entsprechende HTML-Code zurückgeliefert.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes.

Rückgabewert:

Variant

Beispiel:

Hinweis: Bitte beachten Sie, dass "" nicht gleich NULL ist. NULL Werte können wie folgt geprüft werden.

Für ein Beispiel siehe Methode **GetContentsValueByName** des Record-Objektes.

Beispiel VBScript:

```
' Prüft, ob die Informationen im Feld Name innerhalb der Eingabemaske verändert
wurden. Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-
Solution

<!--#pragma keepeditmode-->
Dim oInputForm : Set oInputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2)
Dim sContactName : sContactName = oInputForm.GetContentsValueByName("Name")
Dim sContactFirstName : sContactFirstName =
oInputForm.GetContentsValueByName("Firstname")
Set oInputForm = Nothing

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Dim sContactNameSaved : sContactNameSaved = oRecord.GetContentsValueByName("Name")
```

```
Dim sContactFirstNameSaved : sContactFirstNameSaved =
oRecord.GetContentsValueByName("Firstname")
Set oRecord = Nothing

If ((sContactName & sContactFirstName) <> (sContactNameSaved &
sContactFirstNameSaved)) Then
    Call cRM.DialogMessageBox("Die aktuellen Informationen über den Namen des
Kontakte-Datensatz unterscheiden sich von den zuvor gespeicherten Informationen.",
"InputForm.GetContentsValueByName", vbOkOnly)
End If
```

Beispiel C#-Script:

```
// Prüft, ob die Informationen im Feld Name innerhalb der Eingabemaske verändert
wurden. Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-
Solution

// <!--#pragma keepeditmode-->
InputForm inputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2);
var contactName = inputForm.GetContentsValueByName("Name");
var contactFirstName = inputForm.GetContentsValueByName("Firstname");
inputForm.Dispose();

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
var contactNameSaved = record.GetContentsValueByName("Name");
var contactFirstNameSaved = record.GetContentsValueByName("Firstname");
record.Dispose();

if (contactFirstName + contactName != contactFirstNameSaved + contactNameSaved)
{
    cRM.DialogMessageBox("Die aktuellen Informationen über den Namen des Kontakte-
Datensatz unterscheiden sich von den zuvor gespeicherten Informationen.",
"InputForm.GetContentsByName", 0);
}
```

GetHwndByName

Beschreibung:

Gibt das Handle des übergebenen Feldes zurück.

Parameter:

Parametername	Typ	Beschreibung
FieldName	String	Physikalischer Name des gewünschten Feldes.

Rückgabewert:

Long (VBScript) / IntPtr (C#Script)

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

<!--#pragma keepeditmode-->
Dim nHwndByName : nHwndByName =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).GetHwndByName("First
name")
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

// <!--#pragma keepeditmode-->
InputForm inputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2);
IntPtr hwndByName  = inputForm.GetHwndByName("Name");
```

InvokeContextMenu

Beschreibung:

Mit dieser Funktion ist es möglich Kontext-Menü-Befehle für ein bestimmtes Feld auszulösen. Die Menü IDs sind im Kapitel **Menü-IDs** dokumentiert.

Hinweis: Es werden nur Menü-IDs von Kontext-Menüs unterstützt. Sollte die Methode in einem asynchron ausgeführten Script ausgeführt werden, so ist der Rückgabewert immer True. Der Rückgabewert beschreibt, ob der Aufruf übermittelt werden konnte, nicht jedoch, ob in der aufzurufenden Funktion ggf. ein Problem festgestellt wurde.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes.
Action	Long	Menü ID der auszuführenden Aktion

Rückgabewert:

Bool

Wert	Beschreibung
True	Befehl zum Aufrufen eines Menüeintrags wurde erfolgreich an combit CRM übermittelt.
False	Befehl zum Aufrufen eines Menüeintrags konnte nicht übermittelt werden. Dies kann z. B. der Fall sein, wenn der aufzurufende Menü-Befehl derzeit nicht zur Verfügung steht.

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die Aktivitäten-Ansicht einer combit_Large-
Solution

<!--#pragma keepeditmode-->
Dim oInputForm : Set oInputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2)
Dim nMenuIDOpenDocument : nMenuIDOpenDocument = 33034
Call oInputForm.InvokeContextMenu("Document_Embedded", nMenuIDOpenDocument)
Set oInputForm = Nothing
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist die Aktivitäten-Ansicht einer combit_Large-
Solution

// <!--#pragma keepeditmode-->
InputForm inputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2);
long menuIDOpenDocument = 33034;
inputForm.InvokeContextMenu("Document_Embedded", menuIDOpenDocument);
inputForm.Dispose();
```

Save

Beschreibung:

Speichert die Inhalte der Eingabemaske ohne Rückfrage, ob Änderungen gespeichert werden sollen.

Rückgabewert:

Bool

Beispiel VBSript:

```
' Speichert die Eingabemaske nachdem der Nutzer einen neuen Namen für den
dargestellten Kontakte-Datensatz hinterlegt hat. Basis für dieses Beispiel ist die
Kontakte-Ansicht einer combit_Large-Solution

<!--#pragma keepeditmode-->
Dim oInputForm : Set oInputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2)
Dim sUserInput : sUserInput = cRM.DialogInputBox("Welcher Name soll für den
Kontakte-Datensatz hinterlegt werden?", "InputForm.Save", "Soleil")

If (sUserInput <> "$CANCEL$" And Len(sUserInput) > 0) Then
    Call oInputForm.SetContentsByName("Name", sUserInput)

    If (oInputForm.Save()) Then
        Call cRM.DialogMessageBox("Der eingegebene Inhalt konnte erfolgreich
gespeichert werden.", "InputForm.Save", vbOkOnly)
    End If
End If

Set oInputForm = Nothing
```

Beispiel C#-Script:

```
// Speichert die Eingabemaske nachdem der Nutzer einen neuen Namen für den
dargestellten Kontakte-Datensatz hinterlegt hat. Basis für dieses Beispiel ist die
Kontakte-Ansicht einer combit_Large-Solution

// <!--#pragma keepeditmode-->
InputForm inputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2);
string userInput = cRM.DialogInputBox("Welcher Name soll für den Kontakte-
Datensatz hinterlegt werden?", "InputForm.Save", "Soleil");

if (userInput != "$CANCEL$" && userInput != "")
{
    inputForm.SetContentsByName("Name", userInput);

    if (inputForm.Save() == true)
    {
        cRM.DialogMessageBox("Der eingegebene Inhalt konnte erfolgreich
gespeichert werden.", "InputForm.Save", 0);
    }
}

inputForm.Dispose();
```

SetContentsByName

Beschreibung:

Legt den Inhalt des Feldes fest, dessen physikalischer Feldname übergeben wurde.

Hinweis: Wird nur ausgeführt, falls sich das **InputForm**-Objekt im Bearbeiten-Modus befindet!

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes in der verwendeten/aktiven Eingabemaske.
Contents	String	(Neuer) Feldinhalt

		Der Inhalt von Datumsfeldern muss im Format YYYYMMDD übergeben werden, bei Datumsfeldern mit Zeitanteil ist es YYYYMMDDHHMMSS

Rückgabewert:

Bool

Beispiel VBScript:

```
' Speichert die Eingabemaske nachdem der Nutzer einen neuen Namen für den
dargestellten Kontakte-Datensatz hinterlegt hat. Basis für dieses Beispiel ist die
Kontakte-Ansicht einer combit_Large-Solution

<!--#pragma keepeditmode-->
Dim oInputForm : Set oInputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2)
Dim sUserInput : sUserInput = cRM.DialogInputBox("Welcher Name soll für den
Kontakte-Datensatz hinterlegt werden?", "InputForm.Save", "Soleil")

If (sUserInput <> "$CANCEL$" And Len(sUserInput) > 0) Then
    Call oInputForm.SetContentsByName("Name", sUserInput)

    If (oInputForm.Save()) Then
        Call cRM.DialogMessageBox("Der eingegebene Inhalt konnte erfolgreich
gespeichert werden.", "InputForm.Save", vbOkOnly)
    End If
End If

Set oInputForm = Nothing
```

Beispiel C#-Script:

```
// Speichert die Eingabemaske nachdem der Nutzer einen neuen Namen für den
dargestellten Kontakte-Datensatz hinterlegt hat. Basis für dieses Beispiel ist die
Kontakte-Ansicht einer combit_Large-Solution

// <!--#pragma keepeditmode-->
InputForm inputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2);
string userInput = cRM.DialogInputBox("Welcher Name soll für den Kontakte-
Datensatz hinterlegt werden?", "InputForm.Save", "Soleil");

if (userInput != "$CANCEL$" && userInput != "")
{
    inputForm.SetContentsByName("Name", userInput);

    if (inputForm.Save() == true)
    {
        cRM.DialogMessageBox("Der eingegebene Inhalt konnte erfolgreich
gespeichert werden.", "InputForm.Save", 0);
    }
}

inputForm.Dispose();
```

SetContentsValueByName

Beschreibung:

Legt den Inhalt des Feldes fest, dessen physikalischer Feldname übergeben wurde.

Hinweis: Wird nur ausgeführt, falls sich das **InputForm**-Objekt im Bearbeiten-Modus befindet!

Die übergebene Variable für den Inhalt kann dabei einen zum Feldtyp korrespondierenden Typ haben und muss nicht vorher in eine Zeichenkette umgewandelt werden. Somit werden bspw. Lokalisierungsprobleme (Komma oder Punkt als Dezimalzeichen? Datumsformatierung?) bei der Weiterverarbeitung des Wertes vermieden.

Das Setzen eines Primärschlüssels ist möglich, sofern das Schreiben erlaubt ist.

Hinweis: Bitte prüfen Sie den Rückgabewert der Methode, um sicherzustellen, dass das Setzen des neuen Inhalts funktioniert hat.

Wird ein formatiertes Notizenfeld gesetzt, ist folgendes zu beachten: fängt der Inhalt mit \\plaintext: an, dann wird Klartext angenommen, fängt er mit \\html: an, dann wird HTML Inhalt angenommen, wird kein Präfix übergeben, so wird HTML angenommen.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes.
Contents	Variant	(Neuer) Feldinhalt

Rückgabewert:

Bool

Beispiel:

Hinweis: Bitte beachten Sie, dass "" nicht gleich NULL ist. NULL Werte können wie folgt geprüft werden.

Beispiel VBScript:

```
' Speichert die Eingabemaske nachdem der Nutzer einen neuen Namen für den
dargestellten Kontakte-Datensatz hinterlegt hat. Basis für dieses Beispiel ist die
Kontakte-Ansicht einer combit_Large-Solution

Dim oInputForm : Set oInputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2)
Dim sUserInput : sUserInput = cRM.DialogInputBox("Welcher Name soll für den
Kontakte-Datensatz hinterlegt werden?", "InputForm.Save", "Soleil")

If (sUserInput <> "$CANCEL$" And Len(sUserInput) > 0) Then
    Call oInputForm.SetContentsValueByName("Name", sUserInput)

    If (oInputForm.Save()) Then
        Call cRM.DialogMessageBox("Der eingegebene Inhalt konnte erfolgreich
gespeichert werden.", "InputForm.Save", vbOkOnly)
    End If
End If

Set oInputForm = Nothing
```

Beispiel C#-Script:

```
// Speichert die Eingabemaske nachdem der Nutzer einen neuen Namen für den
dargestellten Kontakte-Datensatz hinterlegt hat. Basis für dieses Beispiel ist die
Kontakte-Ansicht einer combit_Large-Solution

// <!--#pragma keepeditmode-->
InputForm inputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2);
string userInput = cRM.DialogInputBox("Welcher Name soll für den Kontakte-
Datensatz hinterlegt werden?", "InputForm.Save", "Soleil");

if (userInput != "$CANCEL$" && userInput != "")
{
```

```
    inputForm.SetContentsValueByName("Name", userInput);

    if (inputForm.Save() == true)
    {
        cRM.DialogMessageBox("Der eingegebene Inhalt konnte erfolgreich
gespeichert werden.", "InputForm.Save", 0);
    }
}

inputForm.Dispose();
```

SetFocusToField

Beschreibung:

Diese Methode setzt den Eingabecursor in ein bestimmtes Feld.

> **Hinweis**: Wird nur ausgeführt, falls das **InputForm**-Objekt sich im Bearbeiten-Modus befindet!

Parameter:

Parametername	Typ	Beschreibung
FieldName	String	Name des gewünschten Feldes.
SelectText	Long	Wenn SelectText auf 1 (True) gesetzt wurde, wird der bereits existierende Text in dem Feld auch selektiert.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(0).SetFocusToField("Nam
e", 0)
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(0).SetFocusToField("Nam
e", 0);
```

ShowField

Beschreibung:

Stellt das übergebene Feld dar, sofern es in der Eingabemaske platziert wurde, ggf. wird hierzu auch eine Karteikarten-Seite umgeschaltet. Falls das Feld mehrfach platziert wurde, wird das erste Vorkommen dargestellt.

Parameter:

Parametername	Typ	Beschreibung
FieldName	String	Name des gewünschten Feldes. Zum Anzeigen von Containern muss der Feldname in der Form 'Schlüsselfeld.Zielansicht.Fremdschlüsselfeld' übergeben werden, also z. B. 'ID.Kontakte.CompanyID' für den Container mit den Kontakten der Firma.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).ShowField("Name")
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).ShowField("Name");
```

Update

Beschreibung:

Aktualisiert die Eingabemaske und Datenbankinhalte für den aktuellen Datensatz. Es erfolgt keine Veränderung der Position im RecordSet (verglichen mit **View.Update**).

Hinweis: Diese Methode dient dazu, etwaige zwischenzeitliche Änderungen am dargestellten Datensatz forciert neu darzustellen – insbesondere, wenn diese Änderungen nur auf dem Datenbankserver stattfanden (z. B. durch den Einsatz eines Triggers). Es wird keine Aktion durchgeführt, wenn sich die Eingabemaske im Bearbeiten-Modus befindet.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Update()
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Update();
```

UpdateAllContainers

Beschreibung:

Führt für alle sichtbaren Container eine Aktualisierung in der Ansicht durch. Der Datensatz der Ansicht selbst wird dabei nicht explizit aktualisiert (siehe **View.Update**). Der aktuelle Filter und die aktuelle Position innerhalb der Datensätze der Ansicht werden somit nicht verändert.

Einsatzbereich: In einem Script wird ein relationaler Datensatz erzeugt (z. B. Kontaktcontainer-Eintrag), dessen Existenz im Container jetzt durch das Script visuell forciert werden soll, ohne Filter und Position innerhalb der Datensätze der Ansicht zu verändern.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).UpdateAllContainers(
)
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).UpdateAllContainers(
);
```

3.24 Link Objekt

3.24.1 Eigenschaften

DisplayText, read-only

Beschreibung:

Liefert den Anzeigenamen der Verknüpfung zurück.

Typ:

String

Beispiel VBScript:

```
Dim sDisplayText : sDisplayText = oLink.DisplayText
```

Beispiel C#-Script:

```
string displayText = link.DisplayText;
```

PrimaryKeyFldContent, read-only

Beschreibung:

Liefert den Inhalt des Primärschlüsselfelds der Verknüpfung zurück.

Typ:

String

Beispiel VBScript:

```
Dim sPrimaryKeyFldContent : sPrimaryKeyFldContent = oLink.PrimaryKeyFldContent
```

Beispiel C#-Script:

```
string primaryKeyFldContent = link.PrimaryKeyFldContent;
```

PrimaryKeyFldName, read-only

Beschreibung:

Liefert den Name des Primärschlüsselfelds der Verknüpfung zurück.

Typ:

String

Beispiel VBScript:

```
Dim sPrimaryKeyFldName : sPrimaryKeyFldName = oLink.PrimaryKeyFldName
```

Beispiel C#-Script:

```
string primaryKeyFldName = link.PrimaryKeyFldName;
```

Project, read-only

Beschreibung:

Liefert das Projekt der Verknüpfung zurück.

Typ:

String

Beispiel VBScript:

```
Dim sProject : sProject = oLink.Project
```

Beispiel C#-Script:

```
string project = link.Project;
```

ViewName, read-only

Beschreibung:

Liefert den Namen der Ansicht der Verknüpfung zurück.

Typ:

String

Beispiel VBScript:

```
Dim sViewName : sViewName = oLink.ViewName
```

Beispiel C#-Script:

```
string viewName = link.ViewName;
```

ViewFamily, read-only

Beschreibung:

Liefert den Familiennamen der Ansicht der Verknüpfung zurück.

Typ:

String

Beispiel VBScript:

```
Dim sViewFamily : sViewFamily = oLink.ViewFamily
```

Beispiel C#-Script:

```
string viewFamily = link.ViewFamily;
```

3.24.2 Methoden

GetLinkAsString

Beschreibung:

Liefert die Verknüpfung als Zeichenfolge zurück.

Rückgabewert:

String

Beispiel VBScript:

```
Dim sLink : sLink = oLink.GetLinkAsString()
```

Beispiel C#-Script:

```
string link = link.GetLinkAsString();
```

SetLinkFromString

Beschreibung:

Setzt einen Link anhand einer Zeichenfolge.

Parameter:

Parametername	Typ	Beschreibung
Link	String	Link-Zeichenfolge. Einzelne Parameter des Links werden wie folgt durch '|' getrennt: <ProjectID>|<ViewName>|<ViewFamily>|<PKFld-Name>|<PKContent>|<DisplayText>

Beispiel VBScript:

```
Call oLink.SetLinkFromString(sProjectID, sViewName, sViewFamily, sPKFldName,
sPKFldContent, sDisplayText)
```

Beispiel C#-Script:

```
link.SetLinkFromString(projectID, viewName, viewFamily, pkFldName, pkFldContent,
displayText);
```

3.25 Links Objekt

3.25.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der Einträge in der Sammlung zurück.

Typ:

Long

Beispiel VBScript:

```
Dim nLinkCount : nLinkCount = oLinks.Count
```

Beispiel C#-Script:

```
long linkCount = oLinks.Count;
```

3.25.2 Methoden

Add

Beschreibung:

Fügt eine neue Verknüpfung zu einer Aufgabe/einem Termin hinzu.

Parameter:

Parametername	Typ	Beschreibung
Link	Link	Link Objekt.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call oLinks.Add(oLink)
```

Beispiel C#-Script:

```
links.Add(link);
```

Item

Beschreibung:

Gibt eine Verknüpfung zurück. Es muss die Index-Nummer der Verknüpfung übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Link
NULL (wenn die Verknüpfung nicht existiert)

Beispiel VBScript:

```
Dim oLink : Set oLink = oLinks.Item(nIndex)
```

Beispiel C#-Script:

```
Link link = links.Item(index);
```

NewLink

Beschreibung:

Liefert eine neue Verknüpfung vom Typ **Link** zurück.

Rückgabewert:

Link

Beispiel VBScript:

```
Dim oLink : Set oLink = oLinks.NewLink()
```

Beispiel C#-Script:

```
Link link = links.NewLink();
```

Remove

Beschreibung:

Löscht eine Verknüpfung. Es muss die Index-Nummer der Verknüpfung übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oLink : Set oLink = oLinks.Remove(nIndex)
```

Beispiel C#-Script:

```
Link link = links.Remove(index);
```

3.26 ListAddressInfos Objekt

Liste aller konfigurierten Adressen (Ansichtseigenschaften der betreffenden Ansicht, Reiter Adressen) einer Ansicht.

3.26.1 Eigenschaften

DefaultISOCountry, read-only

Beschreibung:

Liefert das in der Länderkonfiguration (Ansichteneigenschaften > Adressen > Länderkürzel konfigurieren...) für die Option "Wenn Land leer" ausgewählte Land im "ISO 3166-1 alpha-2"-Format zurück. Wenn die Auswahl auf "Ignorieren" steht, wird eine leere Zeichenkette zurückgegeben.

Typ:

String

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("AddressInfo.DefaultISOCountry: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").DefaultISOCountry, "AddressInfo.DefaultISOCountry", vbOkOnly)
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox("AddressInfo.DefaultISOCountry: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").DefaultISOCountry, "AddressInfo.DefaultISOCountry", 0);
```

Count, read-only

Beschreibung:

Liefert die Anzahl der konfigurierten Adressen der Ansicht zurück.

Typ:

Long

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("AddressInfo.Count: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").Count, "AddressInfo.Count", vbOkOnly)
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox("AddressInfo.Count: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").Count, "AddressInfo.Count", 0);
```

3.26.2 Methoden

GetISOCountryFromUserCountryCode

Beschreibung:

Liefert das Land im "ISO 3166-1 alpha-2"-Format für den Eintrag in der Länderkonfiguration (Ansichteneigenschaften > Adressen > Länderkürzel konfigurieren...) zurück, der das übergebene Länderkürzel enthält. Wenn das übergebene Länderkürzel nicht gefunden werden kann, wird eine leere Zeichenkette zurückgegeben.

Parameter:

Parametername	Typ	Beschreibung
sCountryCode	String	Länderkürzel

Rückgabewert:

String

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

Call
cRM.DialogMessageBox(cRM.CurrentProject.ActiveViews.ActiveView.Config.AddressInfos
.GetISOCountryFromUserCountryCode("D"),
"AddressInfo.GetISOCountryFromUserCountryCode", vbOkOnly)
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox("AddressInfo.GetISOCountryFromUserCountryCode: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").GetISOCountryFromUserCountryCode("D"),
"AddressInfo.GetISOCountryFromUserCountryCode", 0);
```

GetUserCountryCodesFromISOCountry

Beschreibung:

Liefert alle definierten Kürzel (semikolonsepariert) für das Land in der Länderkonfiguration (Ansichteneigenschaften > Adressen > Länderkürzel konfigurieren...) zurück, das dem übergebenen Länderkürzel entspricht. Wenn das übergebene Länderkürzel keinem Land zugeordnet werden kann, wird eine leere Zeichenkette zurückgegeben.

Parameter:

Parametername	Typ	Beschreibung
sCountryCode	String	Länderkürzel im "ISO 3166-1 alpha-2"-Format, z. B. "DE"

Rückgabewert:

String

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

Call
cRM.DialogMessageBox(cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").Address
Infos.ItemByName("Kontakte_Adresse").GetUserCountryCodesFromISOCountry("DE"),
"AddressInfo.GetUserCountryCodesFromISOCountry", vbOkOnly)
```

Beispiel C#-Script:

```
// Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox("AddressInfo.GetUserCountryCodesFromISOCountry: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos.ItemByName("Kon
takte_Adresse").GetUserCountryCodesFromISOCountry,
"AddressInfo.GetUserCountryCodesFromISOCountry", 0);
```

Item

Beschreibung:

Gibt eine **AddressInfo** entsprechend dem übergebenen Index zurück. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

AddressInfo

Beispiel VBScript:

```
Dim oAddressInfo : Set oAddressInfo = oAddressInfos.Item(nIndex)
```

Beispiel C#-Script:

```
AddressInfo addressInfo = addressInfos.Item(index);
```

ItemByName

Beschreibung:

Gibt eine **AddressInfo** entsprechend dem übergebenen Namen zurück.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name der Adresse.

Rückgabewert:

AddressInfo

Beispiel VBScript:

```
Dim oAddressInfo : Set oAddressInfo = oAddressInfos.ItemByName(sName)
```

Beispiel C#-Script:

```
AddressInfo addressInfo = addressInfos.ItemByName(name);
```

ItemByID

Beschreibung:

Gibt eine **AddressInfo** entsprechend der übergebenen ID zurück.

Parameter:

Parametername	Typ	Beschreibung
Name	String	ID der Adresse.

Rückgabewert:

AddressInfo

Beispiel VBScript:

```
Dim oAddressInfo : Set oAddressInfo = oAddressInfos.ItemByID(sID)
```

Beispiel C#-Script:

```
AddressInfo addressInfo = addressInfos.ItemByID(id);
```

3.27 ListCodeDefinitions Objekt

Collection von Codedefinitionen eines Feldes.

3.27.1 Methoden

Count

Beschreibung:

Liefert die Anzahl der Codedefinitionen eines Feldes.

Rückgabewert:

Integer

Beispiel VBScript:

```
' Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

Dim nCodeDefinitionsCount : nCodeDefinitionsCount =
oCodeDefinitions.Count("Category")
```

Beispiel C#-Script:

```
' Basis für dieses Beispiel ist die Kontakte-Ansicht einer combit_Large-Solution

long codeDefinitionsCount = codeDefinitions.Count("Category");
```

Item

Beschreibung:

Liefert die Codebeschreibung für den Eintrag mit dem übergebenen Index zurück. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Integer	Index-Nummer.

Rückgabewert:

String

Beispiel VBScript:

```
Dim sCodeDefinition : sCodeDefinition = oCodeDefinitions.Item(nIndex)
```

Beispiel C#-Script:

```
string codeDefinition = codeDefinitions.Item(index);
```

3.28 ListCompanyInfoUserDefined Objekt

Liste aller benutzerdefinierten Felder der Firmenstammdaten.

3.28.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der Einträge in der Sammlung zurück.

Typ:

Long

Beispiel VBScript:

```
' Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("AccountNo", oCompanyInfo.AccountNo)
Call dicCompanyInfo.Add("Bank", oCompanyInfo.Bank)
Call dicCompanyInfo.Add("BankCode", oCompanyInfo.BankCode)
Call dicCompanyInfo.Add("City", oCompanyInfo.City)
Call dicCompanyInfo.Add("Company", oCompanyInfo.Company)
Call dicCompanyInfo.Add("Company2", oCompanyInfo.Company2)
Call dicCompanyInfo.Add("Company3", oCompanyInfo.Company3)
Call dicCompanyInfo.Add("Country", oCompanyInfo.Country)
Call dicCompanyInfo.Add("Email", oCompanyInfo.Email)
Call dicCompanyInfo.Add("Extra1", oCompanyInfo.Extra1)
Call dicCompanyInfo.Add("Extra2", oCompanyInfo.Extra2)
Call dicCompanyInfo.Add("Fax", oCompanyInfo.Fax)
Call dicCompanyInfo.Add("IBAN", oCompanyInfo.IBAN)
Call dicCompanyInfo.Add("Internet", oCompanyInfo.Internet)
Call dicCompanyInfo.Add("Logo", oCompanyInfo.Logo)
Call dicCompanyInfo.Add("Phone", oCompanyInfo.Phone)
Call dicCompanyInfo.Add("Street", oCompanyInfo.Street)
Call dicCompanyInfo.Add("VatID", oCompanyInfo.VatID)
Call dicCompanyInfo.Add("ZIP", oCompanyInfo.ZIP)

Dim oListCompanyInfoUserDefined : Set oListCompanyInfoUserDefined =
oCompanyInfo.UserDefinedFields
Dim oCompanyInfoUserDefinedItem
Dim nCount : nCount = 0

For nCount = 1 To oListCompanyInfoUserDefined.Count

    Set oCompanyInfoUserDefinedItem = oListCompanyInfoUserDefined.Item(nCount)

    If (oCompanyInfoUserDefinedItem.Type = 2) Then ' Wenn es eine Textinformation
ist
        Call dicCompanyInfo.Add(oCompanyInfoUserDefinedItem.Key,
oCompanyInfoUserDefinedItem.Value)
    End If

    Set oCompanyInfoUserDefinedItem = Nothing

Next

Set oListCompanyInfoUserDefined = Nothing

Dim sFullCompanyInfo : sFullCompanyInfo = ""
Dim dicItem
```

```
For Each dicItem In dicCompanyInfo
    If (Len(dicCompanyInfo(dicItem)) > 0) Then
        sFullCompanyInfo = sFullCompanyInfo & dicItem & ": " &
dicCompanyInfo(dicItem) & vbCrlf
    End If
Next

WScript.ClipboardText = sFullCompanyInfo

Call cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die
Zwischenablage übernommen.", "CompanyInfo", vbOkOnly)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
// Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("AccountNo", companyInfo.AccountNo);
dictionaryCompanyInfo.Add("Bank", companyInfo.Bank);
dictionaryCompanyInfo.Add("BankCode", companyInfo.BankCode);
dictionaryCompanyInfo.Add("City", companyInfo.City);
dictionaryCompanyInfo.Add("Company", companyInfo.Company);
dictionaryCompanyInfo.Add("Company2", companyInfo.Company2);
dictionaryCompanyInfo.Add("Company3", companyInfo.Company3);
dictionaryCompanyInfo.Add("Country", companyInfo.Country);
dictionaryCompanyInfo.Add("Email", companyInfo.Email);
dictionaryCompanyInfo.Add("Extra1", companyInfo.Extra1);
dictionaryCompanyInfo.Add("Extra2", companyInfo.Extra2);
dictionaryCompanyInfo.Add("Fax", companyInfo.Fax);
dictionaryCompanyInfo.Add("IBAN", companyInfo.IBAN);
dictionaryCompanyInfo.Add("Internet", companyInfo.Internet);
dictionaryCompanyInfo.Add("Logo", companyInfo.Logo);
dictionaryCompanyInfo.Add("Phone", companyInfo.Phone);
dictionaryCompanyInfo.Add("Street", companyInfo.Street);
dictionaryCompanyInfo.Add("VatID", companyInfo.VatID);
dictionaryCompanyInfo.Add("ZIP", companyInfo.ZIP);

ListCompanyInfoUserDefined listCompanyInfoUserDefined =
companyInfo.UserDefinedFields;

foreach (CompanyInfoUserDefinedItem companyInfoUserDefinedItem in
listCompanyInfoUserDefined)
{
    if (companyInfoUserDefinedItem.Type == 2)
    {
        dictionaryCompanyInfo.Add(companyInfoUserDefinedItem.Key,
companyInfoUserDefinedItem.Value);
    }
}

string fullCompanyInfo = string.Empty;

foreach (var dictionaryItem in dictionaryCompanyInfo)
{
    if (dictionaryItem.Value != "")
    {
        fullCompanyInfo = fullCompanyInfo + dictionaryItem.Key + ": " +
dictionaryItem.Value + System.Environment.NewLine;
    }
}

WScript.ClipboardText = fullCompanyInfo;
cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die Zwischenablage
übernommen.", "CompanyInfo", 0);
```

```
companyInfo.Dispose();
```

3.28.2 Methoden

Item

Beschreibung:

Gibt ein benutzerdefiniertes Feld der Firmenstammdaten zurück. Es muss die Index-Nummer des Feldes übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

CompanyInfoUserDefinedItem

Beispiel VBScript:

```
' Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("AccountNo", oCompanyInfo.AccountNo)
Call dicCompanyInfo.Add("Bank", oCompanyInfo.Bank)
Call dicCompanyInfo.Add("BankCode", oCompanyInfo.BankCode)
Call dicCompanyInfo.Add("City", oCompanyInfo.City)
Call dicCompanyInfo.Add("Company", oCompanyInfo.Company)
Call dicCompanyInfo.Add("Company2", oCompanyInfo.Company2)
Call dicCompanyInfo.Add("Company3", oCompanyInfo.Company3)
Call dicCompanyInfo.Add("Country", oCompanyInfo.Country)
Call dicCompanyInfo.Add("Email", oCompanyInfo.Email)
Call dicCompanyInfo.Add("Extra1", oCompanyInfo.Extra1)
Call dicCompanyInfo.Add("Extra2", oCompanyInfo.Extra2)
Call dicCompanyInfo.Add("Fax", oCompanyInfo.Fax)
Call dicCompanyInfo.Add("IBAN", oCompanyInfo.IBAN)
Call dicCompanyInfo.Add("Internet", oCompanyInfo.Internet)
Call dicCompanyInfo.Add("Logo", oCompanyInfo.Logo)
Call dicCompanyInfo.Add("Phone", oCompanyInfo.Phone)
Call dicCompanyInfo.Add("Street", oCompanyInfo.Street)
Call dicCompanyInfo.Add("VatID", oCompanyInfo.VatID)
Call dicCompanyInfo.Add("ZIP", oCompanyInfo.ZIP)

Dim oListCompanyInfoUserDefined : Set oListCompanyInfoUserDefined =
oCompanyInfo.UserDefinedFields
Dim oCompanyInfoUserDefinedItem
Dim nCount : nCount = 0

For nCount = 1 To oListCompanyInfoUserDefined.Count

    Set oCompanyInfoUserDefinedItem = oListCompanyInfoUserDefined.Item(nCount)

    If (oCompanyInfoUserDefinedItem.Type = 2) Then ' Wenn es eine Textinformation
ist
        Call dicCompanyInfo.Add(oCompanyInfoUserDefinedItem.Key,
oCompanyInfoUserDefinedItem.Value)
    End If

    Set oCompanyInfoUserDefinedItem = Nothing

Next

Set oListCompanyInfoUserDefined = Nothing
```

```
Dim sFullCompanyInfo : sFullCompanyInfo = ""
Dim dicItem

For Each dicItem In dicCompanyInfo
    If (Len(dicCompanyInfo(dicItem)) > 0) Then
        sFullCompanyInfo = sFullCompanyInfo & dicItem & ": " &
dicCompanyInfo(dicItem) & vbCrlf
    End If
Next

WScript.ClipboardText = sFullCompanyInfo

Call cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die
Zwischenablage übernommen.", "CompanyInfo", vbOkOnly)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
// Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("AccountNo", companyInfo.AccountNo);
dictionaryCompanyInfo.Add("Bank", companyInfo.Bank);
dictionaryCompanyInfo.Add("BankCode", companyInfo.BankCode);
dictionaryCompanyInfo.Add("City", companyInfo.City);
dictionaryCompanyInfo.Add("Company", companyInfo.Company);
dictionaryCompanyInfo.Add("Company2", companyInfo.Company2);
dictionaryCompanyInfo.Add("Company3", companyInfo.Company3);
dictionaryCompanyInfo.Add("Country", companyInfo.Country);
dictionaryCompanyInfo.Add("Email", companyInfo.Email);
dictionaryCompanyInfo.Add("Extra1", companyInfo.Extra1);
dictionaryCompanyInfo.Add("Extra2", companyInfo.Extra2);
dictionaryCompanyInfo.Add("Fax", companyInfo.Fax);
dictionaryCompanyInfo.Add("IBAN", companyInfo.IBAN);
dictionaryCompanyInfo.Add("Internet", companyInfo.Internet);
dictionaryCompanyInfo.Add("Logo", companyInfo.Logo);
dictionaryCompanyInfo.Add("Phone", companyInfo.Phone);
dictionaryCompanyInfo.Add("Street", companyInfo.Street);
dictionaryCompanyInfo.Add("VatID", companyInfo.VatID);
dictionaryCompanyInfo.Add("ZIP", companyInfo.ZIP);

ListCompanyInfoUserDefined listCompanyInfoUserDefined =
companyInfo.UserDefinedFields;

foreach (CompanyInfoUserDefinedItem companyInfoUserDefinedItem in
listCompanyInfoUserDefined)
{
    if (companyInfoUserDefinedItem.Type == 2)
    {
        dictionaryCompanyInfo.Add(companyInfoUserDefinedItem.Key,
companyInfoUserDefinedItem.Value);
    }
}

string fullCompanyInfo = string.Empty;

foreach (var dictionaryItem in dictionaryCompanyInfo)
{
    if (dictionaryItem.Value != "")
    {
        fullCompanyInfo = fullCompanyInfo + dictionaryItem.Key + ": " +
dictionaryItem.Value + System.Environment.NewLine;
    }
}
```

```
WScript.ClipboardText = fullCompanyInfo;
cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die Zwischenablage
übernommen.", "CompanyInfo", 0);

companyInfo.Dispose();
```

ItemByName

Beschreibung:

Liefert ein Objekt vom Typ **CompanyInfoUserDefinedItem** mit dem angegebenen Namen zurück.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name des gewünschten Feldes.

Rückgabewert:

CompanyInfoUserDefinedItem

Beispiel VBScript:

```
' Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("AccountNo", oCompanyInfo.AccountNo)
Call dicCompanyInfo.Add("Bank", oCompanyInfo.Bank)
Call dicCompanyInfo.Add("BankCode", oCompanyInfo.BankCode)
Call dicCompanyInfo.Add("City", oCompanyInfo.City)
Call dicCompanyInfo.Add("Company", oCompanyInfo.Company)
Call dicCompanyInfo.Add("Company2", oCompanyInfo.Company2)
Call dicCompanyInfo.Add("Company3", oCompanyInfo.Company3)
Call dicCompanyInfo.Add("Country", oCompanyInfo.Country)
Call dicCompanyInfo.Add("Email", oCompanyInfo.Email)
Call dicCompanyInfo.Add("Extra1", oCompanyInfo.Extra1)
Call dicCompanyInfo.Add("Extra2", oCompanyInfo.Extra2)
Call dicCompanyInfo.Add("Fax", oCompanyInfo.Fax)
Call dicCompanyInfo.Add("IBAN", oCompanyInfo.IBAN)
Call dicCompanyInfo.Add("Internet", oCompanyInfo.Internet)
Call dicCompanyInfo.Add("Logo", oCompanyInfo.Logo)
Call dicCompanyInfo.Add("Phone", oCompanyInfo.Phone)
Call dicCompanyInfo.Add("Street", oCompanyInfo.Street)
Call dicCompanyInfo.Add("VatID", oCompanyInfo.VatID)
Call dicCompanyInfo.Add("ZIP", oCompanyInfo.ZIP)

Dim oListCompanyInfoUserDefined : Set oListCompanyInfoUserDefined =
oCompanyInfo.UserDefinedFields
Dim oCompanyInfoUserDefinedItem
Dim nCount : nCount = 0

For nCount = 1 To oListCompanyInfoUserDefined.Count

    Set oCompanyInfoUserDefinedItem = oListCompanyInfoUserDefined.Item(nCount)

    If (oCompanyInfoUserDefinedItem.Type = 2) Then ' Wenn es eine Textinformation
ist
        Call dicCompanyInfo.Add(oCompanyInfoUserDefinedItem.Key,
oCompanyInfoUserDefinedItem.Value)
    End If

    Set oCompanyInfoUserDefinedItem = Nothing

Next

Set oListCompanyInfoUserDefined = Nothing
```

```
Dim sFullCompanyInfo : sFullCompanyInfo = ""
Dim dicItem

For Each dicItem In dicCompanyInfo
    If (Len(dicCompanyInfo(dicItem)) > 0) Then
        sFullCompanyInfo = sFullCompanyInfo & dicItem & ": " &
dicCompanyInfo(dicItem) & vbCrlf
    End If
Next

WScript.ClipboardText = sFullCompanyInfo

Call cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die
Zwischenablage übernommen.", "CompanyInfo", vbOkOnly)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
// Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("AccountNo", companyInfo.AccountNo);
dictionaryCompanyInfo.Add("Bank", companyInfo.Bank);
dictionaryCompanyInfo.Add("BankCode", companyInfo.BankCode);
dictionaryCompanyInfo.Add("City", companyInfo.City);
dictionaryCompanyInfo.Add("Company", companyInfo.Company);
dictionaryCompanyInfo.Add("Company2", companyInfo.Company2);
dictionaryCompanyInfo.Add("Company3", companyInfo.Company3);
dictionaryCompanyInfo.Add("Country", companyInfo.Country);
dictionaryCompanyInfo.Add("Email", companyInfo.Email);
dictionaryCompanyInfo.Add("Extra1", companyInfo.Extra1);
dictionaryCompanyInfo.Add("Extra2", companyInfo.Extra2);
dictionaryCompanyInfo.Add("Fax", companyInfo.Fax);
dictionaryCompanyInfo.Add("IBAN", companyInfo.IBAN);
dictionaryCompanyInfo.Add("Internet", companyInfo.Internet);
dictionaryCompanyInfo.Add("Logo", companyInfo.Logo);
dictionaryCompanyInfo.Add("Phone", companyInfo.Phone);
dictionaryCompanyInfo.Add("Street", companyInfo.Street);
dictionaryCompanyInfo.Add("VatID", companyInfo.VatID);
dictionaryCompanyInfo.Add("ZIP", companyInfo.ZIP);

ListCompanyInfoUserDefined listCompanyInfoUserDefined =
companyInfo.UserDefinedFields;

foreach (CompanyInfoUserDefinedItem companyInfoUserDefinedItem in
listCompanyInfoUserDefined)
{
    if (companyInfoUserDefinedItem.Type == 2)
    {
        dictionaryCompanyInfo.Add(companyInfoUserDefinedItem.Key,
companyInfoUserDefinedItem.Value);
    }
}

string fullCompanyInfo = string.Empty;

foreach (var dictionaryItem in dictionaryCompanyInfo)
{
    if (dictionaryItem.Value != "")
    {
        fullCompanyInfo = fullCompanyInfo + dictionaryItem.Key + ": " +
dictionaryItem.Value + System.Environment.NewLine;
    }
}
```

```
WScript.ClipboardText = fullCompanyInfo;
cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die Zwischenablage
übernommen.", "CompanyInfo", 0);

companyInfo.Dispose();
```

3.29 ListContainers Objekt

Liste aller Container einer Ansicht.

3.29.1 Methoden

ItemByName

Beschreibung:

Liefert ein Objekt vom Typ **Container** mit dem angegebenen Namen zurück.

Beim Erzeugen des **ListContainers**-Objekts ist darauf zu achten, dass dabei **CurrentInputForm(2)** verwendet wird. Damit wird der aktuelle Modus des Containers samt aktuell selektiertem Datensatz beibehalten.

Parameter:

Parametername	Typ	Beschreibung
ContainerName	String	Name des gewünschten Containers. Der Name setzt sich dabei wie folgt zusammen: Vollständige Relation (Achtung! Hier darf kein Relationsalias verwendet werden. Korrekt wäre bspw. *ID.Aktivitäten.ContactID* für die Relation zwischen Aktivitäten und Kontakten) + # + ID des entsprechenden Containers (bspw. {*B3C0768A-5599-44B5-B4F2-7D31A6C10EC5*}). Der korrekte Parameter für den Aktivitäten-container der Ansicht Kontakte in der mitgelieferten Large Solution sähe dann wie folgt aus: *ID.Aktivitäten.KontaktID#{B3C0768A-5599-44B5-B4F2-7D31A6C10EC5}*

Rückgabewert:

Container

Beispiel VBScript:

```
' Dieses Beispiel basiert auf dem Aktivitäten-Container der Kontakte-Ansicht einer
combit_Large-Solution

Dim oContainer : Set oContainer =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.KontaktID#{B3C0768A-5599-44B5-B4F2-7D31A6C10EC5}")
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf dem Aktivitäten-Container der Kontakte-Ansicht
einer combit_Large-Solution

Container container =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).Containers.ItemByNam
e("ID.Aktivitäten.KontaktID#{B3C0768A-5599-44B5-B4F2-7D31A6C10EC5}");
```

3.30 ListRelations Objekt

Liste aller konfigurierten Relationen (Menüpunkt: **Ansichteneigenschaft der betreffenden Ansicht > Relationen**) einer Ansicht.

3.30.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der konfigurierten Relationen der Ansicht zurück.

> **Hinweis**: Aus Sicherheitsgründen ist der Zugriff auf **Relation** Objekte von Ansichten, auf die der aktuelle Benutzer keine Zugriffsrechte besitzt, nicht möglich. Alle diese **Relation** Objekte werden in der übergeordneten Collection (**ListRelations**) für die Eigenschaften/Methoden *Count* und *Item* nicht angeboten. Bei einem versuchten Direktzugriff per *ItemByName* wird kein Objekt zurückgegeben – es erfolgt zudem eine Fehlerausgabe auf das Debug-Tool Debwin.

Typ:

Long

Beispiel VBScript:

```
' Durchläuft alle Relationen der Kontakte-Ansicht und prüft, ob für diese
Relationen das Papierkorb-Feature aktiviert wurde

Dim oListRelations : Set oListRelations =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").Relations
Dim nRelationCount : nRelationCount = oListRelations.Count
Dim nCounter : nCounter = 0
Dim oRelation
Dim sListCascadeOnDeleteViews : sListCascadeOnDeleteViews = ""

For nCounter = 1 To nRelationsCount

    Set oRelation = oListRelations.Item(nCounter)

    If (oRelation.CascadeOnDelete = True) Then
        If (nCounter = nRelationsCount) Then
            sListCascadeOnDeleteViews = sListCascadeOnDeleteViews & oRelation.Name
        Else
            sListCascadeOnDeleteViews = sListCascadeOnDeleteViews & oRelation.Name
& vbCrLf
        End If

    End If

    Set oRelation = Nothing

Next

Set oListRelations = Nothing

If (Len(sListCascadeOnDeleteViews) > 0) Then
    Call cRM.DialogMessageBox("Für folgende Relation(en) wurde der kaskadierende
Papierkorb aktiviert: " & vbCrLf &  sListCascadeOnDeleteViews, "ListRelations",
vbOkOnly)
Else
    Call cRM.DialogMessageBox("Derzeit wurde bei keiner Relation der kaskadierende
Papierkorb aktiviert.", "ListRelations", vbOkOnly)
End If
```

Beispiel C#-Script:

```
// Durchläuft alle Relationen der Kontakte-Ansicht und prüft, ob für diese
Relationen das Papierkorb-Feature aktiviert wurde

ListRelations relations =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").Relations;
System.Collections.Generic.List<string> relationsWithCascadeOnDelete = new
System.Collections.Generic.List<string>();

foreach (Relation relation in relations)
{
    if (relation.CascadeOnDelete == true)
    {
        relationsWithCascadeOnDelete.Add(relation.Alias);
    }
}

if (relationsWithCascadeOnDelete.Count > 0)
{
    cRM.DialogMessageBox("Für folgende Relation(en) wurde der kaskadierende
Papierkorb aktiviert: " + System.Environment.NewLine + string.Join(", ",
relationsWithCascadeOnDelete.ToArray()), "ListRelations", 0);
}
```

3.30.2 Methoden

Item

Beschreibung:

Gibt eine **Relation** entsprechend dem übergebenen Index zurück. Der Index geht von 1 bis Count.

Hinweis: Aus Sicherheitsgründen ist der Zugriff auf **Relation** Objekte von Ansichten, auf die der aktuelle Benutzer keine Zugriffsrechte besitzt, nicht möglich. Alle diese **Relation** Objekte werden in der übergeordneten Collection (**ListRelations**) für die Eigenschaften/Methoden *Count* und *Item* nicht angeboten. Bei einem versuchten Direktzugriff per *ItemByName* wird kein Objekt zurückgegeben – es erfolgt zudem eine Fehlerausgabe auf das Debug-Tool Debwin.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Relation

Beispiel VBScript:

```
' Durchläuft alle Relationen der Kontakte-Ansicht und prüft, ob für diese
Relationen das Papierkorb-Feature aktiviert wurde

Dim oListRelations : Set oListRelations =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").Relations
Dim nRelationCount : nRelationCount = oListRelations.Count
Dim nCounter : nCounter = 0
Dim oRelation
Dim sListCascadeOnDeleteViews : sListCascadeOnDeleteViews = ""

For nCounter = 1 To nRelationsCount

    Set oRelation = oListRelations.Item(nCounter)
```

```
        If (oRelation.CascadeOnDelete = True) Then
            If (nCounter = nRelationsCount) Then
                sListCascadeOnDeleteViews = sListCascadeOnDeleteViews & oRelation.Name
            Else
                sListCascadeOnDeleteViews = sListCascadeOnDeleteViews & oRelation.Name
& vbCrLf
            End If

        End If

        Set oRelation = Nothing

    Next

    Set oListRelations = Nothing

    If (Len(sListCascadeOnDeleteViews) > 0) Then
        Call cRM.DialogMessageBox("Für folgende Relation(en) wurde der kaskadierende
Papierkorb aktiviert: " & vbCrLf &  sListCascadeOnDeleteViews, "ListRelations",
vbOkOnly)
    Else
        Call cRM.DialogMessageBox("Derzeit wurde bei keiner Relation der kaskadierende
Papierkorb aktiviert.", "ListRelations", vbOkOnly)
    End If
```

Beispiel C#-Script:

```
// Durchläuft alle Relationen der Kontakte-Ansicht und prüft, ob für diese
Relationen das Papierkorb-Feature aktiviert wurde

ListRelations relations =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").Relations;
System.Collections.Generic.List<string> relationsWithCascadeOnDelete = new
System.Collections.Generic.List<string>();

foreach (Relation relation in relations)
{
    if (relation.CascadeOnDelete == true)
    {
        relationsWithCascadeOnDelete.Add(relation.Alias);
    }
}

if (relationsWithCascadeOnDelete.Count > 0)
{
    cRM.DialogMessageBox("Für folgende Relation(en) wurde der kaskadierende
Papierkorb aktiviert: " + System.Environment.NewLine + string.Join(", ",
relationsWithCascadeOnDelete.ToArray()), "ListRelations", 0);
}
```

ItemByName

Beschreibung:

Gibt eine **Relation** entsprechend dem übergebenen Namen zurück.

Hinweis: Aus Sicherheitsgründen ist der Zugriff auf **Relation** Objekte von Ansichten, auf die der aktuelle Benutzer keine Zugriffsrechte besitzt, nicht möglich. Alle diese **Relation** Objekte werden in der übergeordneten Collection (**ListRelations**) für die Eigenschaften/Methoden *Count* und *Item* nicht angeboten. Bei einem versuchten Direktzugriff per *ItemByName* wird kein Objekt zurückgegeben – es erfolgt zudem eine Fehlerausgabe auf das Debug-Tool Debwin.

Parameter:

Parametername	Typ	Beschreibung

Name	String	Name der Relation im Format *ID.Aktivitäten.ContactID* (Primärschlüssel-feld.Relationsalias.Fremschlüsselfeld).

Rückgabewert:

Relation

Beispiel VBScript:

```
' Erstellt einen neuen Aktivitäten-Datensatz und fügt ein Dokument hinzu. Basis
ist hierbei der Aktivitäten-Container der Firmen-Ansicht einer combit_Large-
Solution

Dim oDocMngr : Set oDocMngr = cRM.CurrentProject.DocMngr
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Dim oRelation : Set oRelation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID")
Dim sFileToAppend : sFileToAppend = "C:\Firmen-Dossier.docx"
Dim sDocDescription : sDocDescription = "docx"

If (oDocMngr.AppendFile(oRecord, oRelation, sFileToAppend, sDocDescription) =
True) Then
    Call cRM.DialogMessageBox("Die Datei """ & sFileToAppend & """ konnte
erfolgreich hinzugefügt werden.", "DocMngr.AppendFile", vbOkOnly)
Else
    If (oDocMngr.LastError = 32) Then
        Call cRM.DialogMessageBox("Das Hinzufügen der Datei """ & sFileToAppend &
""" konnte nicht erfolgreich durchgeführt werden. Der Schreibzugriff wurde
verweigert.", "DocMngr.AppendFile", vbOkOnly)
    ElseIf (oDocMngr.LastError = 33) Then
        Call cRM.DialogMessageBox("Das Hinzufügen der Datei """ & sFileToAppend &
""" konnte nicht erfolgreich durchgeführt werden. Das Kopieren der Datei ist
fehlgeschlagen.", "DocMngr.AppendFile", vbOkOnly)
    End If
End If

Set oRelation = Nothing
Set oRecord = Nothing
Set oDocMngr = Nothing
```

Beispiel C#-Script:

```
// Erstellt einen neuen Aktivitäten-Datensatz und fügt ein Dokument hinzu. Basis
ist hierbei der Aktivitäten-Container der Firmen-Ansicht einer combit_Large-
Solution

DocMngr docMngr = cRM.CurrentProject.DocMngr;
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
Relation relation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID");
string fileToAppend = @"C:\Firmen-Dossier.docx";
string docDescription = "docx";

if (docMngr.AppendFile(record, relation, fileToAppend, docDescription) == true)
{
    cRM.DialogMessageBox("Die Datei \"" + fileToAppend + "\" konnte erfolgreich
hinzugefügt werden.", "DocMngr.AppendFile", 0);
}
else
{
    if (docMngr.LastError.ErrorCode == 32)
    {
        cRM.DialogMessageBox("Das Hinzufügen der Datei \"" + fileToAppend + "\"
konnte nicht erfolgreich durchgeführt werden. Der Schreibzugriff wurde
verweigert.", "DocMngr.AppendFile", 0);
```

```
    }
    else if (docMngr.LastError.ErrorCode == 33)
    {
        cRM.DialogMessageBox("Das Hinzufügen der Datei \"" + fileToAppend + "\"
konnte nicht erfolgreich durchgeführt werden. Das Kopieren der Datei ist
fehlgeschlagen.", "DocMngr.AppendFile", 0);
    }
}

relation.Dispose();
record.Dispose();
docMngr.Dispose();
```

3.31 ListViewConfigs Objekt

3.31.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der konfigurierten Ansichten.

> Hinweis: Aus Sicherheitsgründen ist der Zugriff auf ViewConfig Objekte von Ansichten, auf die der aktuelle Benutzer keine Zugriffsrechte besitzt, nicht möglich. Alle diese ViewConfig Objekte werden in der übergeordneten Collection (ListViewConfigs) für die Eigenschaften/Methoden *Count* und *Item* nicht angeboten. Bei einem versuchten Direktzugriff per *ItemByName* wird kein Objekt zurückgegeben – es erfolgt zudem eine Fehlerausgabe auf das Debug-Tool Debwin.

Typ:

Long

Beispiel VBScript:

```
' Durchläuft alle Ansichten und prüft welche Ansichten mehr als 30 Felder besitzen

Dim oListViewConfigs : Set oListViewConfigs = cRM.CurrentProject.ViewConfigs
Dim nCounter : nCounter = 0
Dim oViewConfig
Dim sViewNames : sViewNames = ""

For nCounter = 1 To oListViewConfigs.Count

    Set oViewConfig = oListViewConfigs.Item(nCounter)

    If (oViewConfig.FldCount > 30) Then

        If (nCounter = oListViewConfigs.Count) Then
            sViewNames = sViewNames & oViewConfig.Name
        Else
            sViewNames = sViewNames & oViewConfig.Name & vbCrLf
        End If

    End If

    Set oViewConfig = Nothing

Next

If (Len(sViewNames) > 0) Then
    Call cRM.DialogMessageBox("Folgende Ansichten verwenden mehr als 30 Felder: "
& vbCrLf &  sViewNames, "ListViewConfigs", vbOkOnly)
Else
```

```
    Call cRM.DialogMessageBox("Derzeit verwendet keine Ansicht mehr als 30
Felder.", "ListViewConfigs", vbOkOnly)
End If

Set oListViewConfigs = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Ansichten und prüft welche Ansichten mehr als 30 Felder
besitzen

ListViewConfigs viewConfigs = cRM.CurrentProject.ViewConfigs;
System.Collections.Generic.List<string> viewsWithMoreThanThirtyFields = new
System.Collections.Generic.List<string>();

foreach (ViewConfig viewConfig in viewConfigs)
{
    if (viewConfig.FldCount > 30)
    {
        viewsWithMoreThanThirtyFields.Add(viewConfig.Name);
    }
}

if (viewsWithMoreThanThirtyFields.Count > 0)
{
    cRM.DialogMessageBox("Folgende Ansichten verwenden mehr als 30 Felder: " +
System.Environment.NewLine + string.Join(", ",
viewsWithMoreThanThirtyFields.ToArray()), "ListViewConfigs", 0);
}

viewConfigs.Dispose();
```

3.31.2 Methoden

Item

Beschreibung:

Liefert eine konfigurierte Ansicht als Objekt zurück. Es muss die Index-Nummer der Ansicht übergeben werden. Der Index geht von 1 bis Count.

Hinweis: Aus Sicherheitsgründen ist der Zugriff auf ViewConfig Objekte von Ansichten, auf die der aktuelle Benutzer keine Zugriffsrechte besitzt, nicht möglich. Alle diese ViewConfig Objekte werden in der übergeordneten Collection (ListViewConfigs) für die Eigenschaften/Methoden *Count* und *Item* nicht angeboten. Bei einem versuchten Direktzugriff per *ItemByName* wird kein Objekt zurückgegeben – es erfolgt zudem eine Fehlerausgabe auf das Debug-Tool Debwin.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

ViewConfig

Beispiel VBScript:

```
' Durchläuft alle Ansichten und prüft welche Ansichten mehr als 30 Felder besitzen

Dim oListViewConfigs : Set oListViewConfigs = cRM.CurrentProject.ViewConfigs
Dim nCounter : nCounter = 0
Dim oViewConfig
Dim sViewNames : sViewNames = ""
```

```
For nCounter = 1 To oListViewConfigs.Count

    Set oViewConfig = oListViewConfigs.Item(nCounter)

    If (oViewConfig.FldCount > 30) Then

        If (nCounter = oListViewConfigs.Count) Then
            sViewNames = sViewNames & oViewConfig.Name
        Else
            sViewNames = sViewNames & oViewConfig.Name & vbCrLf
        End If

    End If

    Set oViewConfig = Nothing

Next

If (Len(sViewNames) > 0) Then
    Call cRM.DialogMessageBox("Folgende Ansichten verwenden mehr als 30 Felder: "
& vbCrLf &  sViewNames, "ListViewConfigs", vbOkOnly)
Else
    Call cRM.DialogMessageBox("Derzeit verwendet keine Ansicht mehr als 30
Felder.", "ListViewConfigs", vbOkOnly)
End If

Set oListViewConfigs = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Ansichten und prüft welche Ansichten mehr als 30 Felder
besitzen

ListViewConfigs viewConfigs = cRM.CurrentProject.ViewConfigs;
System.Collections.Generic.List<string> viewsWithMoreThanThirtyFields = new
System.Collections.Generic.List<string>();

foreach (ViewConfig viewConfig in viewConfigs)
{
    if (viewConfig.FldCount > 30)
    {
        viewsWithMoreThanThirtyFields.Add(viewConfig.Name);
    }
}

if (viewsWithMoreThanThirtyFields.Count > 0)
{
    cRM.DialogMessageBox("Folgende Ansichten verwenden mehr als 30 Felder: " +
System.Environment.NewLine + string.Join(", ",
viewsWithMoreThanThirtyFields.ToArray()), "ListViewConfigs", 0);
}

viewConfigs.Dispose();
```

ItemByName

Beschreibung:

Liefert eine Ansicht mit dem übergebenen Namen als Objekt zurück.

Hinweis: Aus Sicherheitsgründen ist der Zugriff auf ViewConfig Objekte von Ansichten, auf die der aktuelle Benutzer keine Zugriffsrechte besitzt, nicht möglich. Alle diese ViewConfig Objekte werden in der übergeordneten Collection (ListViewConfigs) für die Eigenschaften/Methoden *Count* und *Item* nicht angeboten. Bei einem versuchten Direktzugriff per *ItemByName* wird kein Objekt zurückgegeben – es erfolgt zudem eine Fehlerausgabe auf das Debug-Tool Debwin.

Parameter:

Parametername	Typ	Beschreibung
ViewName	String	Name der Ansicht.

Rückgabewert:

ViewConfig

Beispiel VBScript:

```
Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
...
Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
// ...
viewConfig.Dispose();
```

3.32 ListViews Objekt

3.32.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der aktiven Ansichten.

Typ:

Long

Beispiel VBScript:

```
Dim nViewCount : nViewCount = cRM.CurrentProject.ActiveViews.Count
```

Beispiel C#-Script:

```
long viewCount = cRM.CurrentProject.ActiveViews.Count;
```

3.32.2 Methoden

ActiveView

Beschreibung:

Liefert ein Objekt vom Typ **View**, der aktiven (offen und im Vordergrund befindlichen) Ansicht in der Anwendung.

Rückgabewert:

View

> Wichtig: Bitte beachten Sie die Hinweise im Kapitel Hinweise zur Benutzung von Scripten in Folgeverknüpfungen.

Beispiel VBScript:

```
Dim oActiveView : Set oActiveView = cRM.CurrentProject.ActiveViews.ActiveView
...
Set oActiveView = Nothing
```

Beispiel C#-Script:

```
View activeView = cRM.CurrentProject.ActiveViews.ActiveView;
// ...
activeView.Dispose();
```

CloseActiveView

Beschreibung:

Schließt eine aktive Ansicht.

Rückgabewert:

Bool

> **Wichtig**: Ruft man CloseActiveView auf, während die aktive Ansicht die Ansicht ist, in der das Script läuft, kann dies zu Problemen führen. Dies sollte daher vermeiden werden.

Beispiel VBScript:

```
Dim oActiveView : Set oActiveView = cRM.CurrentProject.ActiveViews.ActiveView

If (Not oActiveView Is Nothing) Then
    Call oActiveView.CloseActiveView()
End If
```

Beispiel C#-Script:

```
View activeView = cRM.CurrentProject.ActiveViews.ActiveView;

if (activeView != null)
{
    cRM.CurrentProject.ActiveViews.CloseActiveView();
}

activeView.Dispose();
```

CloseView

Beschreibung:

Schließt eine geöffnete (konfigurierte) Ansicht. Es muss die Index-Nummer der Ansicht übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Bool

> **Wichtig**: Ruft man CloseView mit dem Index der aktuellen Ansicht auf (in der das Script läuft), kann dies zu Problemen führen. Dies sollte daher vermeiden werden.

Beispiel VBScript:

```
' Schließt alle Ansichten außer der derzeit geöffneten Ansicht

Dim oActiveViews : Set oActiveViews = cRM.CurrentProject.ActiveViews
Dim sCurrentActiveViewName : sCurrentActiveViewName = oActiveViews.ActiveView.Name
Dim nCounter : nCounter = 0
Dim oView
```

```
For nCounter = 1 To oActiveViews.Count

    Set oView = oActiveViews.Item(nCounter)

    If (oView.Name <> sCurrentActiveViewName) Then

        Call oActiveViews.CloseView(nCounter)

    End If

    Set oView = Nothing

Next

Set oActiveViews = Nothing
```

Beispiel C#-Script:

```
// Schließt alle Ansichten außer der derzeit geöffneten Ansicht

ListViews activeViews = cRM.CurrentProject.ActiveViews;
string currentViewName = activeViews.ActiveView.Name;
long index = 0;

foreach (View view in activeViews)
{
    index++;

    if (view.Name != currentViewName)
    {
        activeViews.CloseView(index);
    }
}

activeViews.Dispose();
```

Item

Beschreibung:

Liefert eine aktive Ansicht als Objekt zurück. Es muss die Index-Nummer der Ansicht übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

View

Beispiel VBScript:

```
' Schließt alle Ansichten außer der derzeit geöffneten Ansicht

Dim oActiveViews : Set oActiveViews = cRM.CurrentProject.ActiveViews
Dim sCurrentActiveViewName : sCurrentActiveViewName = oActiveViews.ActiveView.Name
Dim nCounter : nCounter = 0
Dim oView

For nCounter = 1 To oActiveViews.Count

    Set oView = oActiveViews.Item(nCounter)

    If (oView.Name <> sCurrentActiveViewName) Then

        Call oActiveViews.CloseView(nCounter)

    End If
```

```
        Set oView = Nothing

    Next

    Set oActiveViews = Nothing
```

Beispiel C#-Script:

```
// Schließt alle Ansichten außer der derzeit geöffneten Ansicht

ListViews activeViews = cRM.CurrentProject.ActiveViews;
string currentViewName = activeViews.ActiveView.Name;
View view;

for (long counter = 0; counter < activeViews.Count; counter++)
{
    view = activeViews.Item(counter);

    if (view.Name != currentViewName)
    {
        activeViews.CloseView(counter);
    }
}

view.Dispose();
activeViews.Dispose();
```

ItemByName

Beschreibung:

Liefert eine Ansicht mit dem übergebenen Namen als Objekt zurück.

Parameter:

Parametername	Typ	Beschreibung
ViewName	String	Name der Ansicht

Rückgabewert:

View

Beispiel VBScript:

```
Dim oView : Set oView = cRM.CurrentProject.ActiveViews.ItemByName("Firmen")
...
Set oView = Nothing
```

Beispiel C#-Script:

```
View view = cRM.CurrentProject.ActiveViews.ItemByName("Firmen");
//...
view.Dispose();
```

ItemByTagID

Beschreibung:

Liefert eine aktive Ansicht über die TagID (Identifiername) als Objekt zurück.

Parameter:

Parametername	Typ	Beschreibung
TagID	String	Identifiername

Rückgabewert:

View

Beispiel VBScript:

```
Dim oView : Set oView = cRM.CurrentProject.ActiveViews.ItemByTagID(sTagID)
...
Set oView = Nothing
```

Beispiel C#-Script:

```
View view = cRM.CurrentProject.ActiveViews.ItemByTagID(sTagID);
//...
view.Dispose();
```

3.33 ListWebElements Objekt

Liste aller Web-Elemente einer Ansicht.

3.33.1 Methoden

ItemByName

Beschreibung:

Liefert ein Objekt vom Typ **WebElement** mit der angegebenen ID zurück.

Parameter:

Parametername	Typ	Beschreibung
WebElementID	String	ID des gewünschten Web-Elements. Die ID kann dem Eigenschaftsdialog des jeweiligen Web-Elements entnommen werden. Der korrekte Parameter für das Web-Element "Google Maps" der Ansicht WebElemente in der mitgelieferten Large Solution sähe dann wie folgt aus: *{64BBFFD7-EA32-4358-BBFC-744D6A94291D}*

Rückgabewert:

WebElement

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der WebElemente-Ansicht einer combit_Large-Solution

<!--#pragma keepeditmode-->
Dim oListWebElements : Set oListWebElements =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).WebElements
Dim oWebElement : Set oWebElement = oListWebElements.ItemByName("{64BBFFD7-EA32-
4358-BBFC-744D6A94291D}")
Dim oInternetExplorer : Set oInternetExplorer = oWebElement.IE()
Call oInternetExplorer.Navigate("https://www.combit.net")
Set oInternetExplorer = Nothing
Set oWebElement = Nothing
Set oListWebElements = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der WebElemente-Ansicht einer combit_Large-Solution

//<!--#pragma keepeditmode-->
ListWebElements listWebElements =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).WebElements;
WebElement webElement = listWebElements.ItemByName("{64BBFFD7-EA32-4358-BBFC-
744D6A94291D}");
webElement.IE().Navigate("https://www.combit.net");
webElement.Dispose();
listWebElements.Dispose();
```

3.34 OLEError Objekt

Gibt Fehlermeldungen als Fehlercode oder Text zurück.

3.34.1 Eigenschaften

ErrorCode, read-only

Beschreibung:

Gibt den Fehlercode zurück. Welche Fehlercodes das sein können, hängt von den aufrufenden Objekten ab (siehe dort).

Typ:

Long

Beispiel VBScript:

```
Dim oOLEError : Set oOLEError = <Object>.LastError ' DocMngr, EmailTool*, SQLShell
Dim nErrorCode : nErrorCode = oOLEError.ErrorCode
Dim sErrorAppendix : sErrorAppendix = oOLEError.ErrorAppendix
Dim sErrorText : sErrorText = oOLEError.ErrorText
Dim nNativeCodeError : nNativeCodeError = oOLEError.NativeErrorCode

Call cRM.DialogMessageBox("Bitte beachten Sie folgende Informationen: " & vbCrLf &
"Fehlercode: " & CStr(nErrorCode) & " (Nativer Fehlercode: " &
CStr(nNativeCodeError) & ")" & vbCrLf & "Fehlermeldung: " & sErrorText & vbCrLf &
"Zusätzliche Fehlerinformationen: " & sErrorAppendix, "OLEError", vbOkOnly)

Set oOLEError = Nothing
```

Beispiel C#-Script:

```
OLEError error = <Object>.LastError; // DocMngr, EmailTool*, SQLShell
long errorCode = error.ErrorCode;
string errorAppendix = error.ErrorAppendix;
string errorText = error.ErrorText;
long nativeErrorCode = error.NativeErrorCode;

cRM.DialogMessageBox("Bitte beachten Sie folgende Informationen: " +
System.Environment.NewLine + "Fehlercode: " + errorCode.ToString() + " (Nativer
Fehlercode: " + nativeErrorCode.ToString() + ")" + System.Environment.NewLine +
"Fehlermeldung: " + errorText + System.Environment.NewLine + "Zusätzliche
Fehlerinformationen: " + errorAppendix, "OLEError", 0);

error.Dispose();
```

ErrorAppendix, read-only

Beschreibung:

Gibt möglicherweise zusätzliche Informationen zur Fehlermeldung als Text zurück.

Typ:

String

Beispiel VBScript:

```
Dim oOLEError : Set oOLEError = <Object>.LastError ' DocMngr, EmailTool*, SQLShell
Dim nErrorCode : nErrorCode = oOLEError.ErrorCode
Dim sErrorAppendix : sErrorAppendix = oOLEError.ErrorAppendix
Dim sErrorText : sErrorText = oOLEError.ErrorText
Dim nNativeCodeError : nNativeCodeError = oOLEError.NativeErrorCode

Call cRM.DialogMessageBox("Bitte beachten Sie folgende Informationen: " & vbCrLf &
"Fehlercode: " & CStr(nErrorCode) & " (Nativer Fehlercode: " &
```

```
CStr(nNativeCodeError) & ")" & vbCrLf & "Fehlermeldung: " & sErrorText & vbCrLf &
"Zusätzliche Fehlerinformationen: " & sErrorAppendix, "OLEError", vbOkOnly)

Set oOLEError = Nothing
```

Beispiel C#-Script:

```
OLEError error = <Object>.LastError; // DocMngr, EmailTool*, SQLShell
long errorCode = error.ErrorCode;
string errorAppendix = error.ErrorAppendix;
string errorText = error.ErrorText;
long nativeErrorCode = error.NativeErrorCode;

cRM.DialogMessageBox("Bitte beachten Sie folgende Informationen: " +
System.Environment.NewLine + "Fehlercode: " + errorCode.ToString() + " (Nativer
Fehlercode: " + nativeErrorCode.ToString() + ")" + System.Environment.NewLine +
"Fehlermeldung: " + errorText + System.Environment.NewLine + "Zusätzliche
Fehlerinformationen: " + errorAppendix, "OLEError", 0);

error.Dispose();
```

ErrorText, read-only

Beschreibung:

Gibt die Fehlermeldung als Text zurück.

Typ:

String

Beispiel VBScript:

```
Dim oOLEError : Set oOLEError = <Object>.LastError ' DocMngr, EmailTool*, SQLShell
Dim nErrorCode : nErrorCode = oOLEError.ErrorCode
Dim sErrorAppendix : sErrorAppendix = oOLEError.ErrorAppendix
Dim sErrorText : sErrorText = oOLEError.ErrorText
Dim nNativeCodeError : nNativeCodeError = oOLEError.NativeErrorCode

Call cRM.DialogMessageBox("Bitte beachten Sie folgende Informationen: " & vbCrLf &
"Fehlercode: " & CStr(nErrorCode) & " (Nativer Fehlercode: " &
CStr(nNativeCodeError) & ")" & vbCrLf & "Fehlermeldung: " & sErrorText & vbCrLf &
"Zusätzliche Fehlerinformationen: " & sErrorAppendix, "OLEError", vbOkOnly)

Set oOLEError = Nothing
```

Beispiel C#-Script:

```
OLEError error = <Object>.LastError; // DocMngr, EmailTool*, SQLShell
long errorCode = error.ErrorCode;
string errorAppendix = error.ErrorAppendix;
string errorText = error.ErrorText;
long nativeErrorCode = error.NativeErrorCode;

cRM.DialogMessageBox("Bitte beachten Sie folgende Informationen: " +
System.Environment.NewLine + "Fehlercode: " + errorCode.ToString() + " (Nativer
Fehlercode: " + nativeErrorCode.ToString() + ")" + System.Environment.NewLine +
"Fehlermeldung: " + errorText + System.Environment.NewLine + "Zusätzliche
Fehlerinformationen: " + errorAppendix, "OLEError", 0);

error.Dispose();
```

NativeErrorCode, read-only

Beschreibung:

Gibt die Fehlermeldung des Datenbankservers, -clients oder des Betriebssystems zurück.

Typ:

Long

Beispiel VBScript:

```
Dim oOLEError : Set oOLEError = <Object>.LastError ' DocMngr, EmailTool*, SQLShell
Dim nErrorCode : nErrorCode = oOLEError.ErrorCode
Dim sErrorAppendix : sErrorAppendix = oOLEError.ErrorAppendix
Dim sErrorText : sErrorText = oOLEError.ErrorText
Dim nNativeCodeError : nNativeCodeError = oOLEError.NativeErrorCode

Call cRM.DialogMessageBox("Bitte beachten Sie folgende Informationen: " & vbCrLf &
"Fehlercode: " & CStr(nErrorCode) & " (Nativer Fehlercode: " &
CStr(nNativeCodeError) & ")" & vbCrLf & "Fehlermeldung: " & sErrorText & vbCrLf &
"Zusätzliche Fehlerinformationen: " & sErrorAppendix, "OLEError", vbOkOnly)

Set oOLEError = Nothing
```

Beispiel C#-Script:

```
OLEError error = <Object>.LastError; // DocMngr, EmailTool*, SQLShell
long errorCode = error.ErrorCode;
string errorAppendix = error.ErrorAppendix;
string errorText = error.ErrorText;
long nativeErrorCode = error.NativeErrorCode;

cRM.DialogMessageBox("Bitte beachten Sie folgende Informationen: " +
System.Environment.NewLine + "Fehlercode: " + errorCode.ToString() + " (Nativer
Fehlercode: " + nativeErrorCode.ToString() + ")" + System.Environment.NewLine +
"Fehlermeldung: " + errorText + System.Environment.NewLine + "Zusätzliche
Fehlerinformationen: " + errorAppendix, "OLEError", 0);

error.Dispose();
```

3.35 phonemanager Objekt

3.35.1 Eigenschaften

CallList, read-only

Beschreibung:

Liefert ein Objekt vom Typ **CallList** (s.u.) mit der Anrufliste des phone manager zurück.

Typ:

CallList

Beispiel VBScript:

```
' Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen bzw. gibt dem
Nutzer eine Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

Dim oCallList : Set oCallList = cRM.phonemanager.CallList
Dim oCallItem
Dim nCount : nCount = 0
Dim nDialRetriesCount : nDialRetriesCount = 0
Dim sFirstInfo : sFirstInfo = ""
Dim sInfo : sInfo = ""
Dim sLastInfo : sLastInfo = ""
Dim sNumber : sNumber = ""

For nCount = 1 To oCallList.Count

    Set oCallItem = oCallList.Item(nCount)

    If (oCallItem.DialRetriesCount > 10) Then
        Call oCallItem.Remove()
    Else
        nDialRetriesCount = oCallItem.DialRetriesCount
```

```
        sFirstInfo = oCallItem.FirstInfo
        sInfo = oCallItem.Info
        sLastInfo = oCallItem.LastInfo
        sNumber = oCallItem.Number

        If (cRM.DialogMessageBox("Die Nummer " & sNumber & " (" & sFirstInfo & " -
" & sInfo & " - " & sLastInfo & ") konnte mit " & CStr(nDialRetriesCount) & "
Anrufversuchen nicht erreicht werden." & vbCrlf & "Soll diese Rufnummer aus der
Anrufliste augetragen werden?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
            Call oCallItem.Remove()
        Else
            If (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", vbYesNoCancel) = vbYes) Then
                Call oCallItem.GotoRecord()
            End If
        End If
    End If

    Set oCallItem = Nothing

Next

Set oCallList = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Einträge der Anrufliste und entfernt dabei alle die Einträge,
die mehr als 10 Wählversuche hinterlegt haben bzw. gibt dem Nutzer eine
Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen bzw. gibt dem
Nutzer eine Möglichkeit auf den Datensatz des Anruflisteneintrags zu springen

CallList callList = cRM.PhoneManager.CallList;

foreach (CallItem item in callList)
{
    if (item.DialRetriesCount > 10)
    {
        item.Remove();
    }
    else
    {
        if (cRM.DialogMessageBox("Die Nummer " + item.Number + "(" +
item.FirstInfo + " - " + item.Info + " - " + item.LastInfo + ") konnte mit " +
item.DialRetriesCount.ToString() + "Anrufversuchen nicht erreicht werden." +
System.Environment.NewLine + "Soll diese Rufnummer aus der Anrufliste ausgetragen
werden?", "CallList.CallItem", 3) == 6)
        {
            item.Remove();
        }
        else
        {
            if (cRM.DialogMessageBox("Der Eintrag wurde nicht aus der Anrufliste
entfernt. Soll jetzt versucht werden auf den zugehörigen Datensatz im combit CRM
zu springen?", "CallList.CallItem", 3) == 6)
            {
                item.GotoRecord();
            }
        }
    }
}

callList.Dispose();
```

3.35.2 Methoden

Activate

Beschreibung:

Aktiviert den phone manager.

Beispiel VBScript:

```
Dim oPhoneManager : Set oPhoneManager = cRM.phonemanager
Call oPhoneManager.Activate()
Set oPhoneManager = Nothing
```

Beispiel C#-Script:

```
PhoneManager pm = cRM.PhoneManager;
pm.Activate();
pm.Dispose();
```

Dial

Beschreibung:

Wählt die übergebene Telefonnummer.

Hinweis: Hierbei werden im Gegensatz zur Wahl über den Kontextmenüpunkt "Anrufen" eines Telefonfeldes keine weiteren Aktionen wie Autoprotokolle etc. ausgeführt. Siehe auch die Methode DialNumber im Record Objekt.

Parameter:

Parametername	Typ	Beschreibung
Number	String	Die zu wählende Rufnummer.

Beispiel VBScript:

```
Dim oPhoneManager : Set oPhoneManager = cRM.phonemanager
Dim sPhoneNumber : sPhoneNumber =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetConten
tsByName("Phone")
Call oPhoneManager.Dial(sPhoneNumber)
Set oPhoneManager = Nothing
```

Beispiel C#-Script:

```
PhoneManager pm = cRM.PhoneManager;
string phoneNumber =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetConten
tsByName("Phone");
pm.Dial(phoneNumber);
pm.Dispose();
```

InvokeMenu

Beschreibung:

Ruft einen Menüeintrag des phone manager auf. Neben der ID des Menüeintrages wird angegeben, ob das Script solange warten soll, bis der Befehl abgearbeitet wurde (und evtl. Dialoge geschlossen wurden) oder ob das Script direkt weiterlaufen soll. Die Menü-IDs des phone manager finden Sie im Kapitel Menü-IDs.

Hinweis: Es werden nur Menü-IDs von direkt sichtbaren Menüs unterstützt, d.h. Kontextmenüs können nicht verwendet werden. Sollte die Methode in einem asynchron ausgeführten Script ausgeführt werden, so ist der

Rückgabewert immer True. Der Rückgabewert beschreibt, ob der Aufruf übermittelt werden konnte, nicht jedoch, ob in der aufzurufenden Funktion ggf. ein Problem festgestellt wurde.

Parameter:

Parametername	Typ	Beschreibung
MenuID	Long	Die ID des Menüeintrages.
Synchron	Bool	True: synchrone Ausführung False: asynchrone Ausführung

Rückgabewert:

Bool

Wert	Beschreibung
True	Befehl zum Aufrufen eines Menüeintrags wurde erfolgreich an den phone manager übermittelt.
False	Befehl zum Aufrufen eines Menüeintrags konnte nicht übermittelt werden. Dies kann z. B. der Fall sein, wenn der aufzurufende Menü-Befehl derzeit nicht zur Verfügung steht.

Beispiel VBScript:

```
Dim oPhoneManager : Set oPhoneManager = cRM.phonemanager
Dim nMenuIDPrintCallList : nMenuIDPrintCallList = 106
Call oPhoneManager.InvokeMenu(nMenuIDPrintCallList, True)
Set oPhoneManager = Nothing
```

Beispiel C#-Script:

```
PhoneManager pm = cRM.PhoneManager;
long menuIDPrintCallList = 106;
pm.InvokeMenu(menuIDPrintCallList, true);
pm.Dispose();
```

3.36 Project Objekt

3.36.1 Eigenschaften

ActiveViews, read-only

Beschreibung:

Gibt die aktiven geöffneten Ansichten als Objekt vom Typ **ListViews** zurück.

Wichtig: **ListViews** wird beim Erzeugen initialisiert. Um den aktuellen Stand zu erhalten muss das Objekt neu erzeugt werden.

Typ:

ListViews

Beispiel VBScript:

```
Dim oActiveViews : Set oActiveViews = cRM.CurrentProject.ActiveViews
' ...
Set oActiveViews = Nothing
```

Beispiel C#-Script:

```
ListViews activeViews = cRM.CurrentProject.ActiveViews;
// ...
activeViews.Dispose();
```

CompanyInfo, read-only

Beschreibung:

Liefert die Firmenstammdaten zurück.

Typ:

CompanyInfo

Beispiel VBScript:

```
Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
' ...
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
// ...
companyInfo.Dispose();
```

CurrentUser, read-only

Beschreibung:

Liefert den aktiven Benutzer der Anwendung.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Der aktuell angemeldete Benutzer im combit CRM ist: " &
cRM.CurrentProject.CurrentUser, "Project.CurrentUser", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Der aktuell angemeldete Benutzer im combit CRM ist: " +
cRM.CurrentProject.CurrentUser, "Project.CurrentUser", 0);
```

DatabaseName, read-only

Beschreibung:

Gibt den Datenbanknamen des Projektes zurück.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Der physikalische Name der Datenbank des geladenen
combit CRM Projekts lautet: " & cRM.CurrentProject.DatabaseName,
"Project.DatabaseName", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Der physikalische Name der Datenbank des geladenen combit
CRM Projekts lautet: " + cRM.CurrentProject.DatabaseName, "Project.DatabaseName",
0);
```

DBSystemType, read-only

Beschreibung:

Liefert das eingestellte Datenbanksystem zurück.

Typ:

Long

Wert	Beschreibung
0	keine Verbindung aktiv
3	Microsoft SQL Server
10	PostgreSQL

Beispiel VBScript:

```
Dim nDBSystemType : nDBSystemType = cRM.CurrentProject.DBSystemType
Dim nDBSystemMajorVersion : nDBSystemMajorVersion =
cRM.CurrentProject.DBSystemVersion / 65535
Dim nDBSystemMinorVersion : nDBSystemMinorVersion =
cRM.CurrentProject.DBSystemVersion Mod &HFFFF0000

Dim sDBSystemVersion : sDBSystemVersion = CStr(nDBSystemMajorVersion) & "." &
CStr(nDBSystemMinorVersion)
Dim sDBSystemType : sDBSystemType = ""

If (nDBSystemType = 0) Then
    sDBSystemType = "Keine Informationen - derzeit nicht mit einem Datenbankserver
verbunden"
ElseIf (nDBSystemType = 3) Then
    sDBSystemType = "Microsoft SQL Server"
ElseIf (nDBSystemType = 10) Then
    sDBSystemType = "PostgreSQL"
End If

Call cRM.DialogMessageBox("Das aktuell eingestellte Datenbanksystem ist: " &
sDBSystemType & " (Version: " & sDBSystemVersion & ")", "Project.DBSystemType",
vbOkOnly)
```

Beispiel C#-Script:

```
long dbSystemType = cRM.CurrentProject.DBSystemType;
long dbSystemMajorVersion = cRM.CurrentProject.DBSystemVersion / 65535;
long dbSystemMinorVersion = cRM.CurrentProject.DBSystemVersion % 0xFFFF0000;

string dbSystemVersion = dbSystemMajorVersion.ToString() + "." +
dbSystemMinorVersion.ToString();
string dbSystemTypeDetail = string.Empty;

if (dbSystemType == 0)
{
    dbSystemTypeDetail = "Keine Informationen - derzeit nicht mit einem
Datenbankserver verbunden";
}
else if (dbSystemType == 3)
{
    dbSystemTypeDetail = "Microsoft SQL Server";
}
else if (dbSystemType == 10)
{
    dbSystemTypeDetail = "PostgreSQL";
}

cRM.DialogMessageBox("Das aktuell eingestellte Datenbanksystem ist: " +
dbSystemTypeDetail + " (Version: " + dbSystemVersion + ")",
"Project.DBSystemType", 0);
```

DBSystemVersion, read-only

Beschreibung:

Rückgabe der Datenbankserver-Version der installierten Anwendung.

Typ:

Long

Wert	Beschreibung
HIWORD	Hauptversion (Dividiert durch 65535)
LOWORD	Nebenversion (Modulo 0xFFFF0000)

Beispiel VBScript:

```
Dim nDBSystemType : nDBSystemType = cRM.CurrentProject.DBSystemType
Dim nDBSystemMajorVersion : nDBSystemMajorVersion =
cRM.CurrentProject.DBSystemVersion / 65535
Dim nDBSystemMinorVersion : nDBSystemMinorVersion =
cRM.CurrentProject.DBSystemVersion Mod &HFFFF0000

Dim sDBSystemVersion : sDBSystemVersion = CStr(nDBSystemMajorVersion) & "." &
CStr(nDBSystemMinorVersion)
Dim sDBSystemType : sDBSystemType = ""

If (nDBSystemType = 0) Then
    sDBSystemType = "Keine Informationen - derzeit nicht mit einem Datenbankserver
verbunden"
ElseIf (nDBSystemType = 3) Then
    sDBSystemType = "Microsoft SQL Server"
ElseIf (nDBSystemType = 10) Then
    sDBSystemType = "PostgreSQL"
End If

Call cRM.DialogMessageBox("Das aktuell eingestellte Datenbanksystem ist: " &
sDBSystemType & " (Version: " & sDBSystemVersion & ")", "Project.DBSystemVersion",
vbOkOnly)
```

Beispiel C#-Script:

```
long dbSystemType = cRM.CurrentProject.DBSystemType;
long dbSystemMajorVersion = cRM.CurrentProject.DBSystemVersion / 65535;
long dbSystemMinorVersion = cRM.CurrentProject.DBSystemVersion % 0xFFFF0000;

string dbSystemVersion = dbSystemMajorVersion.ToString() + "." +
dbSystemMinorVersion.ToString();
string dbSystemTypeDetail = string.Empty;

if (dbSystemType == 0)
{
    dbSystemTypeDetail = "Keine Informationen - derzeit nicht mit einem
Datenbankserver verbunden";
}
else if (dbSystemType == 3)
{
    dbSystemTypeDetail = "Microsoft SQL Server";
}
else if (dbSystemType == 10)
{
    dbSystemTypeDetail = "PostgreSQL";
}

cRM.DialogMessageBox("Das aktuell eingestellte Datenbanksystem ist: " +
dbSystemTypeDetail + " (Version: " + dbSystemVersion + ")",
"Project.DBSystemType", 0);
```

Description, read-only

Beschreibung:

Gibt die Beschreibung des Projektes zurück.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Folgende Projektbeschreibung wurde hinterlegt: " &
cRM.CurrentProject.Description, "Project.Description", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Folgende Projektbeschreibung wurde hinterlegt: " +
cRM.CurrentProject.Description, "Project.Description", 0);
```

DocMngr

Beschreibung:

Mit Hilfe des **DocMngr**-Objekts können Dateien im konfigurierten Dokumentencontainer bzw. Dokumentenfeldern abgelegt werden.

Typ:

DocMngr

Beispiel VBScript:

```
Dim oDocMngr : Set oDocMngr = cRM.CurrentProject.DocMngr
' ...
Set oDocMngr = Nothing
```

Beispiel C#-Script:

```
DocMngr docMngr = cRM.CurrentProject.DocMngr;
// ...
docMngr.Dispose();
```

EmailTool

Beschreibung:

Mit Hilfe des **EmailTool**-Objekts besteht Zugriff auf ein in combit CRM integriertes externes Tool für professionelles und leistungsfähiges Kampagnenmanagement.

Hinweis: Wenn in der Projektkonfiguration der E-Mail-Versand über "<Integrierte E-Mail-Anbindung>" eingestellt ist, liefert die Eigenschaft **Nothing** zurück.

Typ:

EmailTool

Beispiel VBScript:

```
Dim oEmailTool : Set oEmailTool = cRM.CurrentProject.EmailTool
' ...
Set oEmailTool = Nothing
```

Beispiel C#-Script:

```
EmailTool emailTool = cRM.CurrentProject.EmailTool;
// ...
emailTool.Dispose();
```

FilePath, read-only

Beschreibung:

Gibt den Verzeichnispfad des Projektes zurück.

z. B. C:\Program Files (x86)\combit\combit CRM\Solutions\Large\combit_Large.crm

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Der komplette Pfad zur Projekt-Datei lautet wie folgt:
" & cRM.CurrentProject.FilePath, "Project.FilePath", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Der komplette Pfad zur Projekt-Datei lautet wie folgt: " +
cRM.CurrentProject.FilePath, "Project.FilePath", 0);
```

ID, read-only

Beschreibung:

Gibt die eindeutige ID des Projektes zurück.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Die eindeutige ID des Projektes lautet: " &
cRM.CurrentProject.ID, "Project.ID", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Die eindeutige ID des Projektes lautet: " +
cRM.CurrentProject.ID, "Project.ID", 0);
```

IsDirty, read-only

Beschreibung:

Liefert zurück, ob es zurzeit noch ungespeicherte Änderungen an der Projektdatei gibt.

Typ:

Bool

Beispiel VBScript:

```
If (cRM.CurrentProject.IsDirty = True) Then
    Call cRM.DialogMessageBox("Derzeit gibt es noch ungespeicherte Änderungen an
der Projektdatei.", "Project.IsDirty", vbOkOnly)
End If
```

Beispiel C#-Script:

```
if (cRM.CurrentProject.IsDirty == true)
{
    cRM.DialogMessageBox("Derzeit gibt es noch ungespeicherte Änderungen an der
Projektdatei.", "Project.IsDirty", 0);
}
```

Name, read-only

Beschreibung:

Gibt den Projektnamen des aktiven Projektes zurück.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Der Name der Projekt-Datei lautet wie folgt: " &
cRM.CurrentProject.Name, "Project.Name", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Der Name der Projekt-Datei lautet wie folgt: " +
cRM.CurrentProject.Name, "Project.Name", 0);
```

ProjectDir, read-only

Beschreibung:

Gibt das Projektverzeichnis des aktiven Projektes mit abschließendem Backslash zurück.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Das Projektverzeichnis des aktuell geladenen Projekts
lautet: " & cRM.CurrentProject.ProjectDir, "Project.ProjectDir", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Das Projektverzeichnis des aktuell geladenen Projekts
lautet: " + cRM.CurrentProject.ProjectDir, "Project.ProjectDir", 0);
```

Users, read-only

Beschreibung:

Liefert das **Users**-Objekt für den Zugriff auf die Benutzerstammdaten zurück.

Typ:

Users

Beispiel VBScript:

```
Dim oUsers : Set oUsers = cRM.CurrentProject.Users
' ...
Set oUsers = Nothing
```

Beispiel C#-Script:

```
Users users = cRM.CurrentProject.Users;
// ...
users.Dispose();
```

timemanager

Beschreibung:

Liefert ein Termin-Objekt vom Typ **timemanager** zurück.

Typ:

Timemanager

Beispiel VBScript:

```
Dim oTimeManager : Set oTimeManager = cRM.CurrentProject.timemanager
' ...
Set oTimeManager = Nothing
```

Beispiel C#-Script:

```
TimeManager timeManager = cRM.CurrentProject.TimeManager;
// ...
timeManager.Dispose();
```

ViewConfigs

Beschreibung:

Gibt die konfigurierten Ansichten im aktuellen Projekt als Objekt vom Typ **ListViewConfigs** zurück.

Typ:

ListViewConfigs

Beispiel VBScript:

```
Dim oListViewConfigs : Set oListViewConfigs = cRM.CurrentProject.ViewConfigs
' ...
Set oListViewConfigs = Nothing
```

Beispiel C#-Script:

```
ListViewConfigs listViewConfigs = cRM.CurrentProject.ViewConfigs;
// ...
listViewConfigs.Dispose();
```

3.36.2 Methoden

CheckVATID

Beschreibung:

Führt online beim Bundeszentralamt für Steuern eine qualifizierte Bestätigungsanfrage einer ausländischen Umsatzsteuer-Identifikationsnummer durch (vgl. §6a Ziff. 3 UstG). Beachten Sie dazu die Hinweise auf der Website des Bundeszentralamt für Steuern.

Was ist der Unterschied zwischen einer einfachen und einer qualifizierten Bestätigung?

Bei einer einfachen Bestätigung erhalten Sie Auskunft darüber, ob eine ausländische USt-IdNr. zum Zeitpunkt der Anfrage in dem Mitgliedstaat, der sie erteilt hat, gültig ist.

Eine einfache Bestätigung muss zwingend vor einer qualifizierten Bestätigung durchgeführt werden.

Bei einer qualifizierten Bestätigung können Sie darüber hinaus abfragen, ob die von Ihnen mitgeteilten Angaben zu Firmenname (einschließlich der Rechtsform), Firmenort, Postleitzahl und Straße mit den in der Unternehmerdatei des jeweiligen EU-Mitgliedstaates registrierten Daten übereinstimmen.

Hinweis: Eine Bekanntgabe der in der Unternehmerdatei des jeweiligen EU-Mitgliedstaates registrierten Daten ist nicht möglich.

(Quelle: https://www.bzst.de/DE/Unternehmen/Identifikationsnummern/AuslaendischeUSt-IdNr/auslaendische_ust_idnr_node.html#js-toc-entry1)

Wichtig: Zum Aufruf der Methode wird eine Internetverbindung vorausgesetzt.

Parameter:

Parametername	Typ	Beschreibung	Pflicht
Parameter für die Abfrage			
VATID_1	String	Ihre deutsche Ust-IdNr.	Ja
VATID_2	String	Anzufragende ausländische Ust-IdNr.	Ja
Company	String	Name der anzufragenden Firma einschl. Rechtsform	Ja
City	String	Ort der anzufragenden Firma	Ja
ZIP	String	Postleitzahl der anzufragenden Firma	Nein
Street	String	Straße und Hausnummer der anzufragenden Firma	Nein
Print	Bool	True: Mit schriftlicher amtlicher Bestätigungsmitteilung	Nein

Wichtig: Wird für den Parameter **Print** True übergeben, wird die amtliche Bestätigungsmitteilung schriftlich per Post an die Adresse Ihrer deutschen Ust-IdNr. Versendet.

Hinweis: Der Parameter **Print** wird seitens des Bundeszentralamtes für Steuern (BZSt) derzeit (Stand 02/2023) ignoriert.

Parametername	Typ	Beschreibung
Parameter, die mit Ergebnissen gefüllt werden		
Date	String	Datum der Anfrage (Format: tt.mm.jjjj)
Time	String	Uhrzeit der Anfrage (Format: hh:mm:ss)
ErrorCode	String	Fehlernummer der Anfrage (Übersicht der ErrorCodes)
ResultName	String	Ergebnis für den angefragten Namen der Firma
ResultCity	String	Ergebnis für den angefragten Ort der Firma
ResultZIP	String	Ergebnis für die angefragte Postleitzahl der Firma
ResultStreet	String	Ergebnis für die angefragte Straße der Firma
ValidFrom	String	Wird nur bei ErrorCode 203 bzw. 204 angegeben. Beginn der Gültigkeit der ausländischen Ust-IdNr. (Format: tt.mm.jjjj)
ValidTo	String	Wird nur bei ErrorCode 204 angegeben. Ende der Gültigkeit der ausländischen Ust-IdNr. (Format: tt.mm.jjjj)
ConnectionError	String	Enthält eine Fehlermeldung, wenn es zu einem Fehler während der HTTP-Anfrage gekommen ist
ResponseStream	String	Enhält die komplette Antwort, die vom Server zurückgegeben wird

Folgende Werte sind als Ergebnis für die Parameter **ResultName, ResultCity, ResultZIP** und **ResultStreet** möglich:

A = stimmt überein

B = stimmt nicht überein

C = nicht angefragt

D = vom EU-Mitgliedsstaat nicht mitgeteilt

Rückgabewert:

Bool

True: Wird zurückgeliefert, wenn

- der ErrorCode den Wert 200 hat und die Parameter **Company**, **City**, **ZIP** und **Street** übergeben wurden und die Ergebnisparameter **ResultName**, **ResultCity**, **ResultZIP** und **ResultStreet** den Wert 'A' haben.
- der ErrorCode den Wert 200 hat und die Ergebnisparameter **ResultName** und **ResultCity** den Wert 'A' haben und **ResultZIP** und **ResultStreet** den Wert 'C' haben, da sie nicht übergeben wurden.

Beispiel VBScript:

```
Dim oProject : Set oProject = cRM.CurrentProject
Dim oCurrentInputForm : Set oCurrentInputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2)

Dim sVATID_1, sVATID_2, sCompany, sCity, sZIP, sStreet, bPrint, sDate, sTime,
sErrorCode, sResultName, sResultCity, sResultZIP, sResultStreet, sValidFrom,
sValidTo, sConnectionError, sResponseStream

sVATID_1 = oProject.CompanyInfo.VATID
sVATID_2 = oCurrentInputForm.GetContentsByName(sColumnName_VAT_ID)
sCompany = oCurrentInputForm.GetContentsByName(sColumnName_Company)
sCity = oCurrentInputForm.GetContentsByName(sColumnName_City)
sZIP = oCurrentInputForm.GetContentsByName(sColumnName_ZIP)
sStreet = oCurrentInputForm.GetContentsByName(sColumnName_Street)
bPrint = bPrintSetting

Dim bResult : bResult = oProject.CheckVATID(sVATID_1, sVATID_2, sCompany, sCity,
sZIP, sStreet, bPrint, sDate, sTime, sErrorCode, sResultName, sResultCity,
sResultZIP, sResultStreet, sValidFrom, sValidTo, sConnectionError,
sResponseStream)

Set oCurrentInputForm = Nothing
```

```
Set oProject = Nothing
```

Beispiel C#-Script:

```
Project project = cRM.CurrentProject;
InputForm currentInputForm = project.ActiveViews.ActiveView.CurrentInputForm(2);

string vatID_1 = project.CompanyInfo.VatID;
string vatID_2 = currentInputForm.GetContentsByName("ColumnName_VAT_ID");
string company = currentInputForm.GetContentsByName("ColumnName_Company");
string city = currentInputForm.GetContentsByName("ColumnName_City");
string zip = currentInputForm.GetContentsByName("ColumnName_ZIP");
string street = currentInputForm.GetContentsByName("ColumnName_Street");
bool print = false;

string date = null;
string time = null;
string errorCode = null;
string resultName = null;
string resultCity = null;
string resultZip = null;
string resultStreet = null;
string validFrom = null;
string validTo = null;
string connectionError = null;
string responseStream = null;
bool result = project.CheckVATID(vatID_1, vatID_2, company, city, zip, street,
print, string date, time, errorCode, resultName, resultCity, resultZip,
resultStreet, validFrom, validTo, connectionError, responseStream);

currentInputForm.Dispose();
project.Dispose();
```

ExecuteInstantReportByDescription

Beschreibung:

Führt einen abgespeicherten Sofortbericht anhand seiner Beschreibung aus.

Wichtig: Sofortberichte, die als Datenbasis einen einzelnen Datensatz haben, können mit dieser Methode nicht exportiert werden. In diesem Fall muss die Methode **ExecuteInstantReportByName** aus dem **RecordSet** Objekt verwendet werden. Bei Sofortberichten, die als Datenbasis "aktueller Filter" haben, werden alle Datensätze gedruckt.

Wenn für den Parameter Media der Wert "PRV" und für den Parameter OutputFileName ein leerer String übergeben wird, wird der Sofortbericht in die Berichtsansicht gedruckt. Wird in OutputFileName aber ein Name und Pfad übergeben, wird ein einfacher Export nach *.ll durchgeführt und das Ergebnis nicht angezeigt.

Parameter:

Parametername	Typ	Beschreibung
Media	String	Die zur Verfügung stehenden Ausgabe-Medien finden Sie im Kapitel **Ausgabe-Medium.**
OutputFileName	String	Evtl. Name und Pfad der zu erzeugenden Ausgabe-Datei bei Ausgabe-Medien wie z. B. "RTF" oder "HTML". In allen anderen Fällen kann ein leerer String übergeben werden. Bei der Erzeugung mehrerer Dateien in der Ausgabe (z. B. Bildexport) kann "%d" als Platzhalter für die Seitennummer verwenden werden.
ReportName	String	Beschreibung des Sofortberichts. Groß- / Kleinschreibung wird beachtet! Die Beschreibung ist dabei die im Feld "Beschreibung" in der Sofortberichte-Konfiguration definierte.

Silent	Bool	Optional. Legt fest, ob der Druck (nach Möglichkeit) ohne Benutzerinteraktion erfolgen soll. Voreinstellung: True

Typ:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Call cRM.CurrentProject.ExecuteInstantReportByDescription("PRV", "", "Firmen -
Auswertungen | Kunden", True)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ExecuteInstantReportByDescription("PRV", "", "Firmen -
Auswertungen | Kunden", true);
```

ExecuteInstantReportByName

Beschreibung:

Führt einen abgespeicherten Sofortbericht anhand seines Namens aus.

Wichtig: Sofortberichte, die als Datenbasis einen einzelnen Datensatz haben, können mit dieser Methode nicht exportiert werden. In diesem Fall muss die Methode **ExecuteInstantReportByName** aus dem **RecordSet** Objekt verwendet werden. Bei Sofortberichten, die als Datenbasis "aktueller Filter" haben, werden alle Datensätze gedruckt.

Wenn für den Parameter Media der Wert "PRV" und für den Parameter OutputFileName ein leerer String übergeben wird, wird der Sofortbericht in die Berichtsansicht gedruckt. Wird in OutputFileName aber ein Name und Pfad übergeben, wird ein einfacher Export nach *.ll durchgeführt und das Ergebnis nicht angezeigt.

Parameter:

Parametername	Typ	Beschreibung
Media	String	Die zur Verfügung stehenden Ausgabe-Medien finden Sie im Kapitel **Ausgabe-Medium.**
OutputFileName	String	Evtl. Name und Pfad der zu erzeugenden Ausgabe-Datei bei Ausgabe-Medien wie z. B. "RTF" oder "HTML". In allen anderen Fällen kann ein leerer String übergeben werden. Bei der Erzeugung mehrerer Dateien in der Ausgabe (z. B. Bildexport) kann "%d" als Platzhalter für die Seitennummer verwenden werden.
ReportName	String	Name des Sofortberichts. Groß- / Kleinschreibung wird beachtet! Der Name ist dabei der im Feld "Name für Scripte/Workflows" in der Sofortberichte-Konfiguration definierte.
Silent	Bool	Optional. Legt fest, ob der Druck (nach Möglichkeit) ohne Benutzerinteraktion erfolgen soll. Voreinstellung: True

Typ:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution
```

```
Call cRM.CurrentProject.ExecuteInstantReportByName("PRV", "", "Kontakte_Dossier",
True)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ExecuteInstantReportByName("PRV", "", "Kontakte_Dossier",
true);
```

ExecuteScriptByCode

Beschreibung:

Führt den übergebenen Scriptcode als internes Script aus.

Parameter:

Parametername	Typ	Beschreibung
sCode	String	Scriptcode

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim sScriptCode : sScriptCode = "cRM.DialogMessageBox(""Scriptausführung ber
ExecuteScriptByCode"", ""Project.ExecuteScriptByCode"", vbOkOnly)"
Call cRM.CurrentProject.ExecuteScriptByCode(sScriptCode)
```

Beispiel C#-Script:

```
string scriptCode = @"cRM.DialogMessageBox(""Scriptausfhrung ber
ExecuteScriptByCode"", ""Project.ExecuteScriptByCode"", 0)";
cRM.CurrentProject.ExecuteScriptByCode(scriptCode);
```

ExecuteScriptByFilename

Beschreibung:

Führt das übergebene Script aus.

Parameter:

Parametername	Typ	Beschreibung
sFilePath	String	Dateiname inkl. Pfad des Scripts

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ExecuteScriptByFileName("%PRJDIR%\Scripts\cRMProjectToVisio.vbs
")
```

Beispiel C#-Script:

```
cRM.CurrentProject.ExecuteScriptByFilename(@"%PRJDIR%\Scripts\cRMProjectToVisio.vb
s");
```

ExecuteWorkflowByFilename

Beschreibung:

Führt den übergebenen Workflow aus.

Wichtig: Diese Methode steht erst ab der Professional-Edition zur Verfügung.

Parameter:

Parametername	Typ	Beschreibung
sFilePath	String	Dateiname inkl. Pfad des Workflows

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ExecuteWorkflowByFileName("%PRJDIR%\Workflows\Infoversand.xoml"
)
```

Beispiel C#-Script:

```
cRM.CurrentProject.ExecuteWorkflowByFilename(@"%PRJDIR%\Workflows\Infoversand.xoml
");
```

FetchConfigFile

Beschreibung:

Holt die projektspezifischen Konfigurationsdateien.

Parameter:

Parametername	Typ	Beschreibung
sSrcFilePath	String	Dateiname in der cmbt_Files Tabelle in der System-Datenbank
sDestFilePath	String	Dateiname, unter dem die Datei abgespeichert werden soll.

Rückgabewert:

Bool

Wert	Beschreibung
True	Befehl wurde ausgeführt.
False	Befehl konnte nicht ausgeführt werden, dies deutet darauf hin, dass einer der Parameterwerte ungültig ist (Dateiname in cmbt_Files Tabelle kann nicht gefunden werden, Pfad für Speicherort ist ungültig, ...).

Beispiel VBScript:

```
Dim sSrcFilePath : sSrcFilePath = "Administrator.xml"
Dim sDestFilePath : sDestFilePath = "C:\temp\" & sSrcFilePath
Call cRM.CurrentProject.FetchConfigFile(sSrcFilePath, sDestFilePath)
```

Beispiel C#-Script:

```
string srcFilePath = "Administrator.xml";
string destFilePath = @"C:\temp\" + srcFilePath;
cRM.CurrentProject.FetchConfigFile(srcFilePath, destFilePath);
```

GetGlobalProperty

Beschreibung:

Liest eine globale Script-Einstellung aus, die zuvor mit **SetGlobalProperty** gesetzt wurde oder gibt den Standardwert zurück.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der Einstellung.
Default	String	Der Standardwert der Einstellung, der zurückgegeben wird, wenn keine Einstellung mit dem angegebenen Namen existiert.

Rückgabewert:

String

Beispiel VBScript:

```
Call cRM.CurrentProject.SetGlobalProperty("EigenerEintrag", "Wert")
Call cRM.CurrentProject.GetGlobalProperty("EigenerEintrag", "")
```

Beispiel C#-Script:

```
cRM.CurrentProject.SetGlobalProperty("EigenerEintrag", "Wert");
cRM.CurrentProject.GetGlobalProperty("EigenerEintrag", "");
```

GetMessageText

Beschreibung:

Lädt Texte aus einer Messages.txt im Projektverzeichnis. Falls die Oberflächen-Sprache auf Englisch eingestellt ist, dann aus der "Messages.0009.lng.txt".

Hinweis: Der Inhalt der Text-Dateien hat folgenden Aufbau:

<Name des Eintrags>=<Rückgabewert>

Zum Beispiel, Messages.txt:

SalesOpportunitySuccessful=Die Verkaufschance {0} wurde erfolgreich vom Benutzer {1} abgeschlossen. Soll nun ein neuer Datensatz in der Ansicht [2} erstellt werden, um eine Auftragsbestätigung zu erstellen?

Messages.0009.lng.txt

SalesOpportunitySuccessful=The sales opportunity {0} was successfully completed by the user {1}. Should a new record be created in the view {2} to create an order confirmation?

Die Platzhalter {0} bis {4} können über die nachfolgenden Parameter (sParam1 bis sParam5) befüllt werden. Erfolgt eine leere Parameterübergabe, so wird der Platzhalter nicht entfernt.

Im genannten Beispiel sähe der Scriptcode für eine deutsche Meldung wie folgt aus: cRM.CurrentProject.GetMessageText("", "SalesOpportunitySuccessful", "combit Software GmbH", "LFrisch", "Belege")

Wenn die Anwendungssprache Englisch ist, würde man die Methode wie folgt aufrufen: cRM.CurrentProject.GetMessageText("", "SalesOpportunitySuccessful", "combit Software GmbH", "LFrisch", "SalesDocuments")

Der Rückgabewert wäre bei deutscher Anwendungssprache wie folgt: Die Verkaufschance combit Software GmbH wurde erfolgreich vom Benutzer LFrisch abgeschlossen. Soll nun ein neuer Datensatz in der Ansicht Belege erstellt werden, um eine Auftragsbestätigung zu erstellen?

Bei englischer Anwendungssprache ist der Rückgabewert wie folgt: The sales opportunity combit Software GmbH was successfully completed by the user LFrisch. Should a new record be created in the view SalesDocuments to create an order confirmation?

Platzhalter können auch rekursiv in aufsteigender Reihenfolge verwendet werden. Dies bedeutet, dass ein übergebener Parameter1 selbst wiederum einen Wert wie {1} oder {2} enthalten kann, welcher durch die übergebenen Parameter2 und Parameter3 befüllt wird.

Erweiterte Formatspezifizierer oder andere Datentypen als Zeichenketten werden nicht unterstützt.

Alle Einträge der Messages.txt (bzw. Messages.0009.lng.txt) werden solange zeilenweise geladen und in einem Cache im Hauptspeicher gespeichert, bis die gesuchte ID gefunden wurde. Das bedeutet, dass häufig benutzte Einträge zu Beginn der Dateien verwendet werden sollten, um eine gute Balance zwischen Hauptspeicherverbrauch und Performance zu erhalten.

Die Einträge sind projektspezifisch, dies bedeutet, dass Änderungen in den Dateien Messages.txt oder Messages.0009.lng.txt erst beim Neuladen des Projektes Auswirkungen haben.

Bitte beachten Sie, dass die Dateien Messages.txt und Messages.0009.lng.txt als Kodierung ANSI, UTF-8 mit Byte Order Mark (BOM) oder UTF-16 verwenden, um die falsche Darstellung von z. B. Umlauten zu vermeiden.

Parameter:

Parametername	Typ	Beschreibung
sResourceFile	String	Dieser Parameter ist reserviert und wird ignoriert. Aktuell kann eine leere Zeichenkette übergeben werden.
sID	String	Name des Eintrags aus der Messages.txt (bzw. Messages.0009.lng.txt).
sParam1	String	Optional. Erster Parameter, um Informationen für den Platzhalter {0} innerhalb der Messages.txt (bzw. Messages.0009.lng.txt) zu befüllen.
sParam2	String	Optional. Zweiter Parameter, um Informationen für den Platzhalter {1} innerhalb der Messages.txt (bzw. Messages.0009.lng.txt) zu befüllen.
sParam3	String	Optional. Dritter Parameter, um Informationen für den Platzhalter {2} innerhalb der Messages.txt (bzw. Messages.0009.lng.txt) zu befüllen.
sParam4	String	Optional. Vierter Parameter, um Informationen für den Platzhalter {3} innerhalb der Messages.txt (bzw. Messages.0009.lng.txt) zu befüllen.
sParam5	String	Optional. Fünfter Parameter, um Informationen für den Platzhalter {4} innerhalb der Messages.txt (bzw. Messages.0009.lng.txt) zu befüllen.

Rückgabewert:

String

Beispiel VBScript:

```
If (cRM.UILanguageID = 7) Then
    Call cRM.CurrentProject.GetMessageText("", "SalesOppertunitySuccessfull",
"combit Software GmbH", "LFrisch", "Belege")
Else
    Call cRM.CurrentProject.GetMessageText("", "SalesOppertunitySuccessfull",
"combit Software GmbH", "LFrisch", "SalesDocuments")
End If
```

Beispiel C#-Script:

```
if (cRM.UILanguageID == 7)
```

```
{
    cRM.CurrentProject.GetMessageText("", "SalesOppertunitySuccessfull", "combit
Software GmbH", "LFrisch", "Belege");
}
else
{
    cRM.CurrentProject.GetMessageText("", "SalesOppertunitySuccessfull", "combit
Software GmbH", "LFrisch", "SalesDocuments");
}
```

GetSessionProperty

Beschreibung:

Liest eine Session-Variable aus, die zuvor mit **SetSessionProperty** gesetzt wurde oder gibt den Standardwert zurück.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der Einstellung.
Default	String	Der Standardwert der Einstellung, der zurückgegeben wird, wenn keine Einstellung mit dem angegebenen Namen existiert.

Rückgabewert:

String

Beispiel VBScript:

```
Call cRM.CurrentProject.SetSessionProperty("EigeneSessionVariable", "Wert")
Call cRM.CurrentProject.GetSessionProperty("EigeneSessionVariable", "")
Call cRM.CurrentProject.RemoveSessionProperty("EigeneSessionVariable")
```

Beispiel C#-Script:

```
cRM.CurrentProject.SetSessionProperty("EigeneSessionVariable", "Wert");
cRM.CurrentProject.GetSessionProperty("EigeneSessionVariable", "");
cRM.CurrentProject.RemoveSessionProperty("EigeneSessionVariable");
```

GetUserProperty

Beschreibung:

Liest eine benutzerspezifische Script-Einstellung aus, die zuvor mit **SetUserProperty** gesetzt wurde oder gibt den Standardwert zurück.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der Einstellung.
Default	String	Der Standardwert der Einstellung, der zurückgegeben wird, wenn keine Einstellung mit dem angegebenen Namen existiert.

Rückgabewert:

String

Beispiel VBScript:

```
Call cRM.CurrentProject.GetUserProperty("EigeneVariableProBenutzer", "Wert")
Call cRM.CurrentProject.SetUserProperty("EigeneVariableProBenutzer", "")
```

Beispiel C#-Script:

```
cRM.CurrentProject.GetUserProperty("EigeneVariableProBenutzer", "Wert");
cRM.CurrentProject.SetUserProperty("EigeneVariableProBenutzer", "");
```

IsValueBlacklisted

Beschreibung:

Gibt zurück, ob sich der übergebene Wert auf der Sperrliste befindet.

Parameter:

Parametername	Typ	Beschreibung
Value	String	Der zu prüfende Inhalt (E-Mail-Adressen, Fax- und Telefonnummern).

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim sPhoneNumber : sPhoneNumber =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetConten
tsByName("Phone_Private")
If (cRM.CurrentProject.IsValueBlacklisted(sPhoneNumber)) Then
    Call cRM.DialogMessageBox("Der geprüfte Inhalt (" & sPhoneNumber & ") befindet
sich auf der Sperrliste.", "Project.IsValueBlacklisted", vbOkOnly)
End If
```

Beispiel C#-Script:

```
string sPhoneNumber =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetConten
tsByName("Phone_Private");

if (cRM.CurrentProject.IsValueBlacklisted(sPhoneNumber) == true)
{
    cRM.DialogMessageBox("Der geprüfte Inhalt (" + sPhoneNumber + ") befindet sich
auf der Sperrliste.", "Project.IsValueBlacklisted", 0);
}
```

OpenActiveViewByName

Beschreibung:

Liefert eine aktive Ansicht über den Ansichtennamen.

Parameter:

Parametername	Typ	Beschreibung
ViewName	String	Name der aktiven Ansicht in der Anwendung

Rückgabewert:

View

Beispiel VBScript:

```
Dim sViewName : sViewName = "Kontakte"
If (cRM.CurrentProject.OpenActiveViewByName(sViewName) Is Nothing) Then
    Call cRM.CurrentProject.OpenNewViewByName(sViewName)
End If
```

Beispiel C#-Script:

```
string viewName = "Kontakte";

if (cRM.CurrentProject.OpenActiveViewByName(viewName) == null)
{
    cRM.CurrentProject.OpenNewViewByName(viewName);
}
```

OpenNewViewByName

Beschreibung:

Öffnet eine neue Ansicht über den Ansichtennamen.

Parameter:

Parametername	Typ	Beschreibung
ViewName	String	Name der neuen Ansicht
InitialCommand	String	Optional. Ermöglicht die Übergabe einer Sortierung mittels "SetSortOrder:n" (n = Nummer der gewünschten Sortierung, 0 = unsortiert, -n = invertierte Sortierung). Verhindert weitere Datenbankabfragen, die durch das Setzen der *SortOrder*-Eigenschaft ausgeführt werden müssten und erhöht so die Performance. Dieses Schlüsselwort muss als erstes angegeben werden, wenn es benutzt werden soll. Nachfolgende SetFilter*-Schlüsselworte können mit Leerzeichen oder Komma angehängt werden. Ein Filter kann mit Leerzeichen oder Komma getrennt dahinter aufgeführt werden. Der Filterausdruck muss dabei mit einer der folgenden Zeichenfolgen beginnen: "SetFilter: <Filterausdruck aus dem Filter Allgemein-Dialog>" "SetFilterByName:<Name für Scripte/Workflows des abgespeicherten Filters oder Pfad zu einer .crmshare-Datei mit enthaltenem Filterausdruck>" "SetFilterDirectSQL:<Freier SQL Filterausdruck>" "SetFilterByPrimaryKey:<Inhalt des Primärschlüsselfeldes>" Ersetzen Sie den Teil <...> durch den entsprechenden Wert.

Rückgabewert:

View

Beispiel VBScript:

```
Dim sViewName : sViewName = "Kontakte"
If (cRM.CurrentProject.OpenActiveViewByName(sViewName) Is Nothing) Then
    Call cRM.CurrentProject.OpenNewViewByName(sViewName)
End If
```

Beispiel C#-Script:

```
string viewName = "Kontakte";

if (cRM.CurrentProject.OpenActiveViewByName(viewName) == null)
{
    cRM.CurrentProject.OpenNewViewByName(viewName);
}
```

RemoveSessionProperty

Beschreibung:

Entfernt eine zuvor per SetSessionProperty angelegte Variable im Unterzweig "cRM.Project.Session".

Parameter:

Parametername	Typ	Beschreibung
Property	String	Name der Session-Variable.

Beispiel VBScript:

```
Call cRM.CurrentProject.SetSessionProperty("EigeneSessionVariable", "Wert")
Call cRM.CurrentProject.GetSessionProperty("EigeneSessionVariable", "")
Call cRM.CurrentProject.RemoveSessionProperty("EigeneSessionVariable")
```

Beispiel C#-Script:

```
cRM.CurrentProject.SetSessionProperty("EigeneSessionVariable", "Wert");
cRM.CurrentProject.GetSessionProperty("EigeneSessionVariable", "");
cRM.CurrentProject.RemoveSessionProperty("EigeneSessionVariable");
```

SetGlobalProperty

Beschreibung:

Setzt eine globale Script-Einstellung. Die Speicherung erfolgt in der Datei "global.ini" in der "cmbt_Files" Tabelle.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der Einstellung.
String	String	Der (neue) Wert der Einstellung.

Beispiel VBScript:

```
Call cRM.CurrentProject.SetGlobalProperty("EigenerEintrag", "Wert")
Call cRM.CurrentProject.GetGlobalProperty("EigenerEintrag", "")
```

Beispiel C#-Script:

```
cRM.CurrentProject.SetGlobalProperty("EigenerEintrag", "Wert");
cRM.CurrentProject.GetGlobalProperty("EigenerEintrag", "");
```

SetSessionProperty

Beschreibung:

Setzt eine Session-Variable im Formeleditor im Unterzweig "cRM.Project.Session".

Falls Datensatzrechte berücksichtigt werden sollen, dann verwenden Sie **SetSessionProperty** innerhalb des Ereignisses "Projekt wurde geöffnet".

Parameter:

Parametername	Typ	Beschreibung
Property	String	Name der Session-Variable.
Value	String	Inhalt der Session-Variable.

Beispiel VBScript:

```
Call cRM.CurrentProject.SetSessionProperty("EigeneSessionVariable", "Wert")
Call cRM.CurrentProject.GetSessionProperty("EigeneSessionVariable", "")
Call cRM.CurrentProject.RemoveSessionProperty("EigeneSessionVariable")
```

Beispiel C#-Script:

```
cRM.CurrentProject.SetSessionProperty("EigeneSessionVariable", "Wert");
cRM.CurrentProject.GetSessionProperty("EigeneSessionVariable", "");
cRM.CurrentProject.RemoveSessionProperty("EigeneSessionVariable");
```

SetUserProperty

Beschreibung:

Setzt eine benutzerspezifische Script-Einstellung. Die Speicherung erfolgt in der Datei "<BENUTZER>\user_scriptvars.ini" in der "cmbt_Files" Tabelle.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der Einstellung.
String	String	Der (neue) Wert der Einstellung.

Beispiel VBScript:

```
Call cRM.CurrentProject.GetUserProperty("EigeneVariableProBenutzer", "Wert")
Call cRM.CurrentProject.SetUserProperty("EigeneVariableProBenutzer", "")
```

Beispiel C#-Script:

```
cRM.CurrentProject.GetUserProperty("EigeneVariableProBenutzer", "Wert");
cRM.CurrentProject.SetUserProperty("EigeneVariableProBenutzer", "");
```

StoreConfigFile

Beschreibung:

Speichert die projektspezifischen Konfigurationsdateien der Anwendung.

Parameter:

Parametername	Typ	Beschreibung
sFilePath	String	Dateiname unter dem die Datei in der cmbt_Files Tabelle in der System-Datenbank abgelegt werden soll.
sLocalFilePath	String	Dateiname der lokalen Datei, die verwendet werden soll.

Rückgabewert:

Bool

Wert	Beschreibung
True	Befehl wurde ausgeführt.
False	Befehl konnte nicht ausgeführt werden, dies deutet darauf hin, dass einer der Parameterwerte ungültig ist (Dateiname in cmbt_Files Tabelle kann nicht gefunden werden, Pfad für Speicherort ist ungültig, ...).

Beispiel VBScript:

```
Dim sFilePath : sFilePath = "Administrator.xml"
Dim sLocalFilePath : sLocalFilePath = "C:\temp\" & sFilePath
Call cRM.CurrentProject.StoreConfigFile(sFilePath, sLocalFilePath)
```

Beispiel C#-Script:

```
string filePath = "Administrator.xml";
string localFilePath = @"C:\temp\" + filePath;
cRM.CurrentProject.StoreConfigFile(filePath, localFilePath);
```

3.37 Record Objekt

3.37.1 Eigenschaften

Deletable, read-only

Beschreibung:

Über diese Eigenschaft kann geprüft werden, ob ein Datensatz aufgrund von Datensatzrechten löschbar wäre.

Wichtig: Die Verwendung dieser Eigenschaft löst zusätzliche Abfragen an den Datenbankserver aus, welche beim anschließenden **Lock** bzw. **Delete** Aufruf in jedem Falle (nochmal) durchgeführt werden. Diese Eigenschaft sollte daher in Scripten nur mit Bedacht verwendet werden, sie könnte ggf. im Fall, dass **Lock**/**Delete** false liefern, eingesetzt werden, um die Ursache aufgrund von fehlenden Datensatzrechten zu bestimmen.

Die Methode prüft Datensatzrechte und das Ansichtsrecht *Datensatz löschen*. Im Fehlerfall erhält man keine visuelle Meldung, d.h. im Script muss eine visuelle Benachrichtigung erfolgen, wenn die Methode fehlschlägt.

Typ:

Bool

Beispiel VBScript:

```
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
If (oRecord.Deletable = True) Then
    Call oRecord.Delete()
End If
Set oRecord = Nothing
```

Beispiel C#-Script:

```
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
if (record.Deletable == true)
{
    record.Delete();
}
record.Dispose();
```

Editable, read-only

Beschreibung:

Über diese Eigenschaft kann geprüft werden, ob ein Datensatz aufgrund von Datensatzrechten bearbeitbar wäre.

Wichtig: Die Verwendung dieser Eigenschaft löst zusätzliche Abfragen an den Datenbankserver aus, welche beim anschließenden **Lock** bzw. **Delete** Aufruf in jedem Falle (nochmal) durchgeführt werden. Diese Eigenschaft sollte daher in Scripten nur mit Bedacht verwendet werden, sie könnte ggf. im Fall, dass **Lock**/**Delete** false liefern, eingesetzt werden, um die Ursache aufgrund von fehlenden Datensatzrechten zu bestimmen.

Die Methode prüft Datensatzrechte und das Ansichtsrecht *Datensatz ändern*. Im Fehlerfall erhält man keine visuelle Meldung, d.h. im Script muss eine visuelle Benachrichtigung erfolgen, wenn die Methode fehlschlägt.

Typ:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
If (oRecord.Editable = True) Then
    Call oRecord.Lock()
    Call oRecord.SetContentsByName("Name", "Soleil")
    Call oRecord.Save()
    Call oRecord.Unlock()
End If
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
if (record.Editable == true)
{
    record.Lock();
    record.SetContentsByName("Name", "Soleil");
    record.Save();
    record.Unlock();
}
```

3.37.2 Methoden

AddToBlacklist

Beschreibung:

Nimmt den Inhalt des übergebenen Feldes in die Sperrliste auf. Dabei wird die zugehörige Auto-Protokoll Aktion "In Sperrliste aufnehmen" ausgelöst.

Wichtig: Diese Methode nimmt nur Felder der internen Feldtypen 'E-Mail', 'Telefon', 'Mobiltelefon' und 'Telefax' entgegen.

Parameter:

Parametername	Typ	Beschreibung
Feldname	String	Feldname, dessen Inhalt in die Sperrliste aufgenommen werden soll.

Rückgabewert:

Bool

Wert	Beschreibung
True	Inhalt wurde in Sperrliste aufgenommen.
False	Inhalt konnte nicht in Sperrliste aufgenommen werden, z. B. weil ein Feld mit einem nicht unterstützten internen Feldtyp übergeben wurde.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.AddToBlac
klist("Phone_Private")
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.AddToBlac
klist("Phone_Private");
```

AddToFavorites

Beschreibung:

Fügt den aktuellen Datensatz den Favoriten hinzu.

> **Wichtig:** Diese Methode darf nicht aufgerufen werden, wenn das zugrundeliegende **Record** Objekt über **CurrentRecordBuffered** erzeugt wurde.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.AddToFavo
rites()
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.AddToFavo
rites();
```

AddToHistory

Beschreibung:

Fügt den aktuellen Datensatz dem Verlauf hinzu.

> **Wichtig:** Diese Methode darf nicht aufgerufen werden, wenn das zugrundeliegende **Record** Objekt über **CurrentRecordBuffered** erzeugt wurde.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.AddToHist
ory()
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.AddToHist
ory();
```

AddToPhoneManager

Beschreibung:

Fügt den aktuellen Datensatz dem phone manager hinzu.

Parameter:

Parametername	Typ	Beschreibung
Feldname	String	Feldname mit der Nummer, die hinzugefügt werden soll.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.AddToPhon
eManager("Phone_Private")
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.AddToPhon
eManager("Phone_Private");
```

Delete

Beschreibung:

Löscht einen Datensatz.

Hinweis: Die Methode prüft das Ansichtsrecht *Datensatz löschen*. Im Fehlerfall erhält man keine visuelle Meldung, d.h. im Script muss eine visuelle Benachrichtigung erfolgen, wenn die Methode fehlschlägt.

Beim Löschen eines Datensatzes wird berücksichtigt, ob der Papierkorb für die Ansicht aktiviert wurde.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
If (oRecord.Deletable = True) Then
    Call oRecord.Delete()
End If
Set oRecord = Nothing
```

Beispiel C#-Script:

```
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
if (record.Deletable == true)
{
    record.Delete();
}
record.Dispose();
```

DialNumber

Beschreibung:

Wählt die übergebene Telefonnummer und entspricht insofern der manuellen Wahl.

Wichtig: Diese Methode darf nicht aufgerufen werden, wenn das zugrundeliegende **Record** Objekt über **CurrentRecordBuffered** erzeugt wurde.

Parameter:

Parametername	Typ	Beschreibung
Feldname	String	Feldname mit der zu wählenden Nummer.

Hinweis: Enthält der Parameter 'Feldname' als erstes Zeichen keinen Buchstaben, so wird der Inhalt als direkt übergebene Telefonnummer interpretiert. Dies ermöglicht Datensatz-spezifische (notwendig für z. B. Autoprotokolle) Telefonwahl, ohne dass die Telefonnummer selbst in einem Datensatz-Feld stehen muss.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.DialNumbe
r("Phone_Private")
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.DialNumbe
r("Phone_Private");
```

GetContentsByFormula

Beschreibung:

Liefert das Ergebnis einer Formel auf Basis des aktuellen Datensatzes zurück. Die Felder des Datensatzes stehen dabei als Variablen zur Verfügung.

Hinweis: Wird ein formatiertes Notizenfeld abgerufen, so wird immer der entsprechende HTML-Code zurückgeliefert.

Parameter:

Parametername	Typ	Beschreibung
Formula	String	Auszuwertende Formel. Weitere Informationen finden Sie in der Hilfe des Druckvorlagendesigners.

Hinweis: Für Feldnamen, die in einer Formel verwendet werden, muss der physikalische Feldname verwendet werden. Das Verwenden von Feldaliasen wird ausdrücklich nicht empfohlen, da eine Änderung des Feldalias (ggf. nur indirekt, z. B. durch die Änderung eines 1:1 Relationsalias) dazu führt, dass das Script nicht mehr korrekt funktioniert.

Tipp: Eine funktionierende Formel lässt sich bspw. in den Ansichteneigenschaften der betreffenden Ansicht im Reiter "Datensatzverweis" in einem der Formel-Dialoge zusammenstellen. Im Reiter "Felder" können Sie im Zweifel prüfen, welcher Feldalias zu welchem phys. Feldnamen gehört. Kopieren Sie anschließend die zusammengestellte Formel und verwenden diese im Script.

Rückgabewert:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetConten
tsByFormula("""Firma '"" + Company + ""'""")
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution
```

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetContentsByFormula(@"""Firma '"" + Company + ""'""");
```

GetContentsByName

Beschreibung:

Liefert den Inhalt des Feldes als Zeichenkette zurück, dessen physikalischer Feldname übergeben wurde.

Hinweis: Wird ein formatiertes Notizenfeld abgerufen, so wird immer der entsprechende HTML-Code zurückgeliefert.

Um Aggregationsfelder anzusprechen verwendet man als Feldnamen folgende Struktur:

<Name des Primärschlüsselfeldes>.$Aggregate_<Name der Ansicht>.ID.<Physikalischer Name des Aggregationsfeldes>

Beispiel: ID.$Aggregate_Kontakte.ID.ActivityCount

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes.

Rückgabewert:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim sCompany : sCompany =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetContentsByName("Company")
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

string company =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetContentsByName("Company");
```

GetContentsByNameToFile

Beschreibung:

Das angegebene BLOB-Feld wird ausgelesen und dessen Inhalt unter dem angegebenen Dateinamen gespeichert.

Parameter:

Parametername	Typ	Beschreibung
Feldname	String	Physikalischer Name des gewünschten (BLOB)-Feldes.
Path	String	Speicherpfad der angelegten Datei.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Auslesen und Übertragen eines Dokuments in das Dateisystem. Dieses Beispiel
basiert auf der Aktivitäten-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Dim sDocumentFileName : sDocumentFileName =
oRecord.GetContentsByName("Document_FileName")
Dim sDocumentFileType : sDocumentFileType =
oRecord.GetContentsByName("Document_FileType")

Call oRecord.GetContentsByNameToFile("Document_Embedded", "C:\temp\" &
sDocumentFileName & "." & sDocumentFileType)

Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Auslesen und Übertragen eines Dokuments in das Dateisystem. Dieses Beispiel
basiert auf der Aktivitäten-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
string documentFileName = record.GetContentsByName("Document_FileName");
string documentFileType = record.GetContentsByName("Document_FileType");

record.GetContentsByNameToFile("Document_Embedded", @"C:\temp\" + documentFileName
+ "." + documentFileType);

record.Dispose();
```

GetContentsValueByFormula

Beschreibung:

Liefert das Ergebnis einer Formel auf Basis des aktuellen Datensatzes entsprechend des zurückgelieferten Datentyps der Formel zurück. Die Felder des Datensatzes stehen dabei als Variablen zur Verfügung.

Hinweis: Wird ein formatiertes Notizenfeld abgerufen, so wird immer der entsprechende HTML-Code zurückgeliefert.

Parameter:

Parametername	Typ	Beschreibung
Formula	String	Auszuwertende Formel. Weitere Informationen finden Sie in der Hilfe des Druckvorlagendesigners.

Hinweis: Für Feldnamen, die in einer Formel verwendet werden, muss der physikalische Feldname verwendet werden. Das Verwenden von Feldaliasen wird ausdrücklich nicht empfohlen, da eine Änderung des Feldalias (ggf. nur indirekt, z. B. durch die Änderung eines 1:1 Relationsalias) dazu führt, dass das Script nicht mehr korrekt funktioniert.

Tipp: Eine funktionierende Formel lässt sich bspw. in den Ansichteneigenschaften der betreffenden Ansicht im Reiter "Datensatzverweis" in einem der Formel-Dialoge zusammenstellen. Im Reiter "Felder" können Sie im Zweifel prüfen, welcher Feldalias zu welchem phys. Feldnamen gehört. Kopieren Sie anschließend die zusammengestellte Formel und verwenden diese im Script.

Rückgabewert:

Variant

Hinweis: Bitte beachten Sie, dass "" nicht gleich NULL ist.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetContentsValueByFormula("""Firma '"" + Company + ""'""")
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetContentsValueByFormula(@"""Firma '"" + Company + ""'""");
```

GetContentsValueByName

Beschreibung:

Liefert den Inhalt entsprechend des Feldtyps des Feldes zurück, dessen Feldname übergeben wurde, z. B. Datumzeit-Typen als Datumsvariable, numerische Typen als numerische Variable etc. Somit werden bspw. Lokalisierungsprobleme (Komma oder Punkt als Dezimalzeichen? Datumsformatierung?) bei der Weiterverarbeitung des Wertes vermieden.

Hinweis: Wird ein formatiertes Notizenfeld abgerufen, so wird immer der entsprechende HTML-Code zurückgeliefert.

Um Aggregationsfelder anzusprechen verwendet man als Feldnamen folgende Struktur:

<Name des Primärschlüsselfeldes>.$Aggregate_<Name der Ansicht>.ID.<Name des Aggregationsfeldes>

Beispiel: ID.$Aggregate_Kontakte.ID.ActivityCount

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes.

Rückgabewert:

Variant

Hinweis: Bitte beachten Sie, dass "" nicht gleich NULL ist.

Beispiel VBScript:

```
' Durchläuft alle Datensätze in einem kopierten RecordSet und summiert die
Gesamtumsätze. Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-
Solution

Dim oRecordSetCopy : Set oRecordSetCopy =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy
Dim oRecord
Dim nTurnover : nTurnover = 0

If (oRecordSetCopy.MoveFirst() = True) Then
    Set oRecord = oRecordSetCopy.CurrentRecord

    Do
        nTurnover = nTurnover + oRecord.GetContentsValueByName("Turnover")
    Loop Until Not oRecordSetCopy.MoveNext()
```

```
    Set oRecord = Nothing
End If

Call cRM.DialogMessageBox("Der Gesamtumsatz der aktuell angezeigten Datensätze
beträgt: " & CStr(nTurnover) & " EUR.", "RecordSet.MoveFirst", vbOkOnly)

Set oRecordSetCopy = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Datensätze in einem kopierten RecordSet und summiert die
Gesamtumsätze. Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-
Solution

RecordSet recordSetCopy =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy();
Record record;
double turnover = 0;

if (recordSetCopy.MoveFirst() == true)
{
    record = recordSetCopy.CurrentRecord;

    do
    {
        turnover = turnover + (double)record.GetContentsValueByName("Turnover");
    } while (!recordSetCopy.MoveNext());

    record.Dispose();

}

cRM.DialogMessageBox("Der Gesamtumsatz der aktuell angezeigten Datensätze beträgt:
" + turnover.ToString() + " EUR.", "RecordSet.MoveFirst", 0);

recordSetCopy.Dispose();
```

GetRecordRefDescription

Beschreibung:

Liefert die evaluierte Datensatzverweis-Formel für Kommentar/Betreff zurück. Im Fehlerfall wird ein leerer String zurückgeliefert.

Rückgabewert:

String

Beispiel VBScript:

```
Dim sRecordReference : sRecordReference =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetRecord
RefDescription()
```

Beispiel C#-Script:

```
string recordRefDescription =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetRecord
RefDescription();
```

GetRelationalRecordSet

Beschreibung:

Liefert ein **RecordSet** aufgrund einer relationalen Struktur zurück.

Hinweis: Wir empfehlen, nach Erzeugung eines **RecordSet**-Objektes zunächst mittles Aufruf der Methode "MoveFirst" die Existenz mindestens eines **Record**-Objektes zu überprüfen.

Wichtiger Hinweis für die Verwendung des Parameters CursorModel mit dem Wert 2 (forward-only) unter Microsoft SQL Server: Die Datensätze eines forward-only-RecordSets müssen nach dessen Erstellung direkt und unmittelbar über eine "GotoNext"-Schleife ohne Interaktion vollständig durchlaufen werden. Anderenfalls kann es, wenn das RecordSet viele Zeilen enthält, am Datenbankserver zu einem *ASYNC_NETWORK_IO*-Wartezustand kommen, der dann andere Abfragen (vor allem Änderungen) auf dieselbe Tabelle blockiert.

Parameter:

Parametername	Typ	Beschreibung
Relation	String	Bezeichnung der Relation, z. B. ID.Kontakte.CompanyID

CursorModel	Long	Optional.
		Ermöglicht die Spezifikation des Datenbankcursormodells, das für den zurückgegebenen RecordSet genutzt werden soll.
		
		Werte:
		0 (Standardwert): Erzeugt ein RecordSet mit einem Datenbankcursormodell, welches innerhalb der combit CRM-Projektdatei spezifiziert werden kann:
		
		`...`
		`<!-- DATA -->`
		`<profile>`
		`<list name="">`
		`<list name="ExtendedSettings">`
		`<item name="COMRecordSetCursorDefault">2</item>`
		`</list>`
		`...`
		
		Wird in der combit CRM-Projektdatei keine Eigenschaft COMRecordSetCursorDefault gefunden, so wird immer ein fully-dynamic RecordSet erzeugt. Mögliche Werte für die Eigenschaft sind: 1 – fully-dynamic RecordSet, 2 – forward-only RecordSet.
		
		1: Erzeugt ein RecordSet mit fully-dynamic Datenbankcursor.
		
		2: Erzeugt ein RecordSet mit forward-only Datenbankcursor. Ermöglicht deutliche Performance-Gewinne, insbesondere bei großen Datenmengen und komplexen Filterausdrücken, erlaubt aber lediglich das einmalige Durchlaufen in Vorwärtsrichtung durch den RecordSet.

		Die Methoden **RecordSet.DialogSelectRecord**, **RecordSet.DialogSelectRecordMultiple**, **RecordSet.SendBulkMail** (bei anzuzeigendem integrierten Mail-Editor), **RecordSet.MovePrevious**, **RecordSet.MoveLast**, **InputForm.DialogSelectRecordDropDown** werden einen Scriptfehler werfen, wenn diese für einen forward-only **RecordSet** genutzt werden. Für diese Methoden muss der **RecordSet** explizit ohne forward-only (Werte 0 oder 1) erzeugt werden.

Rückgabewert:

RecordSet

Beispiel VBScript:

```
' Zählt alle Datensätze in einem RecordSet. Dieses Beispiel basiert auf der
Kontakte-Ansicht einer combit_Large-Solution

Dim oContactRecordSet : Set oContactRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetRelati
onalRecordSet("ID.Kontakte.CompanyID")

If (oContactRecordSet Is Nothing) Then
    MsgBox "oContactRecordSet Is Nothing"
    WScript.Quit
End If

Call MsgBox(oContactRecordSet.RecCount & " - Datensätze befinden sich im
oContactRecordSet", vbOkOnly, cRM.AppTitle)

Set oContactRecordSet = Nothing
```

Beispiel C#-Script:

```
// Zählt alle Datensätze in einem RecordSet. Dieses Beispiel basiert auf der
Kontakte-Ansicht einer combit_Large-Solution

RecordSet contactRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.GetRelati
onalRecordSet("ID.Kontakte.CompanyID");

if (contactRecordSet == null)
{
    cRM.DialogMessageBox("contactRecordSet == null", "combit CRM", 16);
}
else
{
    cRM.DialogMessageBox(contactRecordSet.RecCount.ToString() + " - Datensätze
befinden sich im oContactRecordSet", "combit CRM", 0);
}
```

Lock

Beschreibung:

Sperrt den aktuellen Datensatz für die Bearbeitung durch andere Benutzer und sollte vor dem Ändern eines Datensatzes durch **SetContents**... aufgerufen werden. Die Methode liefert im Erfolgsfall True, ansonsten False zurück. Letzteres kann bspw. dadurch bedingt sein, dass der Datensatz bereits gesperrt oder die Bearbeitung aufgrund der aktuellen Berechtigungseinstellung nicht möglich ist.

Wichtig: Für ein mit der Methode **NewRecord** erzeugtes **Record** Objekt dürfen lediglich die Methoden **Lock**, **Get/SetContents...** mit einem (einmaligen) abschließenden **Save** und einem etwaigen (einmaligen) **Unlock** verwendet werden. Um andere Methoden des **Record** Objektes verwenden zu können, muss das **Record** Objekt freigegeben, neu initialisiert und auf den soeben erzeugten Datensatz positioniert werden.

Die Methode prüft Datensatzrechte und das Ansichtsrecht *Datensatz ändern*. Im Fehlerfall erhält man keine visuelle Meldung, d.h. im Script muss eine visuelle Benachrichtigung erfolgen, wenn die Methode fehlschlägt.

Parameter:

Parametername	Typ	Beschreibung
CheckForModifiedFields	Bool	Optional. Steuert, ob beim Sperren eines Datensatzes mit anschließendem Speichern einer Feldänderung überprüft werden soll, ob es Änderungen durch andere Benutzer gibt und das Speichern dann fehlschlagen soll (True) oder ob die Überprüfung deaktiviert wird und die letzte Datensatzänderung gewinnt (False). Dies ist insbesondere bei Feldänderungen über viele Datensätze innerhalb einer Schleife zu bevorzugen, da eine erhebliche Performance-Steigerung erreicht werden kann. Voreinstellung: False

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.Lock()
Call oRecord.SetContentsByName("Company", "Luna Aventuras")
Call oRecord.Save()
Call oRecord.Unlock()
Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.Lock();
record.SetContentsByName("Company", "Luna Aventuras");
record.Save();
record.Unlock();
cRM.CurrentProject.ActiveViews.ActiveView.Update();
record.Dispose();
```

PrintCard

Beschreibung:

Druckt ein Karteikartenprojekt aus. Neben dem Druckziel muss u.a. der Name des gewünschten Druckprojektes angegeben werden.

Wichtig: Diese Methode darf nicht aufgerufen werden, wenn das zugrundeliegende **Record** Objekt über **CurrentRecordBuffered** erzeugt wurde.

Parameter:

Parametername	Typ	Beschreibung
Media	String	Die zur Verfügung stehenden Ausgabe-Medien finden Sie im Kapitel **Ausgabe-Medium.** Über diesen Parameter werden zusätzliche Export-Optionen unterstützt. Weitere Informationen finden Sie im Kapitel **Export-Optionen für Print-Methoden**.

FileName	String	Dateiname inkl. Pfad des Druckprojektes.
Silent	Bool	Legt fest, ob der Druck (nach Möglichkeit) ohne Benutzerinteraktion erfolgen soll.
OutputFileName	String	Evtl. Name und Pfad der zu erzeugenden Ausgabe-Datei bei Ausgabe-Medien wie z. B. "RTF" oder "HTML". In allen anderen Fällen kann ein leerer String übergeben werden.
UseModalPreviewWindow	Bool	Optional. Legt fest, ob der Druck auf Vorschau (PRV) in das normale Vorschaufenster gedruckt wird (True, Voreinstellung) oder in die Berichtsansicht (False).

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.PrintCard("PRV", "%PRJDIR%\Druckvorlagen\Kontakte -
Serienbriefvorlage.crd", True, "", False)
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.PrintCard("PRV", @"%PRJDIR%\Druckvorlagen\Kontakte -
Serienbriefvorlage.crd", true, "", false);
record.Dispose();
```

PrintLabel

Beschreibung:

Druckt ein Etikettenprojekt aus. Neben dem Druckziel muss u.a. der Name des gewünschten Druckprojektes angegeben werden.

Wichtig: Diese Methode darf nicht aufgerufen werden, wenn das zugrundeliegende **Record** Objekt über **CurrentRecordBuffered** erzeugt wurde.

Parameter:

Parametername	Typ	Beschreibung
Media	String	Die zur Verfügung stehenden Ausgabe-Medien finden Sie im Kapitel **Ausgabe-Medium.** Über diesen Parameter werden zusätzliche Export-Optionen unterstützt. Weitere Informationen finden Sie im Kapitel **Export-Optionen für Print-Methoden**.
FileName	String	Dateiname inkl. Pfad des Druckprojektes.
Silent	Bool	Legt fest, ob der Druck (nach Möglichkeit) ohne Benutzerinteraktion erfolgen soll.
OutputFileName	String	Evtl. Name und Pfad der zu erzeugenden Ausgabe-Datei bei Ausgabe-Medien wie z. B. "RTF" oder "HTML". In allen anderen Fällen kann ein leerer String übergeben werden.
UseModalPreviewWindow	Bool	Optional. Legt fest, ob der Druck auf Vorschau (PRV) in das normale Vorschaufenster gedruckt wird (True, Voreinstellung) oder in die Berichtsansicht (False).

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.PrintLabel("PRV", "%PRJDIR%\Druckvorlagen\Kontakte -
Adressetikett.lbl", True, "", False)
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.PrintLabel("PRV", @"%PRJDIR%\Druckvorlagen\Kontakte - Adressetikett.lbl",
true, "", false);
record.Dispose();
```

PrintReport

Beschreibung:

Druckt ein Listenprojekt aus. Neben dem Druckziel muss u.a. der Name des gewünschten Druckprojektes angegeben werden.

Wichtig: Diese Methode darf nicht aufgerufen werden, wenn das zugrundeliegende **Record** Objekt über **CurrentRecordBuffered** erzeugt wurde.

Parameter:

Parametername	Typ	Beschreibung
Media	String	Die zur Verfügung stehenden Ausgabe-Medien finden Sie im Kapitel **Ausgabe-Medium.** Über diesen Parameter werden zusätzliche Export-Optionen unterstützt. Weitere Informationen finden Sie im Kapitel //.
FileName	String	Dateiname inkl. Pfad des Druckprojektes
Silent	Bool	Legt fest, ob der Druck (nach Möglichkeit) ohne Benutzerinteraktion erfolgen soll.
OutputFileName	String	Evtl. Name und Pfad der zu erzeugenden Ausgabe-Datei bei Ausgabe-Medien wie z. B. "RTF" oder "HTML". In allen anderen Fällen kann ein leerer String übergeben werden.
UseModalPreviewWindow	Bool	Optional. Legt fest, ob der Druck auf Vorschau (PRV) in das normale Vorschaufenster gedruckt wird (True, Voreinstellung) oder in die Berichtsansicht (False).

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.PrintReport("PRV", "%PRJDIR%\Druckvorlagen\Kontakte -
Auswertungen.lst", True, "", False)
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution
```

```
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.PrintReport("PRV", @"%PRJDIR%\Druckvorlagen\Kontakte - Auswertungen.lst",
true, "", false);
record.Dispose();
```

RemoveFromBlacklist

Beschreibung:

Entfernt den Inhalt des übergebenen Feldes aus der Sperrliste. Dabei wird die zugehörige Auto-Protokoll Aktion "Aus Sperrliste entfernen" ausgelöst.

Wichtig: Diese Methode nimmt nur Felder der internen Feldtypen 'E-Mail', 'Telefon', 'Mobiltelefon' und 'Telefax' entgegen.

Parameter:

Parametername	Typ	Beschreibung
Feldname	String	Feldname, dessen Inhalt aus der Sperrliste entfernt werden soll.

Rückgabewert:

Bool

Wert	Beschreibung
True	Inhalt wurde aus der Sperrliste entfernt.
False	Inhalt konnte nicht aus der Sperrliste entfernt werden, z. B. weil ein Feld mit einem nicht unterstützten internen Feldtypen übergeben wurde.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.RemoveFromBlacklist("Phone_Private")
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.RemoveFromBlacklist("Phone_Private");
record.Dispose();
```

Save

Beschreibung:

Speichert einen mit Hilfe von **SetContents**... geänderten Datensatz in die Datenbank; zuvor sollte der Datensatz immer mit **Lock** gesperrt werden.

Wichtig: Für ein mit der Methode **NewRecord** erzeugtes **Record** Objekt dürfen lediglich die Methoden **Lock**, **Get/SetContents...** mit einem (einmaligen) abschließenden **Save** und einem etwaigen (einmaligen) **Unlock** verwendet werden. Um andere Methoden des **Record** Objektes verwenden zu können, muss das **Record** Objekt freigegeben, neu initialisiert und auf den soeben erzeugten Datensatz positioniert werden.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.Lock()
Call oRecord.SetContentsByName("Company", "Luna Aventuras")
Call oRecord.Save()
Call oRecord.Unlock()
Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.Lock();
record.SetContentsByName("Company", "Luna Aventuras");
record.Save();
record.Unlock();
cRM.CurrentProject.ActiveViews.ActiveView.Update();
record.Dispose();
```

SaveRecordRef

Beschreibung:

Speichert einen Datensatzverweis.

Parameter:

Parametername	Typ	Beschreibung
FileName	String	Pfad + Dateiname + Dateiendung des zu speichernden Adressverweises.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.SaveRecordRef("C:\temp\RecordRef.crx")
Set oRecord = Nothing
```

Beispiel C#-Script:

```
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.SaveRecordRef(@"C:\temp\RecordRef.crx");
record.Dispose();
```

SendMailDirect

Beschreibung:

Sendet eine E-Mail mit Anhang entsprechend den Einstellungen unter "**Konfigurieren > Allgemein**" ohne Benutzer-Interaktion. Die automatische E-Mail-Ablage wird bei dieser Methode aktiviert. Die E-Mail wird, sofern konfiguriert, im Kontext des aktuellen Datensatz hinterlegt, bzw. verknüpft.

Wichtig: Diese Methode darf nicht aufgerufen werden, wenn das zugrundeliegende **Record** Objekt über **CurrentRecordBuffered** erzeugt wurde.

Parameter:

Parametername	Typ	Beschreibung

EmailAddress	String	E-Mail-Adresse(n) eines oder mehrerer Empfänger(s). Bei mehreren E-Mail-Adressen müssen diese durch Semikolon getrennt werden. Die Empfangsart pro E-Mail-Adresse kann über Präfixe bestimmt werden. Wenn kein Präfix angegeben ist, wird "TO:" angenommen, dieser kann aber optional auch angegeben werden. Für den Versand von E-Mail-Kopien können zusätzlich die Präfixe "CC:" und/oder "BCC:" verwendet werden; beachten Sie hierbei, dass jede E-Mail-Adresse einen eigenen Präfix benötigt. Beispiel 1: TO:maier@combit.net;CC:mueller@combit.net; CC:schmidt@combit.net;BCC:fischer@combit.net; Beispiel 2: maier@combit.net;weber@combit.net;
Subject	String	Betreff der E-Mail.
Contents	String	Text der E-Mail.
Files	String	Kann eine Liste von E-Mail-Anhängen beinhalten. Diese müssen dann per Semikolon separiert übergeben werden. z. B. C:\MyFiles\Report.pdf;C:\Info.doc

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Dim sMail : sMail = oRecord.GetContentsByName("Email")
Dim sSubject : sSubject = "Mailing: Aktuelle Angebote"
Dim sContents : sContents = "Sehr geehrte Damen und Herren, ..."
Dim sFiles : sFiles = "C:\temp\Angebot.pdf;C:\temp\Angebot2.pdf"
Call oRecord.SendMailDirect(sMail, sSubject, sContents, sFiles)
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
string mail = record.GetContentsByName("Email");
string subject = "Mailing: Aktuelle Angebote";
string contents = "Sehr geehrte Damen und Herren, ...";
string files = @"C:\temp\Angebot.pdf;C:\temp\Angebot2.pdf";
record.SendMailDirect(mail, subject, contents, files);
record.Dispose();
```

SendRecordRef

Beschreibung:

Sendet eine E-Mail mit dem Datensatzverweis des aktuellen Datensatzes.

Wichtig: Diese Methode darf nicht aufgerufen werden, wenn das zugrundeliegende **Record** Objekt über **CurrentRecordBuffered** erzeugt wurde.

Parameter:

Parametername	Typ	Beschreibung
EmailAddress	String	Empfänger E-Mail-Adresse.
Subject	String	Betreff der E-Mail.
Contents	String	Text der E-Mail.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.SendRecordRef("soleil@luna-aventuras.net", "Datensatzverweis", "Hallo
Jean, anbei der gewnschte Datensatzverweis.")
Set oRecord = Nothing
```

Beispiel C#-Script:

```
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.SendRecordRef("soleil@luna-aventuras.net", "Datensatzverweis", "Hallo Jean,
anbei der gewnschte Datensatzverweis.");
record.Dispose();
```

SendSingleMail

Beschreibung:

Sendet eine Einzelmail ohne Dialog. Es wird der vollständige Pfad der Vorlagedatei übergeben.

Wichtig: Diese Methode darf nicht aufgerufen werden, wenn das zugrundeliegende **Record** Objekt über **CurrentRecordBuffered** erzeugt wurde.

Parameter:

Parametername	Typ	Beschreibung
TemplatePath	String	Vollständiger Pfad der Vorlagedatei.
Files	String	Optional. Kann eine Liste von E-Mail-Anhängen beinhalten. Diese müssen dann per Semikolon separiert übergeben werden. z. B. C:\MyFiles\Report.pdf;C:\Info.doc Beachten Sie bitte, dass die übergebenen Anhänge stets zusätzlich zu den evtl. bereits in einer über den Parameter "TemplatePath" definierten Mailvorlage hinterlegten Anhängen versendet werden. Dies gilt auch dann, wenn ein übergebener E-Mail-Anhang denselben Pfad hat wie in der Mailvorlage.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Dim sTemplatePath : sTemplatePath = "%PRJDIR%\Mailvorlagen\Kontakte - HTML-Format
Vorlage.mtpx"
Dim sFiles : sFiles = "C:\temp\Angebot.pdf;C:\temp\Angebot2.pdf"
Call oRecord.SendSingleMail(sTemplatePath, sFiles)
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
string templatePath = @"%PRJDIR%\Mailvorlagen\Kontakte - HTML-Format
Vorlage.mtpx";
string files = @"C:\temp\Angebot.pdf;C:\temp\Angebot2.pdf";
record.SendSingleMail(templatePath, files);
record.Dispose();
```

SendSingleMailDialog

Beschreibung:

Sendet eine Einzelmail mit vorherigem Dialog. Es wird der vollständige Pfad der Vorlagedatei übergeben. Bevor der Anwender die Mail verschickt, kann er noch Änderungen/Ergänzungen vornehmen, die danach aber nicht zwingend in der Vorlage abgespeichert werden müssen.

Wichtig: Diese Methode darf nicht aufgerufen werden, wenn das zugrundeliegende **Record** Objekt über **CurrentRecordBuffered** erzeugt wurde.

Parameter:

Parametername	Typ	Beschreibung
TemplatePath	String	Vollständiger Pfad der Vorlagedatei.
Files	String	Optional. Kann eine Liste von E-Mail-Anhängen beinhalten. Diese müssen dann per Semikolon separiert übergeben werden. z. B. C:\MyFiles\Report.pdf;C:\Info.doc Beachten Sie bitte, dass die übergebenen Anhänge stets zusätzlich zu den evtl. bereits in einer über den Parameter "TemplatePath" definierten Mailvorlage hinterlegten Anhängen versendet werden. Dies gilt auch dann, wenn ein übergebener E-Mail-Anhang denselben Pfad hat wie in der Mailvorlage.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Dim sTemplatePath : sTemplatePath = "%PRJDIR%\Mailvorlagen\Kontakte - HTML-Format
Vorlage.mtpx"
Dim sFiles : sFiles = "C:\temp\Angebot.pdf;C:\temp\Angebot2.pdf"
Call oRecord.SendSingleMailDialog(sTemplatePath, sFiles)
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
string templatePath = @"%PRJDIR%\Mailvorlagen\Kontakte - HTML-Format
Vorlage.mtpx";
string files = @"C:\temp\Angebot.pdf;C:\temp\Angebot2.pdf";
record.SendSingleMailDialog(templatePath, files);
record.Dispose();
```

SetContentsByFormula

Beschreibung:

Legt den Inhalt des Feldes mit einer Formel fest, dessen Feldname übergeben wurde. Das Setzen eines Primärschlüssels ist möglich, sofern das Schreiben erlaubt ist.

Hinweis: Bitte prüfen Sie den Rückgabewert der Methode, um sicherzustellen, dass das Setzen des neuen Inhalts funktioniert hat.

Wird ein formatiertes Notizenfeld gesetzt, ist folgendes zu beachten: fängt der Inhalt mit \\plaintext: an, dann wird Klartext angenommen, fängt er mit \\html: an, dann wird HTML Inhalt angenommen, wird kein Präfix übergeben, so wird HTML angenommen.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes.
Formula	String	(Neuer) Feldinhalt aus einer Formel.

Hinweis: Für Feldnamen, die in einer Formel verwendet werden, muss der physikalische Feldname verwendet werden. Das Verwenden von Feldaliasen wird ausdrücklich nicht empfohlen, da eine Änderung des Feldalias (ggf. nur indirekt, z. B. durch die Änderung eines 1:1 Relationsalias) dazu führt, dass das Script nicht mehr korrekt funktioniert.

Tipp: Eine funktionierende Formel lässt sich bspw. in den Ansichteneigenschaften der betreffenden Ansicht im Reiter "Datensatzverweis" in einem der Formel-Dialoge zusammenstellen. Im Reiter "Felder" können Sie im Zweifel prüfen, welcher Feldalias zu welchem phys. Feldnamen gehört. Kopieren Sie anschließend die zusammengestellte Formel und verwenden diese im Script.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.Lock()
Call oRecord.SetContentsByFormula("DateTime", "Date$(Now(),""%02d.%02m.%04y
%02H:%02i:%02s"")")
Call oRecord.Save()
Call oRecord.Unlock()
Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
Set oRecord = Nothing
```

Beispiel C#-Script:

```
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.Lock();
record.SetContentsByFormula("DateTime", @"Date$(Now(),""%02d.%02m.%04y
%02H:%02i:%02s"")");
record.Save();
record.Unlock();
cRM.CurrentProject.ActiveViews.ActiveView.Update();
record.Dispose();
```

SetContentsByName

Beschreibung:

Legt den Inhalt des Feldes fest, dessen physikalischer Feldname übergeben wurde. Das Setzen eines Primärschlüssels ist möglich, sofern das Schreiben erlaubt ist.

Hinweis: Bitte prüfen Sie den Rückgabewert der Methode, um sicherzustellen, dass das Setzen des neuen Inhalts funktioniert hat.

Wird ein formatiertes Notizenfeld gesetzt, ist folgendes zu beachten: fängt der Inhalt mit \\plaintext: an, dann wird Klartext angenommen, fängt er mit \\html: an, dann wird HTML Inhalt angenommen, wird kein Präfix übergeben, so wird HTML angenommen.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes.
Contents	String	(Neuer) Feldinhalt

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.Lock()
Call oRecord.SetContentsByName("Company", "Luna Aventuras")
Call oRecord.Save()
Call oRecord.Unlock()
Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.Lock();
record.SetContentsByName("Company", "Luna Aventuras");
record.Save();
record.Unlock();
cRM.CurrentProject.ActiveViews.ActiveView.Update();
record.Dispose();
```

SetContentsByNameFromFile

Beschreibung:

Die angegebene Datei wird in das übergebene BLOB-Feld eingebettet.

Hinweis: Damit die Datei in der Anwendung korrekt angezeigt bzw. verwendet werden kann, muss zusätzlich die Dateiendung, sowie der Dateiname in die entsprechenden Felder (siehe Konfiguration der 'Dokumentinfos' der Ansicht) geschrieben werden.

Es ist nötig darauf zu achten, dass die einzubettende Datei bis zum Abschluss des Speichervorgangs unter dem im FileName-Parameter angegeben Pfad zu finden ist.

Parameter:

Parametername	Typ	Beschreibung
FieldName	String	Physikalischer Name des gewünschten (BLOB)-Felds.
FileName	String	Dateiname

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Aktivitäten-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.Lock()
Call oRecord.SetContentsByNameFromFile("Document_Embedded", "C:\temp\Excel
Arbeitsblatt.xlsx")
Call oRecord.SetContentsByName("Document_FileName", "Excel Arbeitsblatt")
Call oRecord.SetContentsByName("Document_FileType", "xlsx")
Call oRecord.Save()
Call oRecord.Unlock()
Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Aktivitäten-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.Lock();
record.SetContentsByNameFromFile("Document_Embedded", @"C:\temp\Excel
Arbeitsblatt.xlsx");
record.SetContentsByName("Document_FileName", "Excel Arbeitsblatt");
record.SetContentsByName("Document_FileType", "xlsx");
record.Save();
record.Unlock();
cRM.CurrentProject.ActiveViews.ActiveView.Update();
record.Dispose();
```

SetContentsByNameToNull

Beschreibung:

Setzt Feld auf NULL. Bitte beachten Sie, dass die Methode nicht prüft, ob ein Feld in der Datenbank überhaupt NULL sein darf!

Hinweis: Die Methode liefert beim Setzen eines Primärschlüssels False zurück.

Parameter:

Parametername	Typ	Beschreibung
FieldName	String	Physikalischer Name des Feldes.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.Lock()
Call oRecord.SetContentsByNameToNull("ModifiedBy")
Call oRecord.Save()
```

```
Call oRecord.Unlock()
Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
Set oRecord = Nothing
```

Beispiel C#-Script:

```
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.Lock();
record.SetContentsByNameToNull("ModifiedBy");
record.Save();
record.Unlock();
cRM.CurrentProject.ActiveViews.ActiveView.Update();
record.Dispose();
```

SetContentsValueByFormula

Beschreibung:

Legt den Inhalt des Feldes mit einer Formel fest, dessen Feldname übergeben wurde. Das Setzen eines Primärschlüssels ist möglich, sofern das Schreiben erlaubt ist.

Hinweis: Bitte prüfen Sie den Rückgabewert der Methode, um sicherzustellen, dass das Setzen des neuen Inhalts funktioniert hat.

Wird ein formatiertes Notizenfeld gesetzt, ist folgendes zu beachten: fängt der Inhalt mit \\plaintext: an, dann wird Klartext angenommen, fängt er mit \\html: an, dann wird HTML Inhalt angenommen, wird kein Präfix übergeben, so wird HTML angenommen.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes.
Formula	String	(Neuer) Feldinhalt aus einer Formel.

Hinweis: Für Feldnamen, die in einer Formel verwendet werden, muss der physikalische Feldname verwendet werden. Das Verwenden von Feldaliasen wird ausdrücklich nicht empfohlen, da eine Änderung des Feldalias (ggf. nur indirekt, z. B. durch die Änderung eines 1:1 Relationsalias) dazu führt, dass das Script nicht mehr korrekt funktioniert.

Tipp: Eine funktionierende Formel lässt sich bspw. in den Ansichteneigenschaften der betreffenden Ansicht im Reiter "Datensatzverweis" in einem der Formel-Dialoge zusammenstellen. Im Reiter "Felder" können Sie im Zweifel prüfen, welcher Feldalias zu welchem phys. Feldnamen gehört. Kopieren Sie anschließend die zusammengestellte Formel und verwenden diese im Script.

Rückgabewert:

Bool

Hinweis: Bitte beachten Sie, dass "" nicht gleich NULL ist.

Beispiel VBScript:

```
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.Lock()
Call oRecord.SetContentsValueByFormula("DateTime", "Date$(Now(),""%02d.%02m.%04y
%02H:%02i:%02s"")")
Call oRecord.Save()
Call oRecord.Unlock()
Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
```

```
Set oRecord = Nothing
```

Beispiel C#-Script:

```
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.Lock();
record.SetContentsValueByFormula("DateTime", @"Date$(Now(),""%02d.%02m.%04y
%02H:%02i:%02s"")");
record.Save();
record.Unlock();
cRM.CurrentProject.ActiveViews.ActiveView.Update();
record.Dispose();
```

SetContentsValueByName

Beschreibung:

Legt den Inhalt des Feldes fest, dessen physikalischer Feldname übergeben wurde.

Die übergebene Variable für den Inhalt kann dabei einen zum Feldtyp korrespondierenden Typ haben und muss nicht vorher in eine Zeichenkette umgewandelt werden. Somit werden bspw. Lokalisierungsprobleme (Komma oder Punkt als Dezimalzeichen? Datumsformatierung?) bei der Weiterverarbeitung des Wertes vermieden.

Das Setzen eines Primärschlüssels ist möglich, sofern das Schreiben erlaubt ist.

Hinweis: Bitte prüfen Sie den Rückgabewert der Methode, um sicherzustellen, dass das Setzen des neuen Inhalts funktioniert hat.

Wird ein formatiertes Notizenfeld gesetzt, ist folgendes zu beachten: fängt der Inhalt mit \\plaintext: an, dann wird Klartext angenommen, fängt er mit \\html: an, dann wird HTML Inhalt angenommen, wird kein Präfix übergeben, so wird HTML angenommen.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Name des gewünschten Feldes.
Contents	Variant	(Neuer) Feldinhalt

Rückgabewert:

Bool

Hinweis: Bitte beachten Sie, dass "" nicht gleich NULL ist.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.Lock()
Call oRecord.SetContentsValueByName("Company", "Luna Aventuras")
Call oRecord.Save()
Call oRecord.Unlock()
Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.Lock();
```

```
record.SetContentsValueByName("Company", "Luna Aventuras");
record.Save();
record.Unlock();
cRM.CurrentProject.ActiveViews.ActiveView.Update();
record.Dispose();
```

TransferData

Beschreibung:

Führt die Einzelübernahme mit einer als Parameter übergebenen Übernahmemaske durch.

Hinweis: Falls es sich um einen nicht-visuellen Record handelt, wird das Autoprotokoll ("Einzelübernahme") nicht ausgelöst.

Parameter:

Parametername	Typ	Beschreibung
TransferTemplate	String	Pfad und Dateiname der Übernahmemaske. Die Platzhalter %APPDIR% und %PRJDIR% werden unterstützt.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.TransferD
ata("%PRJDIR%\Übernahmemasken\Kontakte - Microsoft Word.xfx")
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord.TransferD
ata(@"%PRJDIR%Übernahmemasken\Kontakte - Microsoft Word.xfx");
```

Unlock

Beschreibung:

Entsperrt einen zuvor mit **Lock** gesperrten Datensatz wieder und ermöglicht somit die Änderung durch andere Benutzer.

Wichtig: Für ein mit der Methode **NewRecord** erzeugtes **Record** Objekt dürfen lediglich die Methoden **Lock**, **Get/SetContents...** mit einem (einmaligen) abschließenden **Save** und einem etwaigen (einmaligen) **Unlock** verwendet werden. Um andere Methoden des **Record** Objektes verwenden zu können, muss das **Record** Objekt freigegeben, neu initialisiert und auf den soeben erzeugten Datensatz positioniert werden.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
Call oRecord.Lock()
Call oRecord.SetContentsByName("Company", "Luna Aventuras")
Call oRecord.Save()
```

```
Call oRecord.Unlock()
Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
record.Lock();
record.SetContentsByName("Company", "Luna Aventuras");
record.Save();
record.Unlock();
cRM.CurrentProject.ActiveViews.ActiveView.Update();
record.Dispose();
```

3.38 RecordSet Objekt

Wichtig: Bitte beachten Sie, dass wenn das RecordSet-Objekt basierend auf einem ViewConfig-Objekt erzeugt wurde, sämtliche Methodenaufrufe (wie z. B. Druck oder E-Mail-Versand) keinerlei Autoprotokolle, die automatische Mail-Ablage o.ä. auslösen!

3.38.1 Eigenschaften

HasMultipleRecords, read-only

Beschreibung:

Gibt **True** zurück, wenn das **RecordSet** mehr als einen Datensatz beinhaltet. Diese Methode ist deutlich effizienter als die Abfrage von **RecCount**, insofern bei einem Filter genau ein Treffer erwartet wird, aber auch der Fall mit mehreren Treffern sauber behandelt werden soll.

Wichtig: Wird diese Eigenschaft direkt nach einer Filter-Methode aufgerufen, steht das **RecordSet** anschließend bereits auf dem ersten Datensatz. Wenn das Vorhandensein von Treffern anschließend über **RecordSet.MoveNext** geprüft wird, würde der erste Treffer dadurch übersprungen werden. Verwenden Sie daher die Eigenschaft besser erst, wenn Sie schon das Vorhandensein von Treffern abgeprüft haben. Das Überprüfen auf Treffer mittels **RecordSet.MoveFirst** würde immer funktionieren – es führt jedoch zu einer doppelten Datenbank-Operation.

Typ:

Bool

Beispiel VBScript:

```
If (oRecordSet.SetFilter...()) Then
    If (oRecordSet.MoveFirst()) Then
       If (oRecordSet.HasMultipleRecords = True) Then
           ' ... mehrere Treffer
       Else
           ' ... ein Treffer
       End If
    Else
       ' ... keine Treffer
    End If
Else
    ' ... Fehler
End If
```

Beispiel C#-Script:

```
RecordSet recordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet();
```

```
if (recordSet.SetFilterByName("ABC Kunden") == true)
{
    if (recordSet.MoveFirst() == true)
    {
        if (recordSet.HasMultipleRecords == true)
        {
            // ... mehrere Treffer
        }
        else
        {
            // ... ein Treffer
        }
    }
    else
    {
        // ... keine Treffer
    }
}
else
{
    // ... Fehler
}
```

RecCount, read-only

Beschreibung:

Liefert die Anzahl der Datensätze im aktuellen **RecordSet** Objekt zurück.

Sollte das **RecordSet** Objekt zuvor gefiltert worden sein, dann wird die Anzahl der im Filter befindlichen Datensätze zurückgeliefert. Ohne aktiven Filter erhält man die Gesamtzahl der Datensätze.

Typ:

Long

Beispiel VBScript:

```
Dim nRecCount : nRecCount = cRM.CurrentProject.ActiveViews.ActiveView.RecCount
Call cRM.DialogMessageBox("Es befinden sich derzeit " & CStr(nRecCount) & "
Datensätze in der aktuellen Ansicht.", "RecordSet.RecCount", vbOkOnly)
```

Beispiel C#-Script:

```
long recCount = cRM.CurrentProject.ActiveViews.ActiveView.RecCount;
cRM.DialogMessageBox("Es befinden sich derzeit " + recCount.ToString() + "
Datensätze in der aktuellen Ansicht.", "RecordSet.RecCount", 0);
```

SortOrder

Beschreibung:

Setzt eine definierte Sortierung über den Index oder gibt die aktuell gesetzte Sortierung zurück.

Typ:

Long

Wert	Beschreibung
0	<ohne Sortierung>
Variabel	Index der Sortierung Eine absteigende Sortierung kann durch Setzen des entsprechenden negativen Werts erreicht werden.

Beispiel VBScript:

```
Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet
If (oRecordSet.SortOrder <> 3) Then
    oRecordSet.SortOrder = 3
```

```
End If
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
RecordSet recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet;
if (recordSet.SortOrder != 3)
{
    recordSet.SortOrder = 3;
}
recordSet.Dispose();
```

ViewName, read-only

Beschreibung:

Gibt den Namen des übergeordneten **View** Objektes zurück.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Der Name der übergeordneten Ansicht lautet: " &
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.ViewName,
"RecordSet.ViewName", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Der Name der übergeordneten Ansicht lautet " +
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.ViewName,
"RecordSet.ViewName", 0);
```

3.38.2 Methoden

CalcAggregationValues

Beschreibung:

Erlaubt die Berechnung von Aggregationsergebnissen basierend auf dem aktuellen Filter.

Parameter:

Parametername	Typ	Beschreibung
AggregationColumnsSQLExpression	String	Übergabe des SQL-Ausdrucks. Ergebnis sollten Spalten sein, die beispielsweise auch bei der Nutzung von SELECT den Rückgabewert darstellen. Etwaige Spaltendatentypen bleiben, soweit möglich, erhalten. Unbenannte Spaltennamen ohne Alias erhalten die Namen $Field1 bis $FieldN. Sollte versucht werden unerlaubte SQL-Ausdrücke zu verwenden (DROP, FROM, GO, …), wird ein Scriptfehler zurückgegeben. Details können mit dem Debug-Tool Debwin in Erfahrung gebracht werden. Datensatzrechte werden berücksichtigt.

Wichtig: Beachten Sie bitte, dass sofern für eine Spalte ein Alias vergeben wurde, dieser Alias für den späteren Zugriff auch verwendet werden muss. Der Zugriff mittels "$FieldN" ist in diesem Fall nicht möglich.

Rückgabewert:

DataItem

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oDataItem : Set oDataItem =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CalcAggregationValues("
count(*), min(""Birthday"") as Oldest, max(""Birthday"") as Youngest")

Call cRM.DialogMessageBox("Anzahl Datensätze: " &
oDataItem.GetContentsValueByName("$Field1") & vbCrLf & "Geburtsjahr Ältester: " &
CStr(Year(oDataItem.GetContentsValueByName("Oldest"))) & vbCrLf & "Geburtstag
Jüngster: " & oDataItem.GetContentsValueByName("Youngest"),
"RecordSet.CalcAggregationValues", vbOkOnly)

Set oDataItem = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

DataItem item =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CalcAggregationValues("
count(*), min(\"Birthday\") as Oldest, max(\"Birthday\") as Youngest");

cRM.DialogMessageBox("Anzahl Datensätze: " +
item.GetContentsValueByName("$Field1") + System.Environment.NewLine + "Geburtsjahr
Ältester: " + item.GetContentsValueByName("Oldest").ToString() +
System.Environment.NewLine + "Geburtstag Jüngster: " +
item.GetContentsValueByName("Youngest"), "RecordSet.CalcAggregationValues", 0);

item.Dispose();
```

CreateCopy

Beschreibung:

Erstellt eine Kopie des aktuellen **RecordSet** Objekts, um dieses beispielsweise per Filter weiter einzugrenzen, ohne dafür das zugrundeliegende **RecordSet** Objekt verändern zu müssen.

Hinweis: Wir empfehlen, nach Erzeugung eines **RecordSet**-Objektes zunächst mittles Aufruf der Methode "MoveFirst" die Existenz mindestens eines **Record**-Objektes zu überprüfen.

Wichtiger Hinweis für die Verwendung des Parameters CursorModel mit dem Wert 2 (forward-only) unter Microsoft SQL Server: Die Datensätze eines forward-only-RecordSets müssen nach dessen Erstellung direkt und unmittelbar über eine "GotoNext"-Schleife ohne Interaktion vollständig durchlaufen werden. Anderenfalls kann es, wenn das RecordSet viele Zeilen enthält, am Datenbankserver zu einem *ASYNC_NETWORK_IO*-Wartezustand kommen, der dann andere Abfragen (vor allem Änderungen) auf dieselbe Tabelle blockiert.

Parameter:

Parametername	Typ	Beschreibung
CursorModel	Long	Optional. Ermöglicht die Spezifikation des Datenbankcursormodells, das für den zurückgegebenen **RecordSet** genutzt werden soll. Werte: 0 (Standardwert): Erzeugt ein **RecordSet** mit einem Datenbankcursormodell, welches innerhalb der combit CRM-Projektdatei spezifiziert werden kann:

		... <!-- DATA --> <profile> <list name="">> <list name="ExtendedSettings">> <item name="COMRecordSetCursorDefault">2</item> </list> ... Wird in der combit CRM-Projektdatei keine Eigenschaft **COMRecordSetCursorDefault** gefunden, so wird immer ein fully-dynamic **RecordSet** erzeugt. Mögliche Werte für die Eigenschaft sind: 1 – fully-dynamic RecordSet, 2 – forward-only RecordSet. 1: Erzeugt ein **RecordSet** mit fully-dynamic Datenbankcursor. 2: Erzeugt ein **RecordSet** mit forward-only Datenbankcursor. Ermöglicht deutliche Performance-Gewinne, insbesondere bei großen Datenmengen und komplexen Filterausdrücken, erlaubt aber lediglich das einmalige Durchlaufen in Vorwärtsrichtung durch den **RecordSet**. Die Methoden **RecordSet.DialogSelectRecord**, **RecordSet.DialogSelectRecordMultiple**, **RecordSet.SendBulkMail** (bei anzuzeigendem integrierten Mail-Editor), **RecordSet.MovePrevious**, **RecordSet.MoveLast**, **InputForm.DialogSelectRecordDropDown** werden einen Scriptfehler werfen, wenn diese für einen forward-only **RecordSet** genutzt werden. Für diese Methoden muss der **RecordSet** explizit ohne forward-only (Werte 0 oder 1) erzeugt werden.

Rückgabewert:

RecordSet

Beispiel VBScript:

```
Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CreateCopy()
' ...
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
RecordSet recordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CreateCopy();
// ...
recordSet.Dispose();
```

CurrentRecord

Beschreibung:

Liefert den aktuellen Datensatz als Objekt vom Typ **Record** zurück. Wenn z. B. keine Datensätze im **RecordSet** enthalten sind, liefert diese Methode kein **Record** Objekt zurück.

Wichtig: Im Gegensatz zur Methode **CurrentRecordBuffered** aktualisiert sich ein so erzeugter Record immer automatisch, wenn anschließend **Move**...-Befehle für den zugehörigen **RecordSet** aufgerufen werden.

Da das komplette Puffern der Feldinhalte entfällt, ist diese Methode deutlich performanter als **CurrentRecordBuffered** und stellt den empfohlenen Weg dar, um auf Feldinhalte von Datensätzen zuzugreifen.

Mit **CurrentRecord** erzeugte **Record**-Objekte können dadurch jedoch nicht als Variablen für unterschiedliche Datensätze benutzt werden (vgl. Beispiel zu **CurrentRecordBuffered**)! Ist dies erforderlich, so muss anstattdessen **CurrentRecordBuffered** benutzt werden.

Wichtiger Hinweis zu geändertem Verhalten: CurrentRecord verhält sich ab der Version 8 von combit CRM wie ein **CurrentRecordSynchronized**. (Aus Kompatibilitätsgründen vorher wie ein **CurrentRecordBuffered**). Wir empfehlen die eingesetzten Scripte sorgfältig auf die Verwendung von **CurrentRecord** zur prüfen. Ein typisches Beispiel für die Verwendung ist ein **Record**-Objekt, welches per **CurrentRecord** geholt wird und danach im zugehörigen **RecordSet** eine **Goto**.../**SetFilter**... Methode aufgerufen wird und danach trotzdem auf den zuvor geholten (d.h. gewollt absichtlich "alten") **Record** zugegriffen wird:

```
RecordSet.GotoFirst

Set oRecord = RecordSet.CurrentRecord

oRecord.GetContentsByName "Company" 'liefert Firmenname von Firmendatensatz #1

RecordSet.GotoNext

oRecord.GetContentsByName "Company" 'cRM7: Firmenname (weiterhin) von Firmendatensatz #1 / cRM8: Firmenname von Firmendatensatz #2
```

Wenn das cRM7 Verhalten der letzten Code-Zeile oben ZWINGEND erforderlich sein sollte, dann schafft Abhilfe entweder die Verwendung eines explizit gepufferten Records, d.h.

```
Set oRecord = RecordSet.CurrentRecordBuffered
```

oder das Setzen eines Kompatibilitätsschalters in der Solution Projekt-Datei (wirkt sich global auf alle **CurrentRecord** Stellen innerhalb der Solution aus). Erstellen Sie dazu in der Solution Projekt-Datei eine Untersektion namens "ExtendedSettings" und setzen darin die Option "COMCurrentRecordAlwaysAsBuffered" auf 1:

```
 ...

<!–DATA –->

<profile>

<list name="">

  <list name="ExtendedSettings">

   <item name="COMCurrentRecordAlwaysAsBuffered">1</item>

  </list>

...
```

Beachten Sie dabei aber, dass die letztere Variante nur als Notlösung eingesetzt wird, da die problematische Codestelle z. B. in einem verschlüsselten Script liegt und daher nicht selbst angepasst werden kann. Wir empfehlen grundsätzlich möglichst ohne gepufferte Records auszukommen!

Rückgabewert:

Record

Beispiel VBScript:

```
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord
' ...
Set oRecord = Nothing
```

Beispiel C#-Script:

```
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord;
// ...
record.Dispose();
```

CurrentRecordBuffered

Beschreibung:

Liefert den aktuellen Datensatz als Objekt vom Typ **Record** zurück. Diese Methode funktioniert nur, wenn zuvor über eine der **Move**...-Methoden auf einen gültigen Datensatz gesprungen wurde. Wenn z. B. keine Datensätze im **RecordSet** enthalten sind, liefert diese Methode kein **Record** Objekt zurück.

Wichtig: Werden anschließend **Move**...-Befehle für den zugehörigen **RecordSet** aufgerufen, so aktualisiert sich das **Record**-Objekt nicht. Es behält die zum Zeitpunkt seiner Erzeugung aktuellen Feldinhalte. Um die neuen Inhalte zu bekommen, muss ein neues Objekt von **RecordSet** durch erneuten Aufruf von **CurrentRecordBuffered** erzeugt werden.

Diese Methode ist deutlich weniger performant als **CurrentRecord** und sollte nur verwendet werden, wenn unterschiedliche **Record**-Objekte als Variablen für unterschiedliche Datensätze benutzt werden müssen (vgl. Beispiel).

Rückgabewert:

Record

Beispiel VBScript:

```
' Vergleicht zwei Umsätze von Kontakte-Datensätzen. Dieses Beispiel basiert auf
der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet
Dim oRecord1 : Set oRecord1 = oRecordSet.CurrentRecord
Dim oRecord2

If (oRecordSet.MoveNext() = True) Then
    Set oRecord2 = oRecordSet.CurrentRecordBuffered

    If (CDbl(oRecord1.GetContentsByName("Turnover") >
CDbl(oRecord2.GetContentsByName("Turnover"))) Then
        Call cRM.DialogMessageBox("Der Umsatz des ersten Datensatz ist höher als
der Umsatz des zweiten Datensatz.", "RecordSet.CurrentRecordBuffered", vbOkOnly)
    End If

    Set oRecord2 = Nothing
End If

Set oRecord1 = Nothing
```

Beispiel C#-Script:

```
// Vergleicht zwei Umsätze von Kontakte-Datensätzen. Dieses Beispiel basiert auf
der Kontakte-Ansicht einer combit_Large-Solution

RecordSet recordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet();
Record record = recordSet.CurrentRecord;
Record recordBuffered;
double.TryParse(record.GetContentsByName("Turnover"), out double
valueCurrentRecordParsed);

if (recordSet.MoveNext() == true)
{
    recordBuffered = recordSet.CurrentRecordBuffered;
    double.TryParse(recordBuffered.GetContentsByName("Turnover"), out double
valueBufferedRecordParsed);

    if (valueCurrentRecordParsed > valueBufferedRecordParsed)
    {
        cRM.DialogMessageBox("Der Umsatz des ersten Datensatz ist höher als der
Umsatz des zweiten Datensatz.", "RecordSet.CurrentRecordBuffered", 0);
    }

    recordBuffered.Dispose();
}

record.Dispose();
```

CurrentRecordSynchronized

Beschreibung:

Liefert den aktuellen Datensatz als Objekt vom Typ **Record** zurück. Wenn z. B. keine Datensätze im **RecordSet** enthalten sind, liefert diese Methode kein **Record** Objekt zurück. Diese Methode ist aus Kompatibilitätsgründen enthalten, siehe die Methode **CurrentRecord** für den empfohlenen Weg.

Wichtig: Im Gegensatz zur Methode **CurrentRecordBuffered** aktualisiert sich ein so erzeugter Record immer automatisch, wenn anschließend **Move**...-Befehle für den zugehörigen **RecordSet** aufgerufen werden.

Da das komplette Puffern der Feldinhalte entfällt, ist diese Methode deutlich performanter als **CurrentRecordBuffered** und stellt den empfohlenen Weg dar, um auf Feldinhalte von Datensätzen zuzugreifen.

Mit **CurrentRecordSynchronized** erzeugte **Record**-Objekte können dadurch jedoch nicht als Variablen für unterschiedliche Datensätze benutzt werden (vgl. Beispiel zu **CurrentRecordBuffered**)! Ist dies erforderlich, so muss anstattdessen **CurrentRecordBuffered** benutzt werden.

Rückgabewert:

Record

Beispiel VBScript:

```
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecordSynchroniz
ed
' ...
Set oRecord = Nothing
```

Beispiel C#-Script:

```
Record record =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecordSynchroniz
ed;
// ...
record.Dispose();
```

DeleteAllRecords

Beschreibung:

Löscht alle Datensätze im Filter des aktuellen **RecordSet**-Objektes.

Hinweis: Beim Löschen mehrerer Datensätze über diese Methode wird das Papierkorb-Feature berücksicht, insofern dieses aktiviert wurde.

Rückgabewert:

Bool

Wert	Beschreibung
True	Alle Datensätze wurden gelöscht.
False	Befehl konnte nicht ausgeführt werden.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet
Call oRecordSet.DeleteAllRecords()
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

RecordSet recordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet();
recordSet.DeleteAllRecords();
recordSet.Dispose();
```

DialogEditFormula

Beschreibung:

Stellt den Formel-Assistenten basierend auf den Feldern des aktuellen **RecordSet** dar.

Parameter:

Parametername	Typ	Beschreibung
sTitle	String	Titel des Dialogs inklusive des Anhangs " bearbeiten". Wenn eine leere Zeichenkette übergeben wird, wird "Formel bearbeiten" verwendet.
sFormula	String	Zu bearbeitende Formel.
bUseRealData	Bool	Optional. Standardwert: False True: Es wird in der Formelergebnis-Vorschau mit den aktuellen Werten des aktuellen Datensatzes gearbeitet. False: Es werden keine Echtdaten verwendet. Wichtig: Bei Verwendung der Vorschau anhand aktueller Werte des Datensatzes werden alle Felder der Ansicht, sowie alle Felder der 1:1-Relationen vom Datenbankserver angefordert. Dies kann je nach Komplexität der 1:1-Relationen zu einer erheblichen Geschwindigkeitsverzögerung führen.
nAllowedResultType	Hexadezimal	Optional. Standardwert: Text + Numerisch &H10000000 = Text, &H08000000 = Numerisch,

		&H04000000 = Datum, &H02000000 = Boolean Diese Werte können beliebig miteinander kombiniert („verODERt“) werden.
nParentWindow	Long	Optional. Standardwert: Aktuell aktives cRM-Fenster. Ermöglicht die Übergabe eines Handles für das Elternfenster.

Rückgabewert:

String

Wert	Beschreibung
Formel	Die Formel konnte erfolgreich erstellt werden und befindet sich nun im Rückgabewert der Methode.
$CANCEL$	Der Formel-Assistent wurde vom Benutzer geschlossen bzw. die Eingabe abgebrochen.
$ERROR$\t<Fehlertext>	Es ist ein Fehler aufgetreten. Genauere Informationen finden sich TAB-getrennt im Fehlertext.

Beispiel VBScript:

```
Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet
Call oRecordSet.DialogEditFormula("Berechnungsart", "Sum(Einzelpreis)", True,
&H08000000)
Call oRecordSet.DialogEditFormula("", "Now()", False, &H04000000)
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
RecordSet recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet;
recordSet.DialogEditFormula("Berechnungsart", "Sum(Einzelpreis)", true,
0x08000000);
recordSet.DialogEditFormula("", "Now()", false, 0x04000000);
recordSet.Dispose();
```

DialogFilterAssistant

Beschreibung:

Stellt den Dialog des Filter-Assistenten dar. Es besteht die Möglichkeit eine .crmshare-Datei zu übergeben, welche einen bereits erstellten, zum Dialog passenden, Filterausdruck enthält. Als Rückgabewert erhält man eine .crmshare-Datei, welche den zusammengestellten Filterausdruck enthält. Das anschließende Bereinigen der erstellten .crmshare-Datei wird nicht von combit CRM durchgeführt.

Parameter:

Parametername	Typ	Beschreibung
WindowTitle	String	Reserviert: Aktuell bitte eine leere Zeichenkette übergeben.
FilterStatementFile	String	Pfad zu einer .crmshare-Datei, welche einen bereits erstellten, zum Dialog passenden, Filterausdruck enthält. Falls eine solche Datei nicht vorhanden ist kann eine leere Zeichenkette übergeben werden.
DisableAskString	Bool	Optional. Standardwert: False True: Benutzereingabe als Filter-Element wird nicht angeboten. False: Benutzereingabe als Filter-Element wird angeboten.
ParentWindow	Long	Optional. Standardwert: Aktuell aktives cRM-Fenster.

		Ermöglicht die Übergabe eines Handles für das Elternfenster.

Rückgabewert:

String

Wert	Beschreibung
Pfad zu einer .crmshare-Datei	Der Filterausdruck konnte erfolgreich in eine .crmshare-Datei überführt werden.
$CANCEL$	Der Filter-Dialog wurde vom Benutzer geschlossen bzw. die Eingabe abgebrochen.
$ERROR$	Es ist ein Fehler aufgetreten (z. B. .crmshare-Datei konnte nicht erzeugt werden, der übergebene Filter passt nicht zur Ansicht, auf der das RecordSet basiert, der Filter ist von einem anderen Filtertyp, als der Dialog, der angezeigt werden soll).

Beispiel VBScript:

```
Dim sFilterFileToLoad : sFilterFileToLoad = cRM.DialogSelectFile("Auswahl zu
bearbeitender Filter", True, "", "Filter (*.crmshare)|*.crmshare|Alle
Dateien|*.*||", 0)
Dim sUserDefinedFilterFile : sUserDefinedFilterFile =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.DialogFilterAssistant("
", sFilterFileToLoad, False)

If (sUserDefinedFilterFile <> "$CANCEL$" AND InStr(sUserDefinedFilterFile,
"$ERROR$") = 0) Then
    Dim sNewFilePath : sNewFilePath = cRM.DialogSelectFile("Filterausdruck
speichern", False, "", "Filter (*.crmshare)|*.crmshare|Alle Dateien|*.*||", 0)

    If (Len(sNewFilePath) > 0) Then
       Dim oFSO : Set oFSO = CreateObject("Scripting.FileSystemObject")
       Call oFSO.MoveFile(sUserDefinedFilterFile, sNewFilePath)
       Set oFSO = Nothing
    End If

End If
```

Beispiel C#-Script:

```
string filterFileToLoad = cRM.DialogSelectFile("Auswahl zu bearbeitender Filter",
true, "", "Filter (*.crmshare)|*.crmshare|Alle Dateien|*.*||", 0);
string userDefinedFilterFile =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.DialogFilterAssistant("
", filterFileToLoad, false);

if (userDefinedFilterFile != "$CANCEL$" &&
!userDefinedFilterFile.Contains("$ERROR$"))
{
    string newFilePath = cRM.DialogSelectFile("Filterausdruck speichern", false,
"", "Filter (*.crmshare)|*.crmshare|Alle Dateien|*.*||", 0);

    if (newFilePath.Length > 0)
    {
        System.IO.File.Move(userDefinedFilterFile, newFilePath);
    }
}
```

DialogFilterGeneral

Beschreibung:

Stellt den Dialog des Filter Allgemein dar. Es besteht die Möglichkeit eine .crmshare-Datei zu übergeben, welche einen bereits erstellten, zum Dialog passenden, Filterausdruck enthält. Als Rückgabewert erhält man eine .crmshare-Datei, welche den zusammengestellten Filterausdruck enthält. Das anschließende Bereinigen der erstellten .crmshare-Datei wird nicht von combit CRM durchgeführt.

Parameter:

Parametername	Typ	Beschreibung
WindowTitle	String	Reserviert: Aktuell bitte eine leere Zeichenkette übergeben.
FilterStatementFile	String	Pfad zu einer .crmshare-Datei, welche einen bereits erstellten, zum Dialog passenden, Filterausdruck enthält. Falls eine solche Datei nicht vorhanden ist kann eine leere Zeichenkette übergeben werden.
DisableAskString	Bool	Optional. Standardwert: False True: Benutzereingabe als Filter-Element wird nicht angeboten. False: Benutzereingabe als Filter-Element wird angeboten.
ParentWindow	Long	Optional. Standardwert: Aktuell aktives cRM-Fenster. Ermöglicht die Übergabe eines Handles für das Elternfenster.

Rückgabewert:

String

Wert	Beschreibung
Pfad zu einer .crmshare-Datei	Der Filterausdruck konnte erfolgreich in eine .crmshare-Datei überführt werden.
$CANCEL$	Der Filter-Dialog wurde vom Benutzer geschlossen bzw. die Eingabe abgebrochen.
$ERROR$	Es ist ein Fehler aufgetreten (z. B. .crmshare-Datei konnte nicht erzeugt werden, der übergebene Filter passt nicht zur Ansicht, auf der das RecordSet basiert, der Filter ist von einem anderen Filtertyp, als der Dialog, der angezeigt werden soll).

Beispiel VBScript:

```
Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet
Call oRecordSet.DialogFilterGeneral("", "", False)
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
RecordSet recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet;
recordSet.DialogFilterGeneral("", "", false);
recordSet.Dispose();
```

DialogFilterSQLQuery

Beschreibung:

Stellt den Dialog der freien SQL-Abfrage dar. Es besteht die Möglichkeit eine .crmshare-Datei zu übergeben, welche einen bereits erstellten, zum Dialog passenden, Filterausdruck enthält. Als Rückgabewert erhält man eine .crmshare-Datei, welche den zusammengestellten Filterausdruck enthält. Das anschließende Bereinigen der erstellten .crmshare-Datei wird nicht von combit CRM durchgeführt.

Parameter:

Parametername	Typ	Beschreibung
WindowTitle	String	Reserviert: Aktuell bitte eine leere Zeichenkette übergeben.

FilterStatementFile	String	Pfad zu einer .crmshare-Datei, welche einen bereits erstellten, zum Dialog passenden, Filterausdruck enthält. Falls eine solche Datei nicht vorhanden ist kann eine leere Zeichenkette übergeben werden.
DisableAskString	Bool	Optional. Reserviert. Der Parameterwert wird derzeit ignoriert.
ParentWindow	Long	Optional. Standardwert: Aktuelle aktives cRM-Fenster. Ermöglicht die Übergabe eines Handles für das Elternfenster.

Rückgabewert:

String

Wert	Beschreibung
Pfad zu einer .crmshare-Datei	Der Filterausdruck konnte erfolgreich in eine .crmshare-Datei überführt werden.
$CANCEL$	Der Filter-Dialog wurde vom Benutzer geschlossen bzw. die Eingabe abgebrochen.
$ERROR$	Es ist ein Fehler aufgetreten (z. B. .crmshare-Datei konnte nicht erzeugt werden, der übergebene Filter passt nicht zur Ansicht, auf der das RecordSet basiert, der Filter ist von einem anderen Filtertyp, als der Dialog, der angezeigt werden soll).

Beispiel VBScript:

```
Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet
Call oRecordSet.DialogFilterSQLQuery("", "")
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
RecordSet recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet;
recordSet.DialogFilterSQLQuery("", "");
recordSet.Dispose();
```

DialogSelectRecord

Beschreibung:

Ermöglicht die Auswahl eines Datensatzes basierend auf dem aktuellen Filter des **RecordSet** Objektes. Das Ergebnis ist ein Objekt vom Typ **Record**.

Wenn kein Datensatz ausgewählt wurde, wird NULL zurückgegeben. Das Objekt ist dann somit ungültig! Es wird ein fully-dynamic RecordSet als Basis benötigt, weitere Informationen finden Sie unter **Änderungen und Neuerungen.**

Hinweis: **DialogSelectRecord** speichert die Spalten/Layoutkonfiguration und die vom Anwender zuletzt eingestellte Sortierung der Datensatzübersichtsliste ab (Voreinstellung: in der Registry). Der Name der Sektion der Konfiguration lautet "DialogSelect_"+<Ansichtenname>. Dieser kann jedoch individuell anderweitig vorgegeben werden, indem man beim Funktionsaufruf einen eigenen Namen im Parameter für den Fenstertitel durch TAB (chr$(9)) getrennt mit übergibt.

Sobald **DialogSelectRecord** für ein per **ViewConfig** erzeugtes **RecordSet** (*ViewConfig.CreateRecordSet*) verwendet wird, kann die Performance optimiert werden, wenn vor dem Methodenaufruf die Sortierung des **RecordSet** gesetzt wird (*ViewConfig.CreateRecordSet("SetSortOrder:n")*).

Wichtig: Nach dem Aufruf von **DialogSelectRecord** für einen per **ViewConfig.CreateRecordSet** erzeugten RecordSet darf für den betreffenden RecordSet keine **Move**-Methode aufgerufen werden, da sonst der zurückgegebene Record u.U. seine Werte verändert!

Parameter:

Parametername	Typ	Beschreibung
WindowTitle	String	Fenstertitel des erzeugten Dialoges. Der Titel kann auch leer übergeben werden, dann wird eine Grundeinstellung genommen.
AllowUserDefined-SortOrder	Bool	True: Eine etwaige vom Benutzer in diesem Dialog zuletzt eingestellte Sortierung wird verwendet. Standardmäßig wird die erste Sortierung aktiviert, die in der zum RecordSet gehörenden Ansicht definiert wurde. False: Es wird die im RecordSet eingestellte Sortierung verwendet. Ist keine eingestellt, so wird die erste Sortierung aktiviert, die in der zum RecordSet gehörenden Ansicht definiert wurde.
nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Typ:

Record

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet.DialogSelect
Record("Datensatzauswahl", True)
If (Not oRecord Is Nothing) Then
    Call oRecord.PrintLabel("PRV", "%PRJDIR%\Druckvorlagen\Kontakte -
Adressetikett.lbl", True, "", False)
    Set oRecord = Nothing
End If
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet().DialogSele
ctRecord("Datensatzauswahl", true);
if (record != null)
{
    record.PrintLabel("PRV", @"%PRJDIR%\Druckvorlagen\Kontakte -
Adressetikett.lbl", true, "", false);
    record.Dispose();
}
```

DialogSelectRecordMultiple

Beschreibung:

Ermöglicht die Auswahl mehrerer Datensätze basierend auf dem aktuellen Filter des **RecordSet** Objektes. Das Ergebnis ist ein neues unabhängiges Objekt vom Typ **RecordSet**, welches die gefiltertern Datensätze enthält.

Wenn ohne Setzen eines Häkchens einer Checkbox für die Datensatzauswahl der Dialog mit OK bestätigt wird, wird der blau hinterlegte und selektierte Datensatz an das **RecordSet** übergeben. Dies kann ebenso

mit einem Doppelklick auf den gewünschten Datensatz erreicht werden, sofern noch keine anderen Datensätze angehakt wurden. Wenn kein Datensatz ausgewählt werden soll, muss die Abbrechen-Schaltfläche betätigt werden – es wird dann NULL zurückgegeben. Das Objekt ist somit ungültig! Es wird ein fully-dynamic RecordSet als Basis benötigt, weitere Informationen finden Sie unter **Änderungen und Neuerungen.**

Wichtig: Hierfür wird ein Feld vom internen Feldtyp 'Datensatz-ID' benötigt, welches mit entsprechenden global eindeutigen IDs gefüllt ist. Falls es sich nicht um ein Feld vom externen SQL Feldtyp 'uniqueidentifier', sondern 'nvarchar' handelt, muss dieses mindestens 32 Zeichen lang sein und die GUID in der cRM-internen Darstellung enthalten, z. B. "ce05a0f11dbf464f8a7f14666ddd0f3a".

Hinweis: **DialogSelectRecordMultiple** speichert die Spalten/Layoutkonfiguration und die vom Anwender zuletzt eingestellte Sortierung der Datensatzübersichtsliste ab (Voreinstellung: in der Registry). Der Name der Sektion der Konfiguration lautet "DialogSelect_"+<Ansichtenname>. Dieser kann jedoch individuell anderweitig vorgegeben werden, indem man beim Funktionsaufruf einen eigenen Namen im Parameter für den Fenstertitel durch TAB (chr$(9)) getrennt mit übergibt.

Sobald **DialogSelectRecordMultiple** für ein per **ViewConfig** erzeugtes **RecordSet** (*ViewConfig.CreateRecordSet*) verwendet wird, kann die Performance optimiert werden, wenn vor dem Methodenaufruf die Sortierung des **RecordSet** gesetzt wird (*ViewConfig.CreateRecordSet("SetSortOrder:n")*).

Parameter:

Parametername	Typ	Beschreibung
WindowTitle	String	Fenstertitel des erzeugten Dialoges. Der Titel kann auch leer übergeben werden, dann wird eine Grundeinstellung genommen.
AllowUserDefined-SortOrder	Bool	True: Eine etwaige vom Benutzer in diesem Dialog zuletzt eingestellte Sortierung wird verwendet. Standardmäßig wird die erste Sortierung aktiviert, die in der zum RecordSet gehörenden Ansicht definiert wurde. False: Es wird die im RecordSet eingestellte Sortierung verwendet. Ist keine eingestellt, so wird die erste Sortierung aktiviert, die in der zum RecordSet gehörenden Ansicht definiert wurde.
nParentHandle	Long	Optional. Handle eines Fensters, das als Parent-Fenster für den Dialog verwendet werden soll. Voreinstellung: das combit CRM Hauptfenster, sofern es nicht unsichtbar ist, ansonsten das in dem Augenblick gerade aktive Vordergrundfenster

Typ:

RecordSet

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oSelectedRecordSet : Set oSelectedRecordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet.DialogSelect
RecordMultiple("Datensatzauswahl", True)
If (Not oRecordSet Is Nothing) Then
    Call cRM.DialogMessageBox("Es wurden " & CStr(oSelectedRecordSet.RecCount) & "
Datensätze ausgewählt.", "RecordSet.DialogSelectRecordMultiple", vbOkOnly)
    Set oSelectedRecordSet = Nothing
End If
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution
```

```
RecordSet selectedRecords =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet().DialogSele
ctRecordMultiple("Datensatzauswahl", true);
if (selectedRecords != null)
{
    cRM.DialogMessageBox("Es wurden " + selectedRecords.RecCount.ToString() + "
Datensätze ausgewählt.", "RecordSet.DialogSelectRecordMultiple", 0);
    selectedRecords.Dispose();
}
```

ExecuteInstantReportByName

Beschreibung:

Führt einen abgespeicherten Sofortbericht anhand seines Namens aus.

Hinweis: Es wird nur in der aktuell aktiven, d.h. sichtbaren, Ansicht nach dem Sofortbericht gesucht, d.h. gibt man einen Namen eines Sofortberichts an, der in einer anderen Ansicht definiert ist, schlägt die Methode fehl.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name des Sofortberichts. Groß- / Kleinschreibung wird beachtet!

Typ:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.ExecuteInstantReportByN
ame("Scriptname des Sofortberichts")
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.ExecuteInstantReportByN
ame("Scriptname des Sofortberichts");
```

Export

Beschreibung:

Startet den Export-Assistenten und führt das übergebene Benutzerformat aus. Es werden alle Einstellungen dieses Formates verwendet.

Parameter:

Parametername	Typ	Beschreibung
Format	String	Name des Exportformates. Es kann sich dabei um einen kompletten Dateipfad auf eine .etp Format-Datei handeln, alternativ um den Formatnamen, so wie er im Assistenten dargestellt wird, dabei werden zuerst die benutzerspezifischen Formate durchsucht, anschließend die globalen Formate. Bitte beachten Sie die Großkleinschreibung.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.Export("Name des Exportformats")
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.Export("Name des Exportformats");
```

FindRecordByDupeCheckCriteria

Beschreibung:

Führt den Online-Dublettencheck anhand der übergebenen Werte und den definierten Kriterien für Dubletten aus.

Hinweis: Vor dem Aufruf von **FindRecordByDupeCheckCriteria** müssen die benötigten Felder per **ViewConfig.DupeCheckCriteria** ausgelesen werden.

Parameter:

Parametername	Typ	Beschreibung
sRecordID	String	ID des Datensatzes, welcher vom Dubletten-Check ausgeschlossen werden soll. Wenn dieser Parameter leer übergeben wird, dann werden alle Datensätze einbezogen.
sFieldContents1	String	Inhalt des ersten Kriteriums für den Dublettencheck.
sFieldContents2	String	Optional, Inhalt des zweiten Kriteriums für den Dublettencheck.
sFieldContents3	String	Optional, Inhalt des drittens Kriteriums für den Dublettencheck.
sFieldContents4	String	Optional, Inhalt des vierten Kriteriums für den Dublettencheck.
sFieldContents5	String	Optional, Inhalt des fünften Kriteriums für den Dublettencheck.
sFieldContents6	String	Optional, Inhalt des sechsten Kriteriums für den Dublettencheck.
sFieldContents7	String	Optional, Inhalt des siebten Kriteriums für den Dublettencheck.
sFieldContents8	String	Optional, Inhalt des achten Kriteriums für den Dublettencheck.
bUseCurrentQuery	Bool	Optional (Voreinstellung: False), gibt an, ob lediglich innerhalb des aktuellen Filters gesucht werden soll.

Rückgabewert:

RecordSet

Beispiel VBScript:

```
Dim sFieldsDupeCheck : sFieldsDupeCheck =
cRM.CurrentProject.ActiveViews.ActiveView.Config.DupeCheckCriteria
Dim dicFieldsDupeCheck : Set dicFieldsDupeCheck =
CreateObject("Scripting.Dictionary")
Dim aFieldsDupeCheck : aFieldsDupeCheck = Split(sFieldsDupeCheck, vbTab)
Dim sField, sUserInput

For Each sField in aFieldsDupeCheck
    sUserInput = cRM.DialogInputBox("Welcher Inhalt soll für das Feld """ & sField
& """ für den Dublettencheck verwendet werden?",
"RecordSet.FindRecordByDupeCheckCriteria")

    If (sUserInput <> "$CANCEL$") Then
```

```
            Call dicFieldsDupeCheck.Add(sField, sUserInput)
        Else
            Exit For
        End If
    Next

    Dim sUserInputInDictionary
    Dim sFieldContent1 : sFieldContent1 = ""
    Dim sFieldContent2 : sFieldContent2 = ""
    Dim sFieldContent3 : sFieldContent3 = ""
    Dim sFieldContent4 : sFieldContent4 = ""
    Dim sFieldContent5 : sFieldContent5 = ""
    Dim sFieldContent6 : sFieldContent6 = ""
    Dim sFieldContent7 : sFieldContent7 = ""
    Dim sFieldContent8 : sFieldContent8 = ""
    Dim nCounter : nCounter = 0

    If (dicFieldsDupeCheck.Count > 0) Then
        For Each sUserInputInDictionary in dicFieldsDupeCheck.Keys
            nCounter = nCounter + 1

            Select Case nCounter
                Case 1
                    sFieldContent1 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
                Case 2
                    sFieldContent2 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
                Case 3
                    sFieldContent3 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
                Case 4
                    sFieldContent4 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
                Case 5
                    sFieldContent5 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
                Case 6
                    sFieldContent6 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
                Case 7
                    sFieldContent7 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
                Case 8
                    sFieldContent8 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
           End Select
        Next
    End If

    Set dicFieldsDupeCheck = Nothing

    Dim oRecordSet : Set oRecordSet =
    cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy
    Call oRecordSet.FindRecordByDupeCheckCriteria("", sFieldContent1, sFieldContent2,
    sFieldContent3, sFieldContent4, sFieldContent5, sFieldContent6, sFieldContent7,
    sFieldContent8, False)

    If (Not oRecordSet Is Nothing) Then

        If (oRecordSet.RecCount > 0) Then

            Call cRM.DialogMessageBox("Es wurde mindestens eine Dublette gefunden.",
    "RecordSet.FindRecordByDupeCheckCriteria", vbOkOnly)

        End If

    End If

    Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
    string fieldsDupeCheckCriteria =
    cRM.CurrentProject.ActiveViews.ActiveView.Config.DupeCheckCriteria;
    System.Collections.Generic.Dictionary<string, string> dicFieldsDupeCheck = new
    System.Collections.Generic.Dictionary<string, string>();
    string[] fieldsDupeCheck = fieldsDupeCheckCriteria.Split('\t');
    string userInput = null;
```

```
foreach (string field in fieldsDupeCheck)
{
    userInput = cRM.DialogInputBox(@"Welcher Inhalt soll für das Feld """ + field
+ @""" für den Dublettencheck verwendet werden?",
"RecordSet.FindRecordByDupeCheckCriteria");

    if (userInput != "$CANCEL$")
    {
        dicFieldsDupeCheck.Add(field, userInput);
    }
    else
    {
        break;
    }
}

int counter = 0;
string fieldContent1 = null;
string fieldContent2 = null;
string fieldContent3 = null;
string fieldContent4 = null;
string fieldContent5 = null;
string fieldContent6 = null;
string fieldContent7 = null;
string fieldContent8 = null;
RecordSet recordSet = null;

if (dicFieldsDupeCheck.Count > 0)
{
    foreach (var item in dicFieldsDupeCheck)
    {
        counter++;

        switch (counter)
        {
            case 1:
                fieldContent1 = item.Value;
                break;
            case 2:
                fieldContent2 = item.Value;
                break;
            case 3:
                fieldContent3 = item.Value;
                break;
            case 4:
                fieldContent4 = item.Value;
                break;
            case 5:
                fieldContent5 = item.Value;
                break;
            case 6:
                fieldContent6 = item.Value;
                break;
            case 7:
                fieldContent7 = item.Value;
                break;
            case 8:
                fieldContent8 = item.Value;
                break;
            default:
                break;
        }
    }

    recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy();
    recordSet.FindRecordByDupeCheckCriteria("", fieldContent1, fieldContent2,
fieldContent3, fieldContent4, fieldContent5, fieldContent6, fieldContent7,
fieldContent8, false);
```

```
        if (recordSet != null)
        {
            if (recordSet.RecCount > 0)
            {
                cRM.DialogMessageBox("Es wurde mindestens eine Dublette gefunden.",
"RecordSet.FindRecordByDupeCheckCriteria", 0);
                recordSet.Dispose();
            }
        }
}
```

FindRecordByEmail

Beschreibung:

Sucht alle Datensätze, bei denen in einem E-Mail-Feld eine bestimmte E-Mail-Adresse vorhanden ist. Diese unscharfe Suche findet auch E-Mail-Adressen derselben Domain falls kein direkter Treffer vorhanden ist.

Hinweis: Falls der COM-RecordSet auf einer sichtbaren Ansicht basiert, so wechselt die Ansicht bei mehreren Treffern automatisch in die Listenübersicht. Bei lediglich einem einzelnen Treffer erfolgt der Wechsel in die Eingabemasken-Ansicht.

Parameter:

Parametername	Typ	Beschreibung
sEmail	String	Die E-Mail-Adresse, nach der in allen E-Mail-Feldern gesucht werden soll.
bReduceEmail	Bool	Wenn True, dann wird die E-Mail-Adresse schrittweise in bis zu vier Durchläufen durch Auslassen von Top-Level-Domain, Domain, Lokalteil gesucht, falls es nicht bereits Treffer gab. So können auch E-Mails unterschiedlicher Domains gefunden werden.
bUseCurrentQuery	Bool	Optional (Voreinstellung: False), gibt an, ob lediglich innerhalb des aktuellen Filters gesucht werden soll.

Rückgabewert:

Bool

Wert	Beschreibung
True	Es gab Treffer, für den RecordSet ist nun der entsprechende Filter aktiv.
False	Es gab keine Treffer.

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.FindRecordByEmail("sole
il@luna-aventuras.net", True)
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.FindRecordByEmail("sole
il@luna-aventuras.net", true);
```

FindRecordByPhoneNumber

Beschreibung:

Sucht alle Datensätze, bei denen in einem Telefon-Feld eine bestimmte Telefonnummer vorhanden ist, analog zur Rufnummernerkennung durch den phone manager.

Hinweis: Falls der COM-RecordSet auf einer sichtbaren Ansicht basiert, so wechselt die Ansicht bei mehreren Treffern automatisch in die Listenübersicht. Bei lediglich einem einzelnen Treffer erfolgt der Wechsel in die Eingabemasken-Ansicht.

Parameter:

Parametername	Typ	Beschreibung
sPhoneNo	String	Die Telefonnummer, nach der in allen Telefonfeldern gesucht werden soll. Die Telefonnummer darf telefonspez. Sonderzeichen wie zum Beispiel '/', '+', '-' enthalten.
bReducePhoneNo	Bool	Wenn True, dann wird die Telefonnummer schrittweise um bis zu 4 Stellen verkürzt, falls es nicht bereits Treffer gab. So können auch unterschiedliche Durchwahl-Nummern gefunden werden.
bUseCurrentQuery	Bool	Optional (Voreinstellung: False), gibt an, ob lediglich innerhalb des aktuellen Filters gesucht werden soll.

Rückgabewert:

Bool

Wert	Beschreibung
True	Es gab Treffer, für den RecordSet ist nun der entsprechende Filter aktiv.
False	Es gab keine Treffer.

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.FindRecordByPhoneNumber
("07531/0999999-1", True)
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.FindRecordByPhoneNumber
("07531/0999999-1", true);
```

Import

Beschreibung:

Startet den Import-Assistenten und führt das übergebene Benutzerformat aus. Es werden alle Einstellungen dieses Formates verwendet.

Wichtig: Wenn ein Format zum automatischen Ausführen übergeben wird, oder die Methoden für einen unsichtbaren **RecordSet** aufgerufen werden, dann ist ein Import von unbekannten Codefeldern – im Gegensatz zu interaktivem Importieren – nicht möglich, diese werden dann ignoriert.

Parameter:

Parametername	Typ	Beschreibung
Format	String	Name des Importformates. Es kann sich dabei um einen kompletten Dateipfad auf eine .itp Format-Datei handeln, alternativ um den Formatnamen, so wie er im Assistenten dargestellt wird, dabei werden zuerst die benutzerspezifischen Formate durchsucht, anschließend die globalen Formate. Bitte beachten Sie die Großkleinschreibung.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.Import("Name des Importformats")
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.Import("Name des Importformats");
```

MoveFirst

Beschreibung:

Bewegt den Datensatz-Zeiger auf den Anfang des **RecordSet**.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Durchläuft alle Datensätze in einem kopierten RecordSet und summiert die Gesamtumsätze. Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRecordSetCopy : Set oRecordSetCopy = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy
Dim oRecord
Dim nTurnover : nTurnover = 0

If (oRecordSetCopy.MoveFirst() = True) Then
    Set oRecord = oRecordSetCopy.CurrentRecord

    Do
        nTurnover = nTurnover + oRecord.GetContentsValueByName("Turnover")
    Loop Until Not oRecordSetCopy.MoveNext()

    Set oRecord = Nothing
End If

Call cRM.DialogMessageBox("Der Gesamtumsatz der aktuell angezeigten Datensätze beträgt: " & CStr(nTurnover) & " EUR.", "RecordSet.MoveFirst", vbOkOnly)

Set oRecordSetCopy = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Datensätze in einem kopierten RecordSet und summiert die Gesamtumsätze. Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

RecordSet recordSetCopy = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy();
Record record;
double turnover = 0;

if (recordSetCopy.MoveFirst() == true)
{
    record = recordSetCopy.CurrentRecord;

    do
    {
        turnover = turnover + (double)record.GetContentsValueByName("Turnover");
    } while (!recordSetCopy.MoveNext());

    record.Dispose();
```

```
}

cRM.DialogMessageBox("Der Gesamtumsatz der aktuell angezeigten Datensätze beträgt:
" + turnover.ToString() + " EUR.", "RecordSet.MoveFirst", 0);

recordSetCopy.Dispose();
```

MoveLast

Beschreibung:

Bewegt den Datensatz-Zeiger auf das Ende der Datenbank.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Durchläuft alle Datensätze in einem kopierten RecordSet und summiert die
Gesamtumsätze. Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-
Solution

Dim oRecordSetCopy : Set oRecordSetCopy =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy
Dim oRecord
Dim nTurnover : nTurnover = 0

If (oRecordSetCopy.MoveLast() = True) Then
    Set oRecord = oRecordSetCopy.CurrentRecord

    Do
        nTurnover = nTurnover + oRecord.GetContentsValueByName("Turnover")
    Loop Until Not oRecordSetCopy.MovePrevious()

    Set oRecord = Nothing
End If

Call cRM.DialogMessageBox("Der Gesamtumsatz der aktuell angezeigten Datensätze
beträgt: " & CStr(nTurnover) & " EUR.", "RecordSet.MoveFirst", vbOkOnly)

Set oRecordSetCopy = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Datensätze in einem kopierten RecordSet und summiert die
Gesamtumsätze. Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-
Solution

RecordSet recordSetCopy =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy();
Record record;
double turnover = 0;

if (recordSetCopy.MoveLast() == true)
{
    record = recordSetCopy.CurrentRecord;

    do
    {
        turnover = turnover + (double)record.GetContentsValueByName("Turnover");
    } while (!recordSetCopy.MovePrevious());

    record.Dispose();

}

cRM.DialogMessageBox("Der Gesamtumsatz der aktuell angezeigten Datensätze beträgt:
" + turnover.ToString() + " EUR.", "RecordSet.MoveLast", 0);

recordSetCopy.Dispose();
```

MoveNext

Beschreibung:

Bewegt den Datensatz-Zeiger um einen Datensatz vorwärts.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Durchläuft alle Datensätze in einem kopierten RecordSet und summiert die
Gesamtumsätze. Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-
Solution

Dim oRecordSetCopy : Set oRecordSetCopy =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy
Dim oRecord
Dim nTurnover : nTurnover = 0

If (oRecordSetCopy.MoveFirst() = True) Then
    Set oRecord = oRecordSetCopy.CurrentRecord

    Do
        nTurnover = nTurnover + oRecord.GetContentsValueByName("Turnover")
    Loop Until Not oRecordSetCopy.MoveNext()

    Set oRecord = Nothing
End If

Call cRM.DialogMessageBox("Der Gesamtumsatz der aktuell angezeigten Datensätze
beträgt: " & CStr(nTurnover) & " EUR.", "RecordSet.MoveFirst", vbOkOnly)

Set oRecordSetCopy = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Datensätze in einem kopierten RecordSet und summiert die
Gesamtumsätze. Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-
Solution

RecordSet recordSetCopy =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy();
Record record;
double turnover = 0;

if (recordSetCopy.MoveFirst() == true)
{
    record = recordSetCopy.CurrentRecord;

    do
    {
        turnover = turnover + (double)record.GetContentsValueByName("Turnover");
    } while (!recordSetCopy.MoveNext());

    record.Dispose();

}

cRM.DialogMessageBox("Der Gesamtumsatz der aktuell angezeigten Datensätze beträgt:
" + turnover.ToString() + " EUR.", "RecordSet.MoveNext", 0);

recordSetCopy.Dispose();
```

MovePrevious

Beschreibung:

Bewegt den Datensatz-Zeiger um einen Datensatz rückwärts.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Durchläuft alle Datensätze in einem kopierten RecordSet und summiert die
Gesamtumsätze. Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-
Solution

Dim oRecordSetCopy : Set oRecordSetCopy =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy
Dim oRecord
Dim nTurnover : nTurnover = 0

If (oRecordSetCopy.MoveLast() = True) Then
    Set oRecord = oRecordSetCopy.CurrentRecord

    Do
        nTurnover = nTurnover + oRecord.GetContentsValueByName("Turnover")
    Loop Until Not oRecordSetCopy.MovePrevious()

    Set oRecord = Nothing
End If

Call cRM.DialogMessageBox("Der Gesamtumsatz der aktuell angezeigten Datensätze
beträgt: " & CStr(nTurnover) & " EUR.", "RecordSet.MoveFirst", vbOkOnly)

Set oRecordSetCopy = Nothing
```

Beispiel C#-Script:

```
// Durchläuft alle Datensätze in einem kopierten RecordSet und summiert die
Gesamtumsätze. Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-
Solution

RecordSet recordSetCopy =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy();
Record record;
double turnover = 0;

if (recordSetCopy.MoveLast() == true)
{
    record = recordSetCopy.CurrentRecord;

    do
    {
        turnover = turnover + (double)record.GetContentsValueByName("Turnover");
    } while (!recordSetCopy.MovePrevious());

    record.Dispose();

}

cRM.DialogMessageBox("Der Gesamtumsatz der aktuell angezeigten Datensätze beträgt:
" + turnover.ToString() + " EUR.", "RecordSet.MovePrevious", 0);

recordSetCopy.Dispose();
```

NewRecord

Beschreibung:

Erzeugt einen neuen Datensatz und liefert diesen als Objekt vom Typ **Record** zurück.

Wichtig: Der neu angelegte **Record** muss nach dem Speichern nicht mit **Unlock** freigegeben werden (da er ja eben erst erzeugt wurde) und enthält ohne explizites Aufrufen von **SetContents**…-Methoden die jeweiligen Feldvorbelegungen, Auto-Nummern und andere Sonderfeldfunktionen (Erfassungsbenutzer, Erfassungsdatum, …).

Für ein mit dieser Methode erzeugtes **Record** Objekt dürfen lediglich die Methoden **Lock**, **Get/SetContents...** mit einem (einmaligen) abschließenden **Save** und einem etwaigen (einmaligen) **Unlock** verwendet werden. Um andere Methoden des **Record** Objektes verwenden zu können, muss das **Record** Objekt freigegeben, neu initialisiert und auf den soeben erzeugten Datensatz positioniert werden.

Die Methode prüft das Ansichtsrecht *Datensatz neu anlegen*. Im Fehlerfall erhält man keine visuelle Meldung, d.h. im Script muss eine visuelle Benachrichtigung erfolgen, wenn die Methode fehlschlägt.

Die Datensatz-ID steht sofort nach der Ausführung von **NewRecord**, auch ohne explizites Speichern, zur Verfügung und kann per **GetContentsByName** abgerufen werden.

Rückgabewert:

Record

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecord : Set oRecord =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet.NewRecord
Call oRecord.SetContentsByName("Name", "Soleil")
Call oRecord.SetContentsByName("Firstname", "Jean")
Call oRecord.Save()
Set oRecord = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Record record =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet().NewRecord(
);
record.SetContentsByName("Name", "Soleil");
record.SetContentsByName("Firstname", "Jean");
record.Save();
record.Dispose();
```

PrintCard

Beschreibung:

Druckt ein Karteikartenprojekt aus. Neben dem Druckziel muss u.a. der Name des gewünschten Druckprojektes angegeben werden.

Parameter:

Parametername	Typ	Beschreibung
Media	String	Die zur Verfügung stehenden Ausgabe-Medien finden Sie im Kapitel **Ausgabe-Medium.** Über diesen Parameter werden zusätzliche Export-Optionen unterstützt. Weitere Informationen finden Sie im Kapitel **Export-Optionen für Print-Methoden**.
FileName	String	Dateiname inkl. Pfad des Druckprojektes.
Silent	Bool	Legt fest, ob der Druck (nach Möglichkeit) ohne Benutzerinteraktion erfolgen soll.
Range	Bool	True: Es werden alle Datensätze im aktuellen Filter/Datenbank ausgegeben, entspricht dem Seriendruck. False: Es wird nur der aktuelle Datensatz ausgegeben, entspricht dem Einzeldruck.
OutputFileName	String	Evtl. Name und Pfad der zu erzeugenden Ausgabe-Datei bei Ausgabe-Medien wie z. B. "RTF" oder "HTML". In allen anderen Fällen kann ein leerer String übergeben werden.
UseModalPreviewWindow	Bool	Optional. Legt fest, ob der Druck auf Vorschau (PRV) in das normale Vorschaufenster gedruckt wird (True, Voreinstellung) oder in die Berichtsansicht (False).

Rückgabewert:

Bool

Hinweis: Druckt man über ein **RecordSet**-Objekt, welches aus einer **ViewConfig** erstellt wurde, stehen momentan die Variablen *cRM.Project.?* und *cRM.View.?* nicht zur Verfügung.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet
Call oRecordSet.PrintCard("PRV", "%PRJDIR%\Druckvorlagen\Kontakte -
Serienbriefvorlage.crd", True, True, "", False)
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

RecordSet recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet;
recordSet.PrintCard("PRV", @"%PRJDIR%\Druckvorlagen\Kontakte -
Serienbriefvorlage.crd", true, true, "", false);
recordSet.Dispose();
```

PrintLabel

Beschreibung:

Druckt ein Etikettenprojekt aus. Neben dem Druckziel muss u.a. der Name des gewünschten Druckprojektes angegeben werden.

Parameter:

Parametername	Typ	Beschreibung

Media	String	Die zur Verfügung stehenden Ausgabe-Medien finden Sie im Kapitel **Ausgabe-Medium.** Über diesen Parameter werden zusätzliche Export-Optionen unterstützt. Weitere Informationen finden Sie im Kapitel **Export-Optionen für Print-Methoden**.
FileName	String	Dateiname inkl. Pfad des Druckprojektes.
Silent	Bool	Legt fest, ob der Druck (nach Möglichkeit) ohne Benutzerinteraktion erfolgen soll.
Range	Bool	True: Es werden alle Datensätze im aktuellen Filter/Datenbank ausgegeben, entspricht dem Seriendruck. False: Es wird nur der aktuelle Datensatz ausgegeben, entspricht dem Einzeldruck.
OutputFileName	String	Evtl. Name und Pfad der zu erzeugenden Ausgabe-Datei bei Ausgabe-Medien wie z. B. "RTF" oder "HTML". In allen anderen Fällen kann ein leerer String übergeben werden.
UseModalPreviewWindow	Bool	Optional. Legt fest, ob der Druck auf Vorschau (PRV) in das normale Vorschaufenster gedruckt wird (True, Voreinstellung) oder in die Berichtsansicht (False).

Rückgabewert:

Bool

Hinweis: Druckt man über ein **RecordSet** Objekt, welches aus einer **ViewConfig** erstellt wurde, stehen momentan die Variablen *cRM.Project.?* und *cRM.View.?* nicht zur Verfügung.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet
Call oRecordSet.PrintLabel("PRV", "%PRJDIR%\Druckvorlagen\Kontakte -
Adressetikett.lbl", True, True, "", False)
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

RecordSet recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet;
recordSet.PrintLabel("PRV", @"%PRJDIR%\Druckvorlagen\Kontakte -
Adressetikett.lbl", true, true, "", false);
recordSet.Dispose();
```

PrintReport

Beschreibung:

Druckt ein Listenprojekt aus. Neben dem Druckziel muss u.a. der Name des gewünschten Druckprojektes angegeben werden.

Parameter:

Parametername	Typ	Beschreibung
Media	String	Die zur Verfügung stehenden Ausgabe-Medien finden Sie im Kapitel **Ausgabe-Medium.** Über diesen Parameter werden zusätzliche Export-Optionen unterstützt. Weitere Informationen finden Sie im Kapitel **Export-Optionen für Print-Methoden**.
FileName	String	Dateiname inkl. Pfad des Druckprojektes
Silent	Bool	Legt fest, ob der Druck (nach Möglichkeit) ohne Benutzerinteraktion erfolgen soll.

Range	Bool	True: Es werden alle Datensätze in der aktuellen Ansicht ausgegeben, entspricht dem Seriendruck. False: Es wird nur der aktuelle Datensatz ausgegeben, entspricht dem Einzeldruck.
OutputFileName	String	Evtl. Name und Pfad der zu erzeugenden Ausgabe-Datei bei Ausgabe-Medien wie z. B. "RTF" oder "HTML". In allen anderen Fällen kann ein leerer String übergeben werden.
UseModalPreviewWindow	Bool	Optional. Legt fest, ob der Druck auf Vorschau (PRV) in das normale Vorschaufenster gedruckt wird (True, Voreinstellung) oder in die Berichtsansicht (False).

Rückgabewert:

Bool

Hinweis: Druckt man über ein RecordSet Objekt, welches aus einer ViewConfig erstellt wurde, stehen momentan die Variablen cRM.Project.? und cRM.View.? nicht zur Verfügung.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet
Call oRecordSet.PrintReport("PRV", "%PRJDIR%\Druckvorlagen\Kontakte -
Geburtstagsliste.lst", True, True, "", False)
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

RecordSet recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet;
recordSet.PrintReport("PRV", @"%PRJDIR%\Druckvorlagen\Kontakte -
Geburtstagsliste.lst", true, true, "", false);
recordSet.Dispose();
```

RecCountWithoutGeoCoordinates

Beschreibung:

Gibt die Anzahl der unkodierten Datensätze für eine Adress-Definition zurück. Die Geokodierung der Adresse ist Voraussetzung für den Umkreis-Filter, siehe auch SetFilterByGeo.

Parameter:

Parametername	Typ	Beschreibung
AliasName	String	Name der Adresse (siehe Ansichtskonfiguration > Adressen)

Rückgabewert:

Long

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("Derzeit gibt es " &
CStr(cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.RecCountWithoutGeo
Coordinates("Kontakte_Adresse")) & " unkodierte Datensätze.",
"RecordSet.RecCountWithoutGeoCoordinates", vbOkOnly)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox("Derzeit gibt es " +
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.RecCountWithoutGeoCoord
```

```
inates("Kontakte_Adresse").ToString() + " unkodierte Datensätze.",
"RecordSet.RecCountWithoutGeoCoordinates", 0);
```

RelationalAppend

Beschreibung:

Führt eine abgespeicherte Vorlage für relationales Ergänzen über den Namen aus. Diese kann über den Menüpunkt **Daten > Relational ergänzen > Allgemein** erzeugt werden.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name der abgespeicherten Vorlage. Hinweis: Bitte beachten Sie, dass nicht die "Bezeichnung" der Vorlage verwendet wird, sondern der "Name für Scripte/Workflows".

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.RelationalAppend("Name
der Vorlage für das relationale Ergänzen")
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.RelationalAppend("Name
der Vorlage für das relationale Ergänzen");
```

RelationalAppendDirect

Beschreibung:

Direktes relationales Ergänzen in einer Relationsansicht. Die Übergabe der Werte erfolgt in folgendem Format:

```
<FeldName1>=<Inhalt/Formel1>\t<FeldName2>=<Inhalt/Formel2>\t...
```

Parameter:

Parametername	Typ	Beschreibung
Relation	Objekt	Relationsobjekt
Parameter	String	Wertepaar

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ActiveViews.ActiveView.Config
Dim oRelations : Set oRelations = oViewConfig.Relations
Dim oRelation : Set oRelation = oRelations.ItemByName("ID.Aktivitäten.ContactID")

Call oRecordSet.RelationalAppendDirect(oRelation, " Direction =2" & vbTab & "
ActivityType =1" & vbTab & " Comment=AskString$(""Kontaktbetreff?"", .F., "")")

Set oRelation = Nothing
Set oRelations = Nothing
Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

RecordSet recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet;
```

```
ViewConfig viewConfig = cRM.CurrentProject.ActiveViews.ActiveView.Config;
ListRelations relations = viewConfig.Relations;
Relation relation = relations.ItemByName("ID.Aktivitäten.ContactID");

recordSet.RelationalAppendDirect(relation, " Direction =2" + "\t" + " ActivityType
=1" + "\t" + @" Comment=AskString$(""Kontaktbetreff?"", .F., "")");

recordSet.Dispose();
relation.Dispose();
relations.Dispose();
viewConfig.Dispose();
```

SendBulkMail

Beschreibung:

Sendet eine Serien-E-Mail. Der Mailversand über diese Methode verhält sich analog zum Serien-Mailversand in der Anwendung.

Parameter:

Parametername	Typ	Beschreibung
TemplatePath	String	Pfad zur Mailvorlage
ErrorTagFilePath	String	Pfad zur .tag-Datei für manuelles Filtern. Diese wird erstellt, wenn es im RecordSet Datensätze mit leeren E-Mail-Feldern gibt und enthält alle diese Datensätze. Falls ein leerer String übergeben wird, wird eine Datei im temp-Verzeichnis mit Standardnamen angelegt.
Silent	Long	Optional. 0 = Hinweisdialoge werden angezeigt. 1 = Hinweisdialoge werden ausgeblendet. Standardwert: 0
MailsSentToTagFile-Path	String	Optional. Pfad zur .tag-Datei für manuelles Filtern. Diese Datei enthält alle Datensätze, an die eine E-Mail versendet werden konnte. Über diesen Parameter kann der Pfad und Name der Datei geändert werden. Ohne Angabe dieses Parameters wird die Datei in %TEMP%\MailsSentTo.tag gespeichert und bei jedem Serienmailing (auch mittels „Ausgeben > Serien-E-Mail“) überschrieben.
Files	String	Optional. Kann eine Liste von E-Mail-Anhängen beinhalten. Diese müssen dann per Semikolon separiert übergeben werden. z. B. C:\MyFiles\Report.pdf;C:\Info.doc Beachten Sie bitte, dass die übergebenen Anhänge stets zusätzlich zu den evtl. bereits in einer über den Parameter "TemplatePath" definierten Mailvorlage hinterlegten Anhängen versendet werden. Dies gilt auch dann, wenn ein übergebener E-Mail-Anhang denselben Pfad hat wie in der Mailvorlage.
ShowDialog	Bool	Optional. Ermöglicht die Anzeige des integrierten Mail-Editors für ein (nicht-sichtbares) RecordSet Objekt.

		Wird dieser Parameter auf True gesetzt hat der Silent-Parameter keine Auswirkungen mehr. Es erfolgt zudem die Anzeige eines Fortschrittsdialogs. Wird der Silent-Parameter auf False gesetzt und ein RecordSet-Objekt verwendet, welches auf einem sichtbaren View-Objekt beruht, dann wird dieser Paramter (ShowDialog) ignoriert. Es erfolgt dann ebenfalls die Anzeige eines Fortschrittsdialogs. Standardwert: False (Keine Anzeige des Mail-Editors vor dem Mailversand) Es wird ein fully-dynamic RecordSet als Basis benötigt wenn der integrierte Mail-Editor verwendet werden soll. Weitere Informationen finden Sie unter **Änderungen und Neuerungen.**

Rückgabewert:

Bool

Wert	Beschreibung
True	Der Versand war erfolgreich **und** es waren keine Datensätze mit leeren E-Mail-Feldern enthalten.
False	Der Versand ist fehlgeschlagen **oder** es war mindestens ein Datensatz mit leerem E-Mail-Feld enthalten.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim sTemplatePath : sTemplatePath = "%PRJDIR%\Mailvorlagen\Kontakte - Info
Teambuilding Hochseilgarten.mtpx"
Dim sErrorTagFilePath : sErrorTagFilePath = "%PRJDIR%\Serienmailings\Manueller
Filter.tag"
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SendBulkMail(sTemplateP
ath, sErrorTagFilePath, 0)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

string templatePath = @"%PRJDIR%\Mailvorlagen\Kontakte - Info Teambuilding
Hochseilgarten.mtpx";
string errorTagFilePath = @"%PRJDIR%\Serienmailings\Manueller Filter.tag";
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SendBulkMail(templatePa
th, errorTagFilePath, false);
```

SendMailDialog

Beschreibung:

Sendet eine E-Mail. Der Mailversand über diese Methode verhält sich analog zum Kontextmenü.

Hinweis: Ein etwaiges Auto-Protokoll sowie die automatische E-Mail-Ablage werden nur ausgeführt, wenn der integrierte Mail-Editor verwendet wird.

Parameter:

Parametername	Typ	Beschreibung
EmailAddress	String	E-Mail-Adresse(n) eines oder mehrerer Empfänger(s). Bei mehreren E-Mail-Adressen müssen diese durch Semikolon getrennt werden. Die Empfangsart pro E-Mail-Adresse kann über Präfixe bestimmt werden. Wenn kein Präfix angegeben ist, wird "TO:" angenommen, dieser kann aber optional auch angegeben werden. Für den Versand von E-Mail-Kopien können zusätzlich die Präfixe "CC:" und/oder "BCC:" verwendet werden; beachten Sie hierbei, dass jede E-Mail-Adresse einen eigenen Präfix benötigt. Beispiel 1: TO:maier@combit.net;CC:mueller@combit.net; CC:schmidt@combit.net;BCC:fischer@combit.net; Beispiel 2: maier@combit.net;weber@combit.net;
Subject	String	Betreff der E-Mail.
Contents	String	Text der E-Mail.
Files	String	Optional. Kann eine Liste von E-Mail-Anhängen beinhalten. Diese müssen dann per Semikolon separiert übergeben werden. z. B. C:\MyFiles\Report.pdf;C:\Info.doc Hinweis: Wenn in den allgemeinen Einstellungen "Mail per Kontextmenü über" auf "auf 'mailto:' registrierten Mail-Client" gestellt ist, hat dieser Parameter keine Auswirkung.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim sEmail : sEmail = "soleil@luna-aventuras.net"
Dim sSubject : sSubject = "Betreff der E-Mail"
Dim sContents : sContents = "Inhalt der E-Mail"
Dim sFiles : sFiles = "C:\temp\Angebot.pdf;C:\temp\Angebot2.pdf"
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SendMailDialog(sEmail,
sSubject, sContents, sFiles)
```

Beispiel C#-Script:

```
string email = "soleil@luna-aventuras.net";
string subject = "Betreff der E-Mail";
string contents = "Inhalt der E-Mail";
string files = @"C\temp\Angebot.pdf;C\temp\Angebot2.pdf";
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SendMailDialog(email,
subject, contents, files);
```

SendMAPIMail

Beschreibung:

Sendet eine E-Mail über die MAPI-Schnittstelle.

Hinweis: Die E-Mail wird nicht automatisch abgelegt. Das Auto-Protokoll wird nicht ausgelöst.

Parameter:

Parametername	Typ	Beschreibung
EmailAddress	String	E-Mail-Adresse(n) eines oder mehrerer Empfänger(s). Bei mehreren E-Mail-Adressen müssen diese durch Semikolon getrennt werden. Die Empfangsart pro E-Mail-Adresse kann über Präfixe bestimmt werden. Wenn kein Präfix angegeben ist, wird "TO:" angenommen, dieser kann aber optional auch angegeben werden. Für den Versand von E-Mail-Kopien können zusätzlich die Präfixe "CC:" und/oder "BCC:" verwendet werden; beachten Sie hierbei, dass jede E-Mail-Adresse einen eigenen Präfix benötigt. Beispiel 1: TO:maier@combit.net;CC:mueller@combit.net; CC:schmidt@combit.net;BCC:fischer@combit.net; Beispiel 2: maier@combit.net;weber@combit.net;
Subject	String	Betreff der E-Mail.
Contents	String	Text der E-Mail.
Files	String	Kann eine Liste von E-Mail-Anhängen beinhalten. Diese müssen dann per Semikolon separiert übergeben werden. z. B. C:\MyFiles\Report.pdf;C:\Info.doc
ShowDialog	Long	Anzeige eines Bestätigungsdialoges. 0 = Dialog wird nicht angezeigt. 1 = Dialog wird angezeigt.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim sEmail : sEmail = "soleil@luna-aventuras.net"
Dim sSubject : sSubject = "Betreff der E-Mail"
Dim sContents : sContents = "Inhalt der E-Mail"
Dim sFiles : sFiles = "C:\temp\Angebot.pdf;C:\temp\Angebot2.pdf"
Dim bShowDialog : bShowDialog = True
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SendMAPIMail(sEmail,
sSubject, sContents, sFiles, bShowDialog)
```

Beispiel C#-Script:

```
string email = "soleil@luna-aventuras.net";
string subject = "Betreff der E-Mail";
string contents = "Inhalt der E-Mail";
string files = @"C\temp\Angebot.pdf;C\temp\Angebot2.pdf";
bool showDialog = true;
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SendMAPIMail(email,
subject, contents, files, showDialog);
```

SetFilter

Beschreibung:

Erstellt einen Filter auf Basis des übergebenen Filter-Ausdruckes. Die Methode entspricht technisch **Filtern > Allgemein**. Übergeben Sie einen leeren String ("""), so wird ein bestehender Filter aufgehoben.

Der Filter kann immer nur für die jeweilige Ansicht/RecordSet ausgeführt werden. Für einen relationalen Filterausdruck können Sie entweder einen abgespeicherten Filter (**SetFilterByName**) oder **SetFilterDirectSQL** verwenden.

Wichtig: Das **RecordSet**-Objekt muss auf einem **View** basieren! Es sollten zuvor alle etwaig geholten **Record**-Objekte auf Nothing gesetzt werden, keinesfalls dürfen sie anschließend noch verwendet werden.

Parameter:

Parametername	Typ	Beschreibung
FilterContents	String	Das Suchkriterium.
SearchType	Long	0 Normal 1 Exakt 2 Wildcard 3 Phonetisch 4 Enthält
CaseSensitive	Long	Gross-/Kleinschreibung beachten: 0 nicht CaseSensitive 1 CaseSensitive
bUseCurrentQuery (Optional)	Bool	True: Der Filter basiert auf dem aktuell bestehenden Filter. False (Voreinstellung): Es wird ein neuer Filter erstellt.

Rückgabewert:

Bool

Wert	Beschreibung
True	Der Filterausdruck wurde korrekt ausgeführt. Alle gefilterten Datensätze sind ab diesem Zeitpunkt im verwendeten **RecordSet**-Objekt enthalten.
False	Filter konnte nicht ausgeführt werden.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet
Call oRecordSet.SetFilter("upper(""Contacts"".""AccountMngr"") =
upper(N'LFrisch')", False)
Call cRM.DialogMessageBox("Es befinden sich " & oRecordSet.RecCount & " Datensätze
im Filter.", "RecordSet.SetFilter", vbOKOnly)
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

RecordSet recordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet();
recordSet.SetFilter(@"upper(""Contacts"".""AccountMngr"") = upper(N//LFrisch//)",
false);
cRM.DialogMessageBox("Es befinden sich " + recordSet.RecCount.ToString() + "
Datensätze im Filter.", "RecordSet.SetFilter", 0);
recordSet.Dispose();
```

SetFilterByCurrentSortOrder

Beschreibung:

Erzeugt einen Filter basierend auf den Feldern der aktuellen Sortierung

Wichtig: Das **RecordSet**-Objekt muss auf einem **View** basieren! Es sollten zuvor alle etwaig geholten **Record**-Objekte auf Nothing gesetzt werden, keinesfalls dürfen sie anschließend noch verwendet werden.

Parameter:

Parametername	Typ	Beschreibung
FilterContents	String	Das Suchkriterium.
SearchType	Long	0 Normal 1 Exakt 2 Wildcard 3 Phonetisch 4 Enthält
CaseSensitive	Long	Gross-/Kleinschreibung beachten: 0 nicht CaseSensitive 1 CaseSensitive
bUseCurrentQuery (Optional)	Bool	True: Der Filter basiert auf dem aktuell bestehenden Filter. False (Voreinstellung): Es wird ein neuer Filter erstellt.

Rückgabewert:

Bool

Wert	Beschreibung
True	Der Filterausdruck wurde korrekt ausgeführt. Alle gefilterten Datensätze sind ab diesem Zeitpunkt im verwendeten **RecordSet**-Objekt enthalten.
False	Filter konnte nicht ausgeführt werden.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByCurrentSortO
rder("GmbH", 4, 0, False)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByCurrentSortO
rder("GmbH", 4, 0, false);
```

SetFilterByFieldName

Beschreibung:

Setzt einen Filter anhand eines Feldinhalts. Beachten Sie auch gerne die Informationen zur empfohlenen Vorgehensweise im Kapitel **Ausführen eines Filters.**

Wichtig: Es sollten zuvor alle etwaig geholten **Record**-Objekte auf Nothing gesetzt werden, keinesfalls dürfen sie anschließend noch verwendet werden. Werden bei einer Suche über mehrere Felder keine Datensätze gefunden, so wird automatisch eine zweite Suche nach dem kompletten Leerzeichen-separierten Suchwert durchgeführt, jedoch lediglich im ersten Suchfeld.

Parameter:

Parametername	Typ	Beschreibung
FieldName	String	Name des Feldes bzw. mehrere Felder, welche mittels "+" getrennt angegeben werden. Hierbei können auch 1:1-relational oder 1:1:1 relational verknüpfte Felder angegeben werden. Bei 1:1(:1) verknüpften relationalen Feldern muss der vollständige relationale Pfad verwendet werden, z. B. "CompanyID.Firmen.ID.Company".

Contents	String	Suchbegriff(e). Suchbegriffe für mehrere Felder werden per Leerzeichen-Separator übergeben. Wenn ein Suchbegriff ein Leerzeichen enthalten soll, dann werden die Suchbegriffe mit einem Komma getrennt. Das Komma kann zudem verwendet werden, um ein Suchfeld (siehe Parameter **FieldName**) auszulassen.
SearchType	Long	0 Normal 1 Exakt 2 Wildcard 3 Phonetisch 4 Enthält
CaseSensitive	Long	Gross-/Kleinschreibung beachten: 0 nicht CaseSensitive 1 CaseSensitive
bUseCurrentQuery (Optional)	Bool	True: Der Filter basiert auf dem aktuell bestehenden Filter. False (Voreinstellung): Es wird ein neuer Filter erstellt.

Rückgabewert:

Bool

Wert	Beschreibung
True	Filter konnte erstellt werden. Alle gefilterten Datensätze sind ab diesem Zeitpunkt im verwendeten **RecordSet**-Objekt enthalten.
False	Filter konnte nicht ausgeführt werden.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByFieldName("C
ontactID.Kontakte.ID.CompanyID.Firmen.ID.Company" + "ContactID.Kontakte.ID.Name",
"Albatros Flug", 0, 0)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByFieldName("C
ontactID.Kontakte.ID.CompanyID.Firmen.ID.Company" + "ContactID.Kontakte.ID.Name",
"Albatros Flug", 0, 0);
```

SetFilterByGeo

Beschreibung:

Setzt einen Filter anhand eines bestimmten geographischen Punktes. Beachten Sie auch gerne die Informationen zur empfohlenen Vorgehensweise im Kapitel **Ausführen eines Filters.**

Parameter:

Parametername	Typ	Beschreibung
GeoFilterType	Long	Gibt den Suchtyp an: 1 Berechnung anhand der Luftlinie
FilterOperation	Long	Gibt den Filtertyp an: 1 Im Umkreis von 2 Nicht im Umkreis von
NotGeocodedRecordAction	Long	Wie sollen für die Umkreissuche unkodierte Datensätze behandelt werden: 1 Visuell anzeigen

		2 Datensätze geokodieren 3 Filter ohne Geokodierung durchführen 4 Filtervorgang abbrechen
MultipleResultAction	Long	1 Auswahldialog anzeigen, wenn mehrere Adressen für den Umkreispunkt gefunden wurden 2 Immer den ersten Datensatz verwenden, wenn mehrere Adressen für den Umkreispunkt gefunden wurden
AliasName	String	Name der Feldliste der Adresse
Address	String	Adresse des Umkreispunkt Beispiel: Bücklestr. 3-5|78467|Konstanz|DE (Straße|PLZ|Ort|Land) Wichtig: Die Angabe einer Straße ist optional. Geben Sie die Straße nicht an, so muss das entsprechende Pipe-Zeichen trotzdem erhalten bleiben. Beispiel: |78462|Konstanz|DE
Distance	Long	Umkreis in Kilometern, in dem gesucht werden soll
bUseCurrentQuery (Optional)	Bool	True: Der Filter basiert auf dem aktuell bestehenden Filter. False (Voreinstellung): Es wird ein neuer Filter erstellt.

Rückgabewert:

Bool

Wert	Beschreibung
True	Filter konnte erstellt werden. Alle gefilterten Datensätze sind ab diesem Zeitpunkt im verwendeten **RecordSet**-Objekt enthalten.
False	Filter konnte nicht ausgeführt werden.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByGeo(1,
1, 1, 1, "Kontakte_Adresse", "Bücklestraße|3-5|Konstanz|DE", 100, False)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByGeo(1, 1, 1,
1, "Kontakte_Adresse", "Bücklestraße|3-5|Konstanz|DE", 100, false);
```

SetFilterByName

Beschreibung:

Führt einen abgespeicherten Filterausdruck über den Namen aus. Dieser kann über den Menüpunkt **Filtern > Filterausdrücke verwalten** definiert werden. Beachten Sie auch gerne die Informationen zur empfohlenen Vorgehensweise im Kapitel **Ausführen eines Filters.**

Wichtig: Es sollten zuvor alle etwaig geholten **Record**-Objekte auf Nothing gesetzt werden, keinesfalls dürfen sie anschließend noch verwendet werden.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Name des abgespeicherten Filters.

		Hinweis: Bitte beachten Sie, dass nicht die Bezeichnung des Filterausdrucks verwendet wird, sondern der "Name für Scripte/Workflows". Alternativ kann der vollständige Pfad zu einer .crmshare-Datei, welche einen Filterausdruck beinhaltet, angegeben werden. Diese Datei kann über die Funktion „Filter teilen“ erstellt werden.
bUseCurrentQuery (Optional)	Bool	True: Der Filter basiert auf dem aktuell bestehenden Filter. False (Voreinstellung): Es wird ein neuer Filter erstellt.

Rückgabewert:

Bool

Wert	Beschreibung
True	Filter konnte erstellt werden. Alle gefilterten Datensätze sind ab diesem Zeitpunkt im verwendeten **RecordSet**-Objekt enthalten.
False	Filter konnte nicht ausgeführt werden.

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByName("Name
des abgespeicherten Filters", False)
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByName("Name
des abgespeicherten Filters", false);
```

SetFilterByPrimaryKey

Beschreibung:

Setzt einen Filter basierend auf dem Primärschlüssel des **RecordSets**.

Diese Methode funktioniert nur, wenn ein Primärschlüssel vorhanden ist, der genau aus einem Feld besteht. Beachten Sie auch gerne die Informationen zur empfohlenen Vorgehensweise im Kapitel **Ausführen eines Filters.**

Wichtig: Es sollten zuvor alle etwaig geholten **Record**-Objekte auf Nothing gesetzt werden, keinesfalls dürfen sie anschließend noch verwendet werden.

Parameter:

Parametername	Typ	Beschreibung
PrimaryKeyContent	String	Inhalt des Primärschlüsselfeldes.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByPrimaryKey("
Inhalts des Primärschlüsselfeldes")
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByPrimaryKey("
Inhalts des Primärschlüsselfeldes");
```

SetFilterByTagFile

Beschreibung:

Setzt und aktiviert einen manuellen Filter basierend auf einer Datei für eine manuelle Datensatzauswahl (.tag-Datei). Beachten Sie auch gerne die Informationen zur empfohlenen Vorgehensweise im Kapitel **Ausführen eines Filters.**

Wichtig: Diese Methode steht nur für ein **RecordSet** zur Verfügung, welches auf einer visuellen Ansicht (**ActiveView**) basiert.

Es sollten zuvor alle etwaig geholten **Record**-Objekte auf Nothing gesetzt werden, keinesfalls dürfen sie anschließend noch verwendet werden.

Parameter:

Parametername	Typ	Beschreibung
FileName	String	Dateipfad der .tag-Datei.
bUseCurrentQuery (Optional)	Bool	True: Der Filter basiert auf dem aktuell bestehenden Filter. False (Voreinstellung): Es wird ein neuer Filter erstellt.
nManFltCombine-Option	Long	Optional Bestimmt wie die Datensatzauswahl mit einem bereits bestehenden manuellen Filter kombiniert werden soll. 1 = Logisches UND: Wenn ein Datensatz in mindestens einem Filter ausgeschlossen ist, wird er auch im Ergebnisfilter ausgeschlossen. 2 = Logisches ODER: Wenn ein Datensatz in mindestens einem Filter eingeschlossen ist, wird er auch im Ergebnisfilter eingeschlossen. 3 = Neuere: Bei Datensätzen, die sowohl in der bestehenden, als auch in der zu ladenden Datensatzauswahl vorkommen, wird die Datensatzauswahl aus dem zu ladenden Filter übernommen. 4 = Ältere: Bei Datensätzen, die sowohl in der bestehenden, als auch in der zu ladenden Datensatzauswahl vorkommen, wird die Datensatzauswahl aus dem bestehenden Filter übernommen. 5 = Überschreiben: Eine bestehende Datensatzauswahl wird mit der zu ladenden Datensatzauswahl überschrieben.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByTagFile("Pfa
d zur .tag-Datei", False)
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterByTagFile("Pfa
d zur .tag-Datei", false);
```

SetFilterDirectSQL

Beschreibung:

Erstellt einen Filter auf Basis des übergebenen freien Filterausdruckes. Beachten Sie auch gerne die Informationen zur empfohlenen Vorgehensweise im Kapitel **Ausführen eines Filters.**

Hinweis: In der Abfrage muss generell im SELECT-Teil die Primärschlüsselspalte der zum **RecordSet** gehörenden Tabelle selektiert werden, der restliche Aufbau der Abfrage ist beliebig.

Wichtig: Es sollten zuvor alle etwaig geholten **Record**-Objekte auf Nothing gesetzt werden, keinesfalls dürfen sie anschließend noch verwendet werden.

Parameter:

Parametername	Typ	Beschreibung
sSQLQuery	String	Freier SQL Filterausdruck.
bUseCurrentQuery (Optional)	Bool	True: Der Filter basiert auf dem aktuell bestehenden Filter. False (Voreinstellung): Es wird ein neuer Filter erstellt.

Rückgabewert:

Bool

Wert	Beschreibung
True	Filter konnte erstellt werden. Alle gefilterten Datensätze sind ab diesem Zeitpunkt im verwendeten **RecordSet**-Objekt enthalten.
False	Filter konnte nicht ausgeführt werden.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterDirectSQL("SEL
ECT Contacts.ID FROM Contacts WHERE Salutation = 'Herrn'", True)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.SetFilterDirectSQL("SEL
ECT Contacts.ID FROM Contacts WHERE Salutation = //Herrn//", true);
```

Synchronize

Beschreibung:

Startet den Abgleich-Assistenten und führt das übergebene Benutzerformat aus. Es werden alle Einstellungen dieses Formates verwendet.

Wenn ein Format zum automatischen Ausführen übergeben wird, oder die Methoden für einen unsichtbaren **RecordSet** aufgerufen werden, dann ist ein Abgleich von unbekannten Codefeldern – im Gegensatz zum interaktiven Abgleich – nicht möglich, diese werden dann ignoriert.

Parameter:

Parametername	Typ	Beschreibung
Format	String	Name des Abgleichformates.

		Es kann sich dabei um einen kompletten Dateipfad auf eine .itp Format-Datei handeln, alternativ um den Formatnamen, so wie er im Assistenten dargestellt wird, dabei werden zuerst die benutzerspezifischen Formate durchsucht, anschließend die globalen Formate. Bitte beachten Sie die Großkleinschreibung.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.Synchronize("Name des Abgleichformates")
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.Synchronize("Name des Abgleichformates");
```

3.39 RecurrencePattern Objekt

Dieses Objekt ähnelt dem von Outlook verwendeten Objekt in seiner Schnittstelle. Es dient zur Speicherung der Eigenschaften bzw. Einstellungen von Serienterminen.

3.39.1 Eigenschaften

DayOfMonth

Beschreibung:

Setzt den Tag des Monats (1-31) an dem ein Termin auftritt. Wird bei olRecursMonthly(2) und olRecursYearly(5) verwendet.

Typ:

Long

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

DayOfWeekMask

Beschreibung:

Gibt die Wochentage für wiederkehrende Termine an. Wird bei olRecursDaily(0), olRecursMonthNth(3), olRecursWeekly(1) und olRecursYearNth(6) verwendet.

Typ:

Long

Folgende Werte können miteinander ver'oder't werden:

Konstante	Wert	Beschreibung
OlSunday	1	Sonntag
OlMonday	2	Montag
OlTuesday	4	Dienstag
OlWednesday	8	Mittwoch
OlThursday	16	Donnerstag
OlFriday	32	Freitag
OlSaturday	64	Samstag

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfWeekMask = 2
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfWeekMask = 2;
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

Duration

Beschreibung:

Dauer eines Termins in Minuten.

Typ:

Long

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
```

```
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

EndTime

Beschreibung:

Liefert/setzt die Endzeit für ein Serienmuster.

Typ:

Date

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

Instance

Beschreibung:

Dieser Wert ist nur für olRecursMonthNth(3) und olRecursYearNth(6) gültig und ermöglicht die Definition eines Serienmusters, das nur für das n-te Vorkommen gilt, wie z. B. ein Muster der Form "der zweite Sonntag im März".

Die Zahl wird numerisch festgelegt: 1 für das erste, 2 für das zweite und so weiter bis 5 für das letzte.

Typ:

Long

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

Interval

Beschreibung:

Gibt die Anzahl von Einheiten zwischen dem Auftreten zweier Serientermine an, also z. B. Interval = 2 bei Terminen alle zwei Wochen.

Wird bei olRecursDaily(0), olRecursWeekly(1), olRecursMonthly(2), olRecursMonthNth(3), verwendet.

Typ:

Long

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

MonthOfYear

Beschreibung:

Gibt den Monat im Jahr (1-12) für einen Serientermin an.

Wenn diese Eigenschaft z. B. auf 5 gesetzt wird und RecurrenceType olRecursYearly (5) ist, so wird dieser Termin jedes Jahr im Mai auftreten.

Typ:

Long

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

NoEndDate

Beschreibung:

Ist True, wenn ein Serientermin kein Enddatum besitzt.

Typ:

Bool

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
```

```
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

PatternEndDate

Beschreibung:

Enddatum eines Serientermins. Die **NoEndDate**-Eigenschaft wird dabei zurückgesetzt.

Typ:

Date

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

PatternStartDate

Beschreibung:

Anfangsdatum eines Serientermins.

Typ:

Date

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
```

```
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

RecurrenceType

Beschreibung:

Gibt die Art des Serientermins an, die Konstanten sind analog zur Outlook-Schnittstelle.

Konstante	Wert	Beschreibung
olRecursDaily	0	Tägliche Wiederholung
olRecursWeekly	1	Wöchentliche Wiederholung
olRecursMonthly	2	Monatliche Wiederholung
olRecursMonthNth	3	Monatliche Wiederholung spezieller Art
olRecursYearly	5	Jährliche Wiederholung
olRecursYearNth	6	Jährliche Wiederholung spezieller Art

Typ:

Long

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

Hinweis: Diese Eigenschaft muss als erste Eigenschaft des RecurrencePattern-Objektes gesetzt werden.

StartTime

Beschreibung:

Startzeit für ein Serienmuster.

Typ:

Date

Beispiel VBScript:

```
Dim oRecurrencePattern : Set oRecurrencePattern = oAppointment.RecurrencePattern
oRecurrencePattern.RecurrenceType = 2 ' Monatliche Wiederholung
oRecurrencePattern.DayOfMonth = 10
oRecurrencePattern.Duration = 60
oRecurrencePattern.EndTime = 12
oRecurrencePattern.Interval = 1
oRecurrencePattern.NoEndDate = False
oRecurrencePattern.StartTime = 10
oRecurrencePattern.PatternStartDate = Now()
oRecurrencePattern.PatternEndDate = DateAdd("yyyy", 1, Now())
```

Beispiel C#-Script:

```
RecurrencePattern recPattern = appointment.RecurrencePattern;
recPattern.RecurrenceType = 2; // Monatliche Wiederholung
recPattern.DayOfMonth = 10;
recPattern.Duration = 60;
recPattern.EndTime = System.DateTime.Now.AddHours(1);
recPattern.Interval = 1;
recPattern.NoEndDate = false;
recPattern.StartTime = System.DateTime.Now;
recPattern.PatternStartDate = System.DateTime.Now;
recPattern.PatternEndDate = System.DateTime.Now.AddYears(1);

recPattern.Dispose();
```

3.40 Relation Objekt

Hinweis: Aus Sicherheitsgründen ist der Zugriff auf **Relation** Objekte von Ansichten, auf die der aktuelle Benutzer keine Zugriffsrechte besitzt, nicht möglich. Alle diese **Relation** Objekte werden in der übergeordneten Collection (**ListRelations**) für die Eigenschaften/Methoden *Count* und *Item* nicht angeboten. Bei einem versuchten Direktzugriff per *ItemByName* wird kein Objekt zurückgegeben – es erfolgt zudem eine Fehlerausgabe auf das Debug-Tool Debwin.

3.40.1 Eigenschaften

Alias

Beschreibung:

Liefert den Relationsalias zurück.

Typ:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRelation : Set oRelation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID")

Dim sAlias : sAlias = oRelation.Alias
Dim bCascadeOnDelete : bCascadeOnDelete = oRelation.CascadeOnDelete
Dim sFieldName : sFieldName = oRelation.FieldName
Dim sForeignViewName : sForeignViewName = oRelation.ForeignViewName
Dim nRelType : nRelType = oRelation.RelType

Set oRelation = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution
```

```
Relation relation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID");

string alias = relation.Alias;
bool cascadeOnDelete = relation.CascadeOnDelete;
string fieldName = relation.FieldName;
string foreignViewName = relation.ForeignViewName;
long relType = relation.RelType;

relation.Dispose();
```

CascadeOnDelete, read-only

Beschreibung:

Liefert zurück, ob der kaskadierte Papierkorb für diese Relation aktiviert wurde.

Typ:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRelation : Set oRelation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID")

Dim sAlias : sAlias = oRelation.Alias
Dim bCascadeOnDelete : bCascadeOnDelete = oRelation.CascadeOnDelete
Dim sFieldName : sFieldName = oRelation.FieldName
Dim sForeignViewName : sForeignViewName = oRelation.ForeignViewName
Dim nRelType : nRelType = oRelation.RelType

Set oRelation = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Relation relation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID");

string alias = relation.Alias;
bool cascadeOnDelete = relation.CascadeOnDelete;
string fieldName = relation.FieldName;
string foreignViewName = relation.ForeignViewName;
long relType = relation.RelType;

relation.Dispose();
```

FieldName

Beschreibung:

Liefert den Namen des Feldes, von dem die Relation wegführt.

Typ:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution
```

```
Dim oRelation : Set oRelation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID")

Dim sAlias : sAlias = oRelation.Alias
Dim bCascadeOnDelete : bCascadeOnDelete = oRelation.CascadeOnDelete
Dim sFieldName : sFieldName = oRelation.FieldName
Dim sForeignViewName : sForeignViewName = oRelation.ForeignViewName
Dim nRelType : nRelType = oRelation.RelType

Set oRelation = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Relation relation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID");

string alias = relation.Alias;
bool cascadeOnDelete = relation.CascadeOnDelete;
string fieldName = relation.FieldName;
string foreignViewName = relation.ForeignViewName;
long relType = relation.RelType;

relation.Dispose();
```

ForeignViewFieldName

Beschreibung:

Liefert den Namen des Feldes, auf das die Relation zeigt (Fremdschlüssel).

Typ:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRelation : Set oRelation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID")

Dim sAlias : sAlias = oRelation.Alias
Dim bCascadeOnDelete : bCascadeOnDelete = oRelation.CascadeOnDelete
Dim sFieldName : sFieldName = oRelation.FieldName
Dim sForeignViewName : sForeignViewName = oRelation.ForeignViewName
Dim nRelType : nRelType = oRelation.RelType

Set oRelation = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Relation relation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID");

string alias = relation.Alias;
bool cascadeOnDelete = relation.CascadeOnDelete;
string fieldName = relation.FieldName;
string foreignViewName = relation.ForeignViewName;
long relType = relation.RelType;

relation.Dispose();
```

ForeignViewName

Beschreibung:

Liefert den Namen der Ansicht, zu der die Relation hinführt.

Typ:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRelation : Set oRelation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID")

Dim sAlias : sAlias = oRelation.Alias
Dim bCascadeOnDelete : bCascadeOnDelete = oRelation.CascadeOnDelete
Dim sFieldName : sFieldName = oRelation.FieldName
Dim sForeignViewName : sForeignViewName = oRelation.ForeignViewName
Dim nRelType : nRelType = oRelation.RelType

Set oRelation = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Relation relation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID");

string alias = relation.Alias;
bool cascadeOnDelete = relation.CascadeOnDelete;
string fieldName = relation.FieldName;
string foreignViewName = relation.ForeignViewName;
long relType = relation.RelType;

relation.Dispose();
```

RelType

Beschreibung:

Liefert den Typ der Relation zurück.

Typ:

Long

Wert	Beschreibung
1	1:1 Relation
2	1:N Relation
3	N:M Relation

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim oRelation : Set oRelation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID")

Dim sAlias : sAlias = oRelation.Alias
Dim bCascadeOnDelete : bCascadeOnDelete = oRelation.CascadeOnDelete
Dim sFieldName : sFieldName = oRelation.FieldName
Dim sForeignViewName : sForeignViewName = oRelation.ForeignViewName
Dim nRelType : nRelType = oRelation.RelType
```

```
Set oRelation = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Relation relation =
cRM.CurrentProject.ActiveViews.ActiveView.Config.Relations.ItemByName("ID.Aktivitä
ten.CompanyID");

string alias = relation.Alias;
bool cascadeOnDelete = relation.CascadeOnDelete;
string fieldName = relation.FieldName;
string foreignViewName = relation.ForeignViewName;
long relType = relation.RelType;

relation.Dispose();
```

3.41 SQLShell Objekt

Führt eine SQL-Anweisung direkt aus und liefert die Anzahl Datensätze zurück, die davon betroffen sind.

Das Objekt kann nicht von außen instanziert werden und steht daher in dieser Form nur in Scripten zur Verfügung, die innerhalb des Programms aufgerufen werden.

3.41.1 Eigenschaften

LastError, read-only

Beschreibung:

Liefert ein OLEError-Objekt zurück.

Typ:

OLEError

Wert	Beschreibung
0	Kein Fehler aufgetreten.
1	Rückgabe der Fehlermeldung des Servers.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim nResult : nResult = SQLShell.ExecuteCommandRaw("UPDATE ""Companies"" SET
""ABC"" = 'A'")
If (nResult = -1) Then
    Call cRM.DialogMessageBox("Es ist ein Fehler aufgetreten." & vbCrLf &
SQLShell.LastError.ErrorCode & ": " & SQLShell.LastError.ErrorText, "SQLShell",
vbOkOnly)
Else
    Call cRM.DialogMessageBox(nResult & " Datensätze betroffen.", "SQLShell",
vbOkOnly)
End If
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

long result = SQLShell.ExecuteCommandRaw("UPDATE \"Companies\" SET \"ABC\" =
'A'");

if (result == -1)
{
    cRM.DialogMessageBox("Es ist ein Fehler aufgetreten." +
System.Environment.NewLine + SQLShell.LastError.ErrorCode.ToString() + ": " +
SQLShell.LastError.ErrorText, "SQLShell", 0);
}
else
```

```
{
    cRM.DialogMessageBox(result & " Datensätze betroffen.", "SQLShell", 0);
}
```

3.41.2 Methoden

ExecuteCommandRaw

Beschreibung:

Führt eine SQL-Anweisung direkt aus. Es wird dabei die aktuelle Verbindung zur Datenbank der Solution verwendet.

Parameter:

Parametername	Typ	Beschreibung
SQLCommand	String	SQL-Anweisung

Hinweis: Es macht keinen Sinn, hierüber eine SELECT Anweisung abzusetzen, da man hierüber das Ergebnis nicht abfragen/durchlaufen kann. Einsatzzweck sind UPDATE, INSERT, DELETE Anweisungen oder ggf. das Aufrufen von Stored Procedures.

Rückgabewert:

Long

Wert	Beschreibung
>= 0	Anzahl der betroffenen Datensätze bei UPDATE oder DELETE Anweisungen.
0	Rückgabewert bei INSERT Anweisungen oder dem Aufruf von Stored Procedures.
-1	Fehler bzw. Rückgabewert bei Ausführung einer SQL-Abfrage ohne Datenmanipulation (z. B. mittels CREATE/DROP FUNCTION-Anweisungen) und bei gleichzeitigem Rückgabewert von 0 für die Eigenschaft SQLShell.LastError.ErrorCode.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

Dim nResult : nResult = SQLShell.ExecuteCommandRaw("UPDATE ""Companies"" SET
""ABC"" = 'A'")
If (nResult = -1) Then
    Call cRM.DialogMessageBox("Es ist ein Fehler aufgetreten." & vbCrLf &
SQLShell.LastError.ErrorCode & ": " & SQLShell.LastError.ErrorText, "SQLShell",
vbOkOnly)
Else
    Call cRM.DialogMessageBox(nResult & " Datensätze betroffen.", "SQLShell",
vbOkOnly)
End If
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Firmen-Ansicht einer combit_Large-Solution

long result = SQLShell.ExecuteCommandRaw("UPDATE \"Companies\" SET \"ABC\" =
'A'");

if (result == -1)
{
    cRM.DialogMessageBox("Es ist ein Fehler aufgetreten." +
System.Environment.NewLine + SQLShell.LastError.ErrorCode.ToString() + ": " +
SQLShell.LastError.ErrorText, "SQLShell", 0);
}
else
```

```
{
    cRM.DialogMessageBox(result & " Datensätze betroffen.", "SQLShell", 0);
}
```

3.42 timemanager Objekt

3.42.1 Eigenschaften

ActiveView

Beschreibung:

Legt die aktuelle Ansicht fest oder gibt diese zurück. Es muss daher eine Konstante aus dem Wertebereich der **TMViewConstants** übergeben werden.

Konstanten der TMViewConstants:

Konstante	Wert	Beschreibung
TM_VIEW_DAY	0	Tagesansicht
TM_VIEW_WEEK	1	Wochenansicht
TM_VIEW_MONTH	2	Monatsansicht
TM_VIEW_APPLIST	3	Terminliste
TM_VIEW_TODO	4	Aufgaben

Typ:

TMViewConstants

Beispiel VBScript:

```
If (cRM.CurrentProject.timemanager.ActiveView = 0) Then
    cRM.CurrentProject.timemanager.ActiveView = 3
End If
```

Beispiel C#-Script:

```
if (cRM.CurrentProject.TimeManager.ActiveView == 0)
{
    cRM.CurrentProject.TimeManager.ActiveView = 3;
}
```

Appointments, read-only

Beschreibung:

Gibt in einem Objekt vom Typ **Appointments** (s.u.) eine Sammlung aller Termine zurück.

Typ:

Appointments

Beispiel VBScript:

```
Dim oAppointments : Set oAppointments =
cRM.CurrentProject.timemanager.Appointments
' ...
Set oAppointments = Nothing
```

Beispiel C#-Script:

```
Appointments appointments = cRM.CurrentProject.TimeManager.Appointments;
// ...
appointments.Dispose();
```

CurrentUser

Beschreibung:

Name des aktuellen Benutzers.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Der Name des aktuell angemeldeten Benutzers lautet: " &
cRM.CurrentProject.timemanager.CurrentUser, "timemanager.CurrentUser", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Der Name des aktuell angemeldeten Benutzers lautet: " +
cRM.CurrentProject.TimeManager.CurrentUser, "TimeManager.CurrentUser", 0);
```

HostID

Beschreibung:

Eindeutige ID der aktuellen Host Applikation. Dieser Wert korrespondiert zu der HostID-Eigenschaft der ToDo- und Appointment-Objekte, s.u.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Die derzeitige HostID lautet: " &
cRM.CurrentProject.timemanager.HostID, "timemanager.HostID", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Die derzeitige HostID lautet: " +
cRM.CurrentProject.TimeManager.HostID, "TimeManager.HostID", 0);
```

ShowReminder

Beschreibung:

Diese Eigenschaft ermöglicht es (bei False) das Erscheinen von Termin-Erinnerungen zu unterdrücken.

Typ:

Bool

Beispiel VBScript:

```
Dim bShowReminder : bShowReminder = cRM.CurrentProject.timemanager.ShowReminder
```

Beispiel C#-Script:

```
bool showReminder = cRM.CurrentProject.TimeManager.ShowReminder;
```

ToDos

Beschreibung:

Gibt in einem Objekt vom Typ **ToDos** (s.u.) eine Sammlung aller Aufgaben zurück.

Typ:

ToDos

Beispiel VBScript:

```
Dim oToDos : Set oToDos = cRM.CurrentProject.timemanager.ToDos
' ...
```

```
Set oToDos = Nothing
```

Beispiel C#-Script:

```
ToDos todos = cRM.CurrentProject.TimeManager.ToDos;
// ...
todos.Dispose();
```

ViewDate

Beschreibung:

Gibt das aktuelle Datum der Ansichten zurück oder legt dieses fest.

Typ:

Date

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Das derzeitige Datum ist: " &
CStr(cRM.CurrentProject.timemanager.ViewDate), "timemanager.ViewDate", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Das derzeitige Datum ist: " +
cRM.CurrentProject.TimeManager.ViewDate.ToString(), "TimeManager.ViewDate", 0);
```

Visible

Beschreibung:

Setzt den Anzeigezustand der Ansicht *Termine & Aufgaben* oder gibt diesen zurück.

Typ:

Bool

Beispiel VBScript:

```
If (cRM.CurrentProject.timemanager.Visible = False) Then
    cRM.CurrentProject.timemanager.Visible = True
End If
```

Beispiel C#-Script:

```
if (cRM.CurrentProject.TimeManager.Visible == false)
{
    cRM.CurrentProject.TimeManager.Visible = true;
}
```

3.42.2 Methoden

UpdateViews

Beschreibung:

Aktualisiert alle Ansichten.

> **Wichtig**: Muss nach der Neuanlage eines Termins/einer Aufgabe für die Aktualisierung der Anzeige des neu angelegten Termins/Aufgabe aufgerufen werden!

Beispiel VBScript:

```
Call cRM.CurrentProject.timemanager.UpdateViews()
```

Beispiel C#-Script:

```
cRM.CurrentProject.TimeManager.UpdateViews();
```

3.43 ToDo Objekt

Hinweis: Wenn man die Datumswerte auf einen **ungültigen** Wert setzt (hier 0), so wird automatisch 31.12.1900 0 Uhr eingetragen.

Wenn man die Werte NICHT setzt, so wird das heutige Datum genommen.

Beispiel VBScript:

```
Dim oProject : Set oProject = cRM.CurrentProject
Dim oActiveView : Set oActiveView = cRM.CurrentProject.ActiveViews.ActiveView
Dim oViewConfig : Set oViewConfig = oActiveView.Config
Dim oRecord : Set oRecord = oActiveView.CurrentRecordSet.CurrentRecord
Dim sCurrentUserLoginName : sCurrentUserLoginName =
oProject.Users.CurrentUser.LoginName
Dim sPrimaryKeyFieldName : sPrimaryKeyFieldName = oViewConfig.PrimaryKeyFldName
Dim sRecordRefDescription : sRecordRefDescription =
oRecord.GetRecordRefDescription
Dim sRefLink : sRefLink = oProject.ID & "|" & oActiveView.Name & "|" &
oViewConfig.FamilyName & "|" & sPrimaryKeyFieldName & "|" &
oRecord.GetContentsByName(sPrimaryKeyFieldName) & "|" & sRecordRefDescription
Set oRecord = Nothing
Set oViewConfig = Nothing
Set oActiveView = Nothing
Set oProject = Nothing

Dim oToDo : Set oToDo = cRM.CurrentProject.timemanager.ToDos.Add()

oToDo.ActionData = "info@relationship-manager.net"
oToDo.ActionID = "TMMAIL"
oToDo.ActionType = "1"
oToDo.Attendees.Add("LFrisch")
oToDo.Attendees.Add("THeld")
oToDo.Body =
"{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1031{\fonttbl{\f0\fnil\fcharset0
Calibri;}}{\*\generator Riched20 10.0.22000}\viewkind4\uc1
\pard\sa200\sl276\slmult1\i\f0\fs22\lang7 Formatierter \b\i0 Inhalt/Text \b0 der
\ul Aufgabe\ulnone\par}"
' oToDo.BodyPlain = "Inhalt/Text des Termins ohne RTF-Formatierung"
oToDo.Categories.Add("Meeting")
oToDo.ChangeDate = Now()
oToDo.ChangeUser = sCurrentUserLoginName
oToDo.Contact = "Kontakt, mit dem die Aufgabe stattfindet"
oToDo.CreationDate = Now()
oToDo.CreationUser = sCurrentUserLoginName
oToDo.End = DateAdd("h", 2, Now())
oToDo.ExtUserData1 = "Erster Abschnitt von Zusatzinformationen"
oToDo.ExtUserData2 = "Zweiter Abschnitt von Zusatzinformationen"
oToDo.HostDataBase = sRefLink
oToDo.Importance = 3
oToDo.Private = False
oToDo.Reminder = True
oToDo.ReminderMinutesBeforeStart = 30
oToDo.Start = DateAdd("h", 1, Now())
oToDo.Subject = "Priorität " & oToDo.Importance & ": Meeting im Büro mit " &
sRecordRefDescription
oToDo.TimeStamp = FormatDateTime(Now(), vbGeneralDate)
oToDo.User = sCurrentUserLoginName

Dim sUniqueID : sUniqueID = oToDo.UniqueID
Call oToDo.Links.NewLink.SetLinkFromString(sRefLink)

Call oToDo.Save()
Call oToDo.ExportAsICal("C:\" & sRecordRefDescription & ".ical")
Call oToDo.Display()
Call oToDo.Remove()
```

```
Set oToDo = Nothing
```

Beispiel C#-Script:

```
Project project = cRM.CurrentProject;
View activeView = project.ActiveViews.ActiveView;
ViewConfig viewConfig = activeView.Config;
Record record = activeView.CurrentRecordSet.CurrentRecord;
string currentUserLoginName = project.Users.CurrentUser.LoginName;
string primaryKeyFieldName = viewConfig.PrimaryKeyFldName;
string recordRefDescription = record.GetRecordRefDescription();
string refLink = project.ID + "|" + activeView.Name + "|" + viewConfig.FamilyName
+ "|" + primaryKeyFieldName + "|" + record.GetContentsByName(primaryKeyFieldName)
+ "|" + recordRefDescription;
record.Dispose();
viewConfig.Dispose();
activeView.Dispose();
project.Dispose();

ToDo todo = cRM.CurrentProject.TimeManager.ToDos.Add();

todo.ActionData = "info@relationship-manager.net";
todo.ActionID = "TMMAIL";
todo.ActionType = "1";
todo.Attendees.Add("LFrisch");
todo.Attendees.Add("THeld");
todo.Body =
@"{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1031{\fonttbl{\f0\fnil\fcharset0
Calibri;}}{\*\generator Riched20 10.0.22000}\viewkind4\uc1
\pard\sa200\sl276\slmult1\i\f0\fs22\lang7 Formatierter \b\i0 Inhalt/Text \b0 der
\ul Aufgabe\ulnone\par}";
// todo.BodyPlain = "Inhalt/Text des Termins ohne RTF-Formatierung"
todo.Categories.Add("Meeting");
todo.ChangeDate = System.DateTime.Now;
todo.ChangeUser = currentUserLoginName;
todo.Contact = "Kontakt, mit dem die Aufgabe stattfindet";
todo.CreationDate = System.DateTime.Now;
todo.CreationUser = currentUserLoginName;
todo.End = System.DateTime.Now.AddHours(2);
todo.ExtUserData1 = "Erster Abschnitt von Zusatzinformationen";
todo.ExtUserData2 = "Zweiter Abschnitt von Zusatzinformationen";
todo.HostDatabase = refLink;
todo.Importance = 3;
todo.Private = false;
todo.Reminder = true;
todo.ReminderMinutesBeforeStart = 30;
todo.Start = System.DateTime.Now;
todo.Subject = "Priorität" + todo.Importance.ToString() + ": Meeting im Büro mit "
+ recordRefDescription;
todo.TimeStamp = System.DateTime.Now.ToString();
todo.User = currentUserLoginName;

string uniqueID = todo.UniqueID;
todo.Links.NewLink.SetLinkFromString(refLink);

todo.Save();
todo.ExportAsiCal(@"C:\" + recordRefDescription + ".ical");
todo.Display();
todo.Remove();

todo.Dispose();
```

3.43.1 Eigenschaften

ActionData

Beschreibung:

Enthält die Daten einer Aktion.

Typ:

String

Beispiel VBScript:

```
oToDo.ActionData = "info@relationship-manager.net"
```

Beispiel C#-Script:

```
todo.ActionData = "info@relationship-manager.net";
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Siehe auch:

ActionID

ActionID

Beschreibung:

Liefert die eindeutige ID der zugeordneten Aktion zurück oder setzt diese.

Konstante	Beschreibung
TMMAIL	Senden einer E-Mail -Benachrichtigung.
TMSHEX	Datei/Dokument ausführen bzw. öffnen.
DIAL_CRM	Telefonanruf ausführen.

Typ:

String

Beispiel VBScript:

```
oToDo.ActionID = "TMMAIL"
```

Beispiel C#-Script:

```
todo.ActionID = "TMMAIL";
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

ActionType

Beschreibung:

Art der Ausführung der Aktion.

Typ:

String

Wert	Beschreibung
0	Automatisch
1	Manuell in Erinnerungsdialog

Beispiel VBScript:

```
oToDo.ActionType = "1"
```

Beispiel C#-Script:

```
todo.ActionType = "1";
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Attendees

Beschreibung:

Liefert alle Teilnehmer der Aufgabe als Objekt vom Typ **Attendees** zurück.

Typ:

Attendees

Beispiel VBScript:

```
oToDo.Attendees.Add("LFrisch")
oToDo.Attendees.Add("THeld")
```

Beispiel C#-Script:

```
todo.Attendees.Add("LFrisch");
todo.Attendees.Add("THeld");
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Body

Beschreibung:

Setzt den Inhalt/Text der Aufgabe (mit RTF-Formatierung) oder liefert diesen zurück.

Typ:

String

Beispiel VBScript:

```
oToDo.Body =
"{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1031{\fonttbl{\f0\fnil\fcharset0
Calibri;}}{\*\generator Riched20 10.0.22000}\viewkind4\uc1
\pard\sa200\sl276\slmult1\i\f0\fs22\lang7 Formatierter \b\i0 Inhalt/Text \b0 der
\ul Aufgabe\ulnone\par}"
```

Beispiel C#-Script:

```
todo.Body =
@"{\rtf1\ansi\ansicpg1252\deff0\nouicompat\deflang1031{\fonttbl{\f0\fnil\fcharset0
Calibri;}}{\*\generator Riched20 10.0.22000}\viewkind4\uc1
\pard\sa200\sl276\slmult1\i\f0\fs22\lang7 Formatierter \b\i0 Inhalt/Text \b0 der
\ul Aufgabe\ulnone\par}";
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

BodyPlain

Beschreibung:

Setzt den Inhalt/Text der Aufgabe (ohne RTF-Formatierung) oder liefert diesen zurück.

Typ:

String

Beispiel VBScript:

```
oToDo.BodyPlain = "Inhalt/Text des Termins ohne RTF-Formatierung"
```

Beispiel C#-Script:

```
todo.BodyPlain = "Inhalt/Text des Termins ohne RTF-Formatierung"
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Categories

Beschreibung:

Liefert alle Kategorien der Aufgabe als Objekt vom Typ **Categories** zurück.

Typ:

Categories

Beispiel VBScript:

```
oToDo.Categories.Add("Meeting")
```

Beispiel C#-Script:

```
todo.Categories.Add("Meeting");
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

ChangeDate

Beschreibung:

Liefert das letzte Änderungsdatum der Aufgabe zurück oder setzt dieses.

Typ:

Date

Beispiel VBScript:

```
oToDo.ChangeDate = Now()
```

Beispiel C#-Script:

```
todo.ChangeDate = System.DateTime.Now;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

ChangeUser

Beschreibung:

Liefert den letzten Änderungsbenutzer der Aufgabe zurück oder setzt diesen.

Typ:

String

Beispiel VBScript:

```
oToDo.ChangeUser = sCurrentUserLoginName
```

Beispiel C#-Script:

```
todo.ChangeUser = currentUserLoginName;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Contact

Beschreibung:

Liefert den Kontakt, mit dem die Aufgabe stattfindet ("Mit"), zurück oder setzt diesen.

Typ:

String

Beispiel VBScript:

```
oToDo.Contact = "Kontakt, mit dem die Aufgabe stattfindet"
```

Beispiel C#-Script:

```
todo.Contact = "Kontakt, mit dem die Aufgabe stattfindet";
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel ToDo Objekt.

CreationDate

Beschreibung:

Liefert das Erfassungsdatum der Aufgabe zurück oder setzt dieses.

Typ:

Date

Beispiel VBScript:

```
oToDo.CreationDate = Now()
```

Beispiel C#-Script:

```
todo.CreationDate = System.DateTime.Now;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel ToDo Objekt.

CreationUser

Beschreibung:

Liefert den Erfassungsbenutzer der Aufgabe zurück oder setzt diesen.

Typ:

String

Beispiel VBScript:

```
oToDo.CreationUser = sCurrentUserLoginName
```

Beispiel C#-Script:

```
todo.CreationUser = currentUserLoginName;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel ToDo Objekt.

End

Beschreibung:

Liefert das Enddatum und –zeit der Aufgabe/Fälligkeitsdatum zurück oder setzt dieses.

Typ:

Date

Beispiel VBScript:

```
oToDo.End = DateAdd("h", 2, Now())
```

Beispiel C#-Script:

```
todo.End = System.DateTime.Now.AddHours(2);
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

ExtUserData1

Beschreibung:

Ermöglicht das Hinterlegen oder Auslesen von Zusatzinformationen in Form von Zeichenketten (Strings).

> **Hinweis**: Die Zeichenketten können maximal jeweils 250 Zeichen lang werden.

Typ:

String

Beispiel VBScript:

```
oToDo.ExtUserData1 = "Erster Abschnitt von Zusatzinformationen"
```

Beispiel C#-Script:

```
todo.ExtUserData1 = "Erster Abschnitt von Zusatzinformationen";
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

ExtUserData2

Beschreibung:

Ermöglicht das Hinterlegen oder Auslesen von Zusatzinformationen in Form von Zeichenketten (Strings).

> **Hinweis**: Die Zeichenketten können maximal jeweils 250 Zeichen lang werden.

Typ:

String

Beispiel VBScript:

```
oToDo.ExtUserData2 = "Zweiter Abschnitt von Zusatzinformationen"
```

Beispiel C#-Script:

```
todo.ExtUserData2 = "Zweiter Abschnitt von Zusatzinformationen";
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

HostDatabase

Beschreibung:

Setzt die eindeutige Projekt-ID der Aufgabe oder liest diese aus. Der Aufbau ist folgender:

<ProjektID>|<Ansichtenname>|<Ansichtenfamilienname(optional)>|<Primärschlüsselfeld-name>|<Primärschlüsselwert>|<Bezeichnung Datensatzverknüpfung>

Typ:

String

Beispiel VBScript:

```
Dim oProject : Set oProject = cRM.CurrentProject
Dim oActiveView : Set oActiveView = cRM.CurrentProject.ActiveViews.ActiveView
Dim oViewConfig : Set oViewConfig = oActiveView.Config
Dim oRecord : Set oRecord = oActiveView.CurrentRecordSet.CurrentRecord
Dim sPrimaryKeyFieldName : sPrimaryKeyFieldName = oViewConfig.PrimaryKeyFldName
Dim sRecordRefDescription : sRecordRefDescription =
oRecord.GetRecordRefDescription
Dim sRefLink : sRefLink = oProject.ID & "|" & oActiveView.Name & "|" &
oViewConfig.FamilyName & "|" & sPrimaryKeyFieldName & "|" &
oRecord.GetContentsByName(sPrimaryKeyFieldName) & "|" & sRecordRefDescription
Set oRecord = Nothing
Set oViewConfig = Nothing
Set oActiveView = Nothing
Set oProject = Nothing

…

oToDo.HostDataBase = sRefLink
```

Beispiel C#-Script:

```
Project project = cRM.CurrentProject;
View activeView = project.ActiveViews.ActiveView;
ViewConfig viewConfig = activeView.Config;
Record record = activeView.CurrentRecordSet.CurrentRecord;
string primaryKeyFieldName = viewConfig.PrimaryKeyFldName;
string recordRefDescription = record.GetRecordRefDescription();
string refLink = project.ID + "|" + activeView.Name + "|" + viewConfig.FamilyName
+ "|" + primaryKeyFieldName + "|" + record.GetContentsByName(primaryKeyFieldName)
+ "|" + recordRefDescription;
record.Dispose();
viewConfig.Dispose();
activeView.Dispose();
project.Dispose();

…

todo.HostDatabase = refLink;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

HostID

Beschreibung:

Setzt die eindeutige ID des (optional) zugeordneten Mutterprogrammes (HOST) der Aufgabe, wie z. B. der Adressverwaltung, oder liest diese aus.

Typ:

String

Beispiel VBScript:

```
oToDo.HostID = "IDCRM"
```

Beispiel C#-Script:

```
todo.HostID = "IDCRM";
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

HostRecordID

Beschreibung:

Setzt korrespondierend zu **HostDatabase** die eindeutige Datensatznummerierung des (optional) zugeordneten Datensatzes oder liest diese aus.

Typ:

String

Beispiel VBScript:

```
Dim sHostRecordID : sHostRecordID = oToDo.HostRecordID
```

Beispiel C#-Script:

```
string hostRecordID = todo.HostRecordID;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Importance

Beschreibung:

Setzt die Priorität der Aufgabe (1 bis 5) oder liefert diese zurück.

Typ:

Long

Beispiel VBScript:

```
oToDo.Importance = 3
```

Beispiel C#-Script:

```
todo.Importance = 3;
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Links

Beschreibung:

Liefert alle Verknüpfungen der Aufgabe als Objekt vom Typ **Links** zurück.

Typ:

Links

Beispiel VBScript:

```
Call oToDo.Links.NewLink.SetLinkFromString(sRefLink)
```

Beispiel C#-Script:

```
todo.Links.NewLink.SetLinkFromString(refLink);
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

PercentComplete

Beschreibung:

Prozentzahl, zu der die Aufgabe als "erledigt" gilt.

Typ:

Long

Beispiel VBScript:

```
oToDo.PercentComplete = 30
```

Beispiel C#-Script:

```
todo.PercentComplete = 30;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Private

Beschreibung:

Legt fest, ob die Aufgabe als "Privat" markiert werden soll oder liest diese Eigenschaft aus.

Typ:

Bool

Beispiel VBScript:

```
oToDo.Private = False
```

Beispiel C#-Script:

```
todo.Private = false;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Reminder

Beschreibung:

Legt fest, ob an diese Aufgabe erinnert werden soll oder liest diese Eigenschaft aus.

Typ:

Bool

Beispiel VBScript:

```
oToDo.Reminder = True
```

Beispiel C#-Script:

```
todo.Reminder = true;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

ReminderMinutesBeforeStart

Beschreibung:

Gibt die Anzahl der Minuten zurück, die vor der Fälligkeit der Aufgabe an diese erinnert werden soll oder setzt diese.

Typ:

Long

Beispiel VBScript:

```
oToDo.ReminderMinutesBeforeStart = 30
```

Beispiel C#-Script:

```
todo.ReminderMinutesBeforeStart = 30;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Start

Beschreibung:

Setzt oder liefert Startdatum und –zeit der Aufgabe.

Typ:

Date

Beispiel VBScript:

```
oToDo.Start = DateAdd("h", 1, Now())
```

Beispiel C#-Script:

```
todo.Start = System.DateTime.Now;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Subject

Beschreibung:

Setzt oder liefert die Beschreibung der Aufgabe.

Typ:

String

Beispiel VBScript:

```
oToDo.Subject = "Priorität " & oToDo.Importance & ": Meeting im Büro mit " &
sRecordRefDescription
```

Beispiel C#-Script:

```
todo.Subject = "Priorität" + todo.Importance.ToString() + ": Meeting im Büro mit "
+ recordRefDescription;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

TimeStamp, read-only

Beschreibung:

Liefert eine Zeichenkette, die aus dem Änderungsdatum und einem fortlaufenden Zahlenwert (als String) besteht. Der Zahlenwert wird bei jeder Datensatzänderung verändert. Durch diesen Wert kann man auch feststellen, wenn ein Datensatz sich an einem Tag mehrfach geändert hat.

Format:YYYYMMDD:HHMMSS, z. B. 20070919:100120

Typ:

String

Beispiel VBScript:

```
oToDo.TimeStamp = FormatDateTime(Now(), vbGeneralDate)
```

Beispiel C#-Script:

```
todo.TimeStamp = System.DateTime.Now.ToString();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

UniqueID, read-only

Beschreibung:

Liefert die eindeutige RecordID der Aufgabe.

Typ:

String

Beispiel VBScript:

```
Dim sUniqueID : sUniqueID = oToDo.UniqueID
```

Beispiel C#-Script:

```
string uniqueID = todo.UniqueID;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

User

Beschreibung:

Setzt oder liefert den Eigentümer der Aufgabe.

Typ:

String

Beispiel VBScript:

```
oToDo.User = sCurrentUserLoginName
```

Beispiel C#-Script:

```
todo.User = currentUserLoginName;
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

3.43.2 Methoden

Display

Beschreibung:

Zeigt einen Dialog zur Aufgabe an.

Parameter:

Parametername	Typ	Beschreibung
bWait	Bool	True: Das Script wartet auf die Beendigung des Dialoges. False: Das Script läuft wartet nicht auf die Beendigung des Dialoges.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call oToDo.Display()
```

Beispiel C#-Script:

```
todo.Display();
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

ExportAsICal

Beschreibung:

Exportiert die Aufgabe in das iCal-Format.

Parameter:

Parametername	Typ	Beschreibung
sFileName	String	Pfad und Dateiname der zu exportierenden Datei.

Rückgabewert:

Bool

Wert	Beschreibung
True	Datei wurde exportiert.
False	Beim Exportieren der Datei ist ein Fehler aufgetreten.

Beispiel VBScript:

```
Call oToDo.ExportAsICal("C:\" & sRecordRefDescription & ".ical")
```

Beispiel C#-Script:

```
todo.ExportAsiCal(@"C:\" + recordRefDescription + ".ical");
```

Hinweis: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Remove

Beschreibung:

Löscht die Aufgabe.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call oToDo.Remove()
```

Beispiel C#-Script:

```
todo.Remove();
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

Save

Beschreibung:

Speichert die Einstellungen der Aufgabe.

Rückgabewert:

Bool

> **Wichtig**: Um die neu angelegte Aufgabe in der Ansicht gleich angezeigt zu bekommen, muss UpdateViews nach der Speicherung aufgerufen werden.

Beispiel VBScript:

```
Call oToDo.Save()
```

Beispiel C#-Script:

```
todo.Save();
```

> **Hinweis**: Vollständige Beispiele für VBScript und C#-Script befinden sich im Kapitel **ToDo Objekt**.

3.44 ToDos Objekt

3.44.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der Einträge in der Sammlung zurück.

Typ:

Long

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Es befinden sich derzeit " &
CStr(cRM.CurrentProject.timemanager.ToDos.Count) & " Aufgaben in der Termin- und
Aufgabenplanung des combit CRM.", "ToDos.Count", vbOKOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Es befinden sich derzeit " +
cRM.CurrentProject.TimeManager.ToDos.Count.ToString() + " Aufgaben in der Termin-
und Aufgabenplanung des combit CRM.", "ToDos.Count", 0);
```

3.44.2 Methoden

Add

Beschreibung:

Legt eine neue Aufgabe an und liefert diese als Objekt vom Typ **ToDo** zurück.

Rückgabewert:

ToDo

Beispiel VBScript:

```
Dim oToDo : Set oToDo = cRM.CurrentProject.timemanager.ToDos.Add()
```

Beispiel C#-Script:

```
ToDo todo = cRM.CurrentProject.TimeManager.ToDos.Add();
```

Item

Beschreibung:

Gibt eine Aufgabe zurück. Es muss die Index-Nummer der Aufgabe übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

ToDo

Beispiel VBScript:

```
Dim nCount : nCount = 0
Dim nCountAllToDos : nCountAllToDos = cRM.CurrentProject.timemanager.ToDos.Count
Dim nCountAllPrivateToDos : nCountAllPrivateToDos = 0
Dim oToDo

For nCount = 1 To nCountAllToDos

    Set oToDo = cRM.CurrentProject.timemanager.ToDos.Item(nCount)

    If (oToDo.Private = True) Then
        nCountAllPrivateToDos = nCountAllPrivateToDos + 1
    End If

    Set oToDo = Nothing

Next

Call cRM.DialogMessageBox("Bei der Gesamtanzahl von " & CStr(nCountAllToDos) & "
Aufgaben, gibt es " & CStr(nCountAllPrivateToDos) & " private Einträge.",
"ToDos.Item", vbOkonly)
```

Beispiel C#-Script:

```
long counterPrivateToDos = 0;

foreach (ToDo todo in cRM.CurrentProject.TimeManager.ToDos)
{
    if (todo.Private == true)
    {
        counterPrivateToDos++;
    }
```

```
}

cRM.DialogMessageBox("Bei der Gesamtanzahl von " +
cRM.CurrentProject.TimeManager.ToDos.Count.ToString() + " Aufgaben, gibt es " +
counterPrivateToDos + " private Einträge.", "ToDos.Item", 0);
```

ItemByUniqueID

Beschreibung:

Gibt eine Aufgabe anhand der eindeutigen RecordID zurück.

Parameter:

Parametername	Typ	Beschreibung
UniqueID	String	Eindeutige RecordID der Aufgabe.

Rückgabewert:

ToDo

Beispiel VBScript:

```
Dim oToDo : Set oToDo =
cRM.CurrentProject.timemanager.ToDos.ItemByUniqueID("Eindeutige ID der Aufgabe")
```

Beispiel C#-Script:

```
ToDo todo = cRM.CurrentProject.TimeManager.ToDos.ItemByUniqueID("Eindeutige ID der
Aufgabe");
```

Remove

Beschreibung:

Löscht eine Aufgabe. Es muss die Index-Nummer der Aufgabe übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oToDos : Set oToDos = cRM.CurrentProject.timemanager.ToDos
Call oToDos.Remove(nIndex)
Set oToDos = Nothing
```

Beispiel C#-Script:

```
ToDos todos = cRM.CurrentProject.TimeManager.ToDos;
todos.Remove(index);
todos.Dispose();
```

RemoveAll

Beschreibung:

Löscht alle Aufgaben.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oToDos: Set oToDos = cRM.CurrentProject.timemanager.ToDos
Call oToDos.RemoveAll()
```

```
Set oToDos = Nothing
```

Beispiel C#-Script:

```
ToDos todos = cRM.CurrentProject.TimeManager.ToDos;
todos.RemoveAll();
todos.Dispose();
```

SetFilter

Beschreibung:

Filtert die Anzeige auf Aufgaben, die bestimmte Eigenschaften erfüllen. Die Filterung kann ausgehend von Datenbank oder Datensatz der Host-Applikation erfolgen oder auf Basis des Benutzers.

Parameter:

Parametername	Typ	Beschreibung
FilterType	TMListFilter Constants	Art der Filterung.
HostDatabase	String	Datenbank (Pfad+Name).
HostRecordID	String	Eindeutige Datensatznummer des zugeordneten Datensatzes.
UserName	String	Name des Benutzers.

Konstanten der TMListFilterConstants:

Konstante	Wert	Beschreibung
TM_FILTER_HOSTDB	1	Filterung aller Aufgaben zu einer bestimmten Datenbank, korrespondierend zu der Eigenschaft HostDatabase des Appointment-Objektes.
TM_FILTER_HOSTRECID	2	Filterung aller Aufgaben zu einem bestimmten Datensatz, korrespondierend zu der Eigenschaft HostRecordID des Appointment-Objektes. Impliziert die Verwendung eines Filters auf Datenbank-Ebene, vgl. TM_FILTER_HOSTDB.
TM_FILTER_USER	4	Filterung aller Aufgaben zu einem bestimmten Benutzer.

Rückgabewert:

Bool

Beispiel VBScript:

```
Dim oToDos: Set oToDos = cRM.CurrentProject.timemanager.ToDos
Call oToDos.SetFilter(1, hostDatabase, hostRecordID, userName)
Set oToDos = Nothing
```

Beispiel C#-Script:

```
ToDos todos = cRM.CurrentProject.TimeManager.ToDos;
todos.SetFilter(1, hostDatabase, hostRecordID, userName);
todos.Dispose();
```

3.45 User Objekt

Zugriff auf die Stammdaten eines Benutzers der Anwendung. Das **User**-Objekt erhält man über **CurrentUser** oder **ItemByName** der **Users**-Collection.

3.45.1 Eigenschaften

Beispiel für alle Eigenschaften

VBScript:

```
Dim oUser : Set oUser = cRM.CurrentProject.Users.CurrentUser
Dim bIsLoggedIn : bIsLoggedIn = oUser.IsLoggedIn
Dim sDepartment : sDepartment = oUser.Department
Dim sEmail : sEmail = oUser.Email
Dim sEmail2 : sEmail2 = oUser.Email2
Dim sFax : sFax = oUser.Fax
Dim sFirstName : sFirstName = oUser.FirstName
Dim sGroupNames : sGroupNames = oUser.GroupNames
Dim sLastName : sLastName = oUser.LastName
Dim sLoginName : sLoginName = oUser.LoginName
Dim sMobile : sMobile = oUser.Mobile
Dim sMobile2 : sMobile2 = oUser.Mobile2
Dim sPhone : sPhone = oUser.Phone
Dim sPhone2 : sPhone2 = oUser.Phone2
Dim sPicture : sPicture = oUser.Picture
Dim sPosition : sPosition = oUser.Position
Dim sShortName : sShortName = oUser.ShortName
Dim sSignature : sSignature = oUser.Signature
Dim sTitle : sTitle = oUser.Title
Dim sUserDefined1 : sUserDefined1 = oUser.UserDefined1
Dim sUserDefined2 : sUserDefined2 = oUser.UserDefined2

Call cRM.DialogMessageBox("Folgende Informationen sind für den aktuell
angemeldeten Benutzer hinterlegt worden:" & vbCrLf & "Ist angemeldet: " &
bIsLoggedIn & vbCrLf & "Login-Name: " & sLoginName & vbCrLf & "Anrede: " & sTitle
& vbCrLf & "Name: " & sLastName & vbCrLf & "Vorname: " & sFirstName & vbCrLf &
"Kurzname: " & sShortName & vbCrLf & "Position: " & sPosition & vbCrLf &
"Abteilung: " & sDepartment & vbCrLf & "Telefon: " & sPhone & vbCrLf & "Telefon2:
" & sPhone2 & vbCrLf & "Mobiltelefon: " & sMobile & vbCrLf & "Mobiltelefon2: " &
sMobile2 & vbCrLf & "Telefax: " & sFax & vbCrLf & "E-Mail: " & sEmail & vbCrLf &
"E-Mail2: " & sEmail2 & vbCrLf & "Zusatz1: " & sUserDefined1 & vbCrLf & "Zusatz2:
" & sUserDefined2 & vbCrLf & "Unterschrift: " & sSignature & vbCrLf & "Bild: " &
sPicture & vbCrLf & "Mitgliedschaften: " & sGroupNames, "User", vbOkOnly)

Set oUser = Nothing
```

C#-Script:

```
User user = cRM.CurrentProject.Users.CurrentUser;
bool isLoggedIn = user.IsLoggedIn;
string department = user.Department;
string email = user.Email;
string email2 = user.Email2;
string fax = user.Fax;
string firstName = user.FirstName;
string groupNames = user.GroupNames;
string lastName = user.LastName;
string loginName = user.LoginName;
string mobile = user.Mobile;
string mobile2 = user.Mobile2;
string phone = user.Phone;
string phone2 = user.Phone2;
string picture = user.Picture;
string position = user.Position;
string shortName = user.ShortName;
string signature = user.Signature;
string title = user.Title;
string userDefined1 = user.UserDefined1;
string userDefined2 = user.UserDefined2;

cRM.DialogMessageBox("Folgende Informationen sind für den aktuell angemeldeten
Benutzer hinterlegt worden:" + System.Environment.NewLine + "Ist angemeldet: " +
isLoggedIn + System.Environment.NewLine + "Login-Name: " + loginName +
```

```
System.Environment.NewLine + "Anrede: " + title + System.Environment.NewLine +
"Name: " + lastName + System.Environment.NewLine + "Vorname: " + firstName +
System.Environment.NewLine + "Kurzname: " + shortName + System.Environment.NewLine
+ "Position: " + position + System.Environment.NewLine + "Abteilung: " +
department + System.Environment.NewLine + "Telefon: " + phone +
System.Environment.NewLine + "Telefon2: " + phone2 + System.Environment.NewLine +
"Mobiltelefon: " + mobile + System.Environment.NewLine + "Mobiltelefon2: " +
mobile2 + System.Environment.NewLine + "Telefax: " + fax +
System.Environment.NewLine + "E-Mail: " + email + System.Environment.NewLine + "E-
Mail2: " + email2 + System.Environment.NewLine + "Zusatz1: " + userDefined1 +
System.Environment.NewLine + "Zusatz2: " + userDefined2 +
System.Environment.NewLine + "Unterschrift: " + signature +
System.Environment.NewLine + "Bild: " + picture + System.Environment.NewLine +
"Mitgliedschaften: " + groupNames, "User", 0);

user.Dispose();
```

Department, read-only

Beschreibung:

Liefert Abteilung zurück.

Typ:

String

Email, read-only

Beschreibung:

Liefert E-Mail zurück.

Typ:

String

Email2, read-only

Beschreibung:

Liefert E-Mail2 zurück.

Typ:

String

Fax, read-only

Beschreibung:

Liefert Fax zurück.

Typ:

String

FirstName, read-only

Beschreibung:

Liefert Vorname zurück.

Typ:

String

GroupNames, read-only

Beschreibung:

Liefert in einem String kommasepariert alle Gruppen (jeweils in einfache Anführungszeichen eingeschlossen), in denen der Benutzer Mitglied ist, z. B. 'Vertrieb', 'Administratoren'.

Typ:

String

IsLoggedIn, read-only

Beschreibung:

Liefert True zurück, wenn der Benutzer gerade eingeloggt ist.

Typ:

Bool

LastName, read-only

Beschreibung:

Liefert Nachname zurück.

Typ:

String

LoginName, read-only

Beschreibung:

Liefert Login-Name zurück.

Typ:

String

Mobile, read-only

Beschreibung:

Liefert Mobiltelefon zurück.

Typ:

String

Mobile2, read-only

Beschreibung:

Liefert Mobiltelefon2 zurück.

Typ:

String

Phone, read-only

Beschreibung:

Liefert Telefon zurück.

Typ:

String

Phone2, read-only

Beschreibung:

Liefert Telefon2 zurück.

Typ:

String

Picture, read-only

Beschreibung:

Liefert den Dateipfad des Bildes zurück.

Typ:

String

Position, read-only

Beschreibung:

Liefert Position zurück.

Typ:

String

ShortName, read-only

Beschreibung:

Liefert Kurzname zurück.

Typ:

String

Signature, read-only

Beschreibung:

Liefert den Dateipfad der Unterschrift zurück.

Typ:

String

Title, read-only

Beschreibung:

Liefert Anrede zurück.

Typ:

String

UserDefined1, read-only

Beschreibung:

Liefert Zusatz1 zurück.

Typ:

String

UserDefined2, read-only

Beschreibung:

Liefert Zusatz2 zurück.

Typ:

String

3.46 Users Objekt

Bietet den Zugriff auf die konfigurierten Benutzer der Anwendung.

Hinweis: Für den Zugriff auf andere Benutzer als den eigenen, wird das Recht 'Andere Benutzer sehen' benötigt. Ist dieses nicht gewährt, so liefert Count() immer 1 und der Zugriff auf Item(1) liefert immer den eigenen Benutzer. ItemByName funktioniert in diesem Falle auch nur, wenn der eigene Benutzername übergeben wird.

3.46.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl der konfigurierten Benutzer der Anwendung zurück.

Typ:

Long

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Derzeit wurden " & CStr(cRM.CurrentProject.Users.Count)
& " Benutzer im combit CRM konfiguriert.", "Users.Count", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Derzeit wurden " + cRM.CurrentProject.Users.Count.ToString()
+ " Benutzer im combit CRM konfiguriert.", "Users.Count", 0);
```

3.46.2 Methoden

CurrentUser

Beschreibung:

Liefert angemeldeten Benutzer der Anwendung als Objekt vom Typ **User** zurück.

Rückgabewert:

User

Beispiel VBScript:

```
Dim oUser : Set oUser = cRM.CurrentProject.Users.CurrentUser
' ...
Set oUser = Nothing
```

Beispiel C#-Script:

```
User user = cRM.CurrentProject.Users.CurrentUser;
// ...
user.Dispose();
```

Item

Beschreibung:

Gibt einen Benutzer zurück. Es muss die Index-Nummer des Eintrages übergeben werden. Der Index geht von 1 bis Count.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Rückgabewert:

User

Beispiel VBScript:

```
Dim oUsers : Set oUsers = cRM.CurrentProject.Users
Dim nUserCount : nUserCount = oUsers.Count
Dim oUser
Dim nCounter : nCounter = 0
Dim nCurrentlyLoggedIn : nCurrentlyLoggedIn = 0

For nCounter = 1 To nUserCount

    Set oUser = oUsers.Item(nCounter)

    If (oUser.IsLoggedIn = True) Then
        nCurrentlyLoggedIn = nCurrentlyLoggedIn + 1
    End If

    Set oUser = Nothing

Next

Call cRM.DialogMessageBox("Aktuelle Anzahl angemeldeter Benutzer: " &
CStr(nCurrentlyLoggedIn), "Users.Item", vbOkOnly)
```

```
Set oUsers = Nothing
```

Beispiel C#-Script:

```
Users users = cRM.CurrentProject.Users;
long loggedInUserCount = 0;

foreach (User user in users)
{
    if (user.IsLoggedIn == true)
    {
        loggedInUserCount++;
    }
}

cRM.DialogMessageBox("Aktuelle Anzahl angemeldeter Benutzer: " +
loggedInUserCount.ToString(), "Users.Item", 0);

users.Dispose();
```

ItemByName

Beschreibung:

Liefert einen Benutzer mit dem übergebenen Namen als Objekt zurück.

Parameter:

Parametername	Typ	Beschreibung
UserName	String	Name des Benutzers

Rückgabewert:

User

Beispiel VBScript:

```
Dim oUser : Set oUser = cRM.CurrentProject.Users.ItemByName("Administrator")
' ...
Set oUser = Nothing
```

Beispiel C#-Script:

```
User user = cRM.CurrentProject.Users.ItemByName("Administrator");
// ...
user.Dispose();
```

SyncWithActiveDirectory

Beschreibung:

Synchronisiert die Benutzerverwaltung der Anwendung mit dem konfigurierten Active Directory.

Wichtig: Diese Methode steht erst ab der Professional-Edition zur Verfügung.

Rückgabewert:

Long

Wert	Beschreibung
0	Synchronisation erfolgreich.
1	Keine Konfiguration gefunden.
2	Fehler während der Synchronisation.
3	Mehrere Benutzer mit gleichem Login Namen oder Windows Login Namen sind nicht erlaubt.
4	Benutzer ist weder Administrator noch hat er administrative Rechte.

Beispiel VBScript:

```
Dim nReturn : nReturn = cRM.CurrentProject.Users.SyncWithActiveDirectory()
Dim sResult : sResult = ""

Select Case nReturn
    Case 0
        sResult = "Synchronisation erfolgreich"
    Case 1
        sResult = "Keine Konfiguration gefunden"
    Case 2
        sResult = "Fehler während der Synchronisation"
    Case 3
        sResult = "Mehrere Benutzer mit gleichem Login Namen oder Windows Login
Namen sind nicht erlaubt"
    Case 4
        sResult = "Benutzer ist weder Administrator, noch hat er administrative
Rechte"
End Select

Call cRM.DialogMessageBox("Ergebnis der Active Directory-Synchronisation: " &
sResult, "Users.SyncWithActiveDirectory", vbOkOnly)
```

Beispiel C#-Script:

```
long returnValue = cRM.CurrentProject.Users.SyncWithActiveDirectory();
string result = null;

switch (returnValue)
{
    case 0:
        result = "Synchronisation erfolgreich";
        break;
    case 1:
        result = "Keine Konfiguration gefunden";
        break;
    case 2:
        result = "Fehler während der Synchronisation";
        break;
    case 3:
        result = "Mehrere Benutzer mit gleichem Login Namen oder Windows Login
Namen sind nicht erlaubt";
        break;
    case 4:
        result = "Benutzer ist weder Administrator, noch hat er administrative
Rechte";
        break;
    default:
        break;
}

cRM.DialogMessageBox("Ergebnis der Active Directory-Synchronisation: " + result,
"Users.SyncWithActiveDirectory", 0);
```

3.47 View Objekt

Repräsentiert eine aktive (geöffnete) Ansicht in der Anwendung.

Hinweis: Unter C# kann es bei der Initalisierung des Objektes zu einem Konflikt mit einem weiteren View-Objekt des Namespaces System.Windows.Forms kommen. Wir empfehlen daher den kompletten Klassennamen zu verwenden:

Beispiel:
combit.cRM.COM.View activeView = cRM.CurrentProject.ActiveViews.ActiveView;

3.47.1 Eigenschaften

Config, read-only

Beschreibung:

Erzeugt ein Objekt vom Typ ViewConfig.

Typ:

ViewConfig

Beispiel VBScript:

```
Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ActiveViews.ActiveView.Config
' ...
Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
ViewConfig config = cRM.CurrentProject.ActiveViews.ActiveView.Config;
// ...
config.Dispose();
```

Editmode

Beschreibung:

Liefert oder verändert den Status des Bearbeitungsmodus in der aktuellen Ansicht.

Typ:

Bool

Beispiel VBScript:

```
If (cRM.CurrentProject.ActiveViews.ActiveView.EditMode = False) Then
    cRM.CurrentProject.ActiveViews.ActiveView.EditMode = True
End If
```

Beispiel C#-Script:

```
if (cRM.CurrentProject.ActiveViews.ActiveView.EditMode == false)
{
    cRM.CurrentProject.ActiveViews.ActiveView.EditMode = true;
}
```

FilterActive, read-only

Beschreibung:

Liefert True zurück, wenn in der Ansicht ein vom Benutzer angelegter Filter aktiv ist.

Typ:

Bool

Beispiel VBScript:

```
If (cRM.CurrentProject.ActiveViews.ActiveView.FilterActive = True) Then
    Call cRM.DialogMessageBox("In der aktuell geöffneten Ansicht ist ein Filter
aktiv.", "View.FilterActive", vbOkOnly)
End If
```

Beispiel C#-Script:

```
if (cRM.CurrentProject.ActiveViews.ActiveView.FilterActive == true)
{
    cRM.DialogMessageBox("In der aktuell geöffneten Ansicht ist ein Filter
aktiv.", "View.FilterActive", 0);
}
```

FilterRecCount, read-only

Beschreibung:

Gibt die Anzahl der Datensätze zurück. Ein etwaiger Filter wird dabei berücksichtigt. Ist kein Filter aktiv, wird die Anzahl aller Datensätze der Ansicht zurückgegeben (siehe **RecCount**).

Typ:

Long

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Die aktuelle Anzahl der angezeigten Datensätze beträgt:
" & CStr(cRM.CurrentProject.ActiveViews.ActiveView.FilterRecCount),
"View.FilterRecCount", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Die aktuelle Anzahl der angezeigten Datensätze beträgt: " +
cRM.CurrentProject.ActiveViews.ActiveView.FilterRecCount.ToString(),
"View.FilterRecCount", 0);
```

Name, read-only

Beschreibung:

Liefert den Namen der Ansicht zurück.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Der Name der aktuell angezeigten Ansicht lautet """ &
cRM.CurrentProject.ActiveViews.ActiveView.Name & "", "View.Name", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Der Name der aktuell angezeigten Ansicht lautet " +
cRM.CurrentProject.ActiveViews.ActiveView.Name, "View.Name", 0);
```

RecCount, read-only

Beschreibung:

Gibt die Anzahl der Datensätze zurück. Ein etwaiger Filter wird dabei ignoriert. Soll dieser berücksichtigt werden, verwenden Sie stattdessen **FilterRecCount**.

Typ:

Long

Wert	Beschreibung
Long	Anzahl der Datensätze
- 1	Fehler

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Die Gesamtzahl der Datensätze in der aktuell
angezeigten Ansicht beträgt: " &
CStr(cRM.CurrentProject.ActiveViews.ActiveView.RecCount), "View.RecCount",
vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Die Gesamtzahl der Datensätze in der aktuell angezeigten
Ansicht beträgt: " +
cRM.CurrentProject.ActiveViews.ActiveView.RecCount.ToString(), "View.RecCount",
0);
```

RowGUIDFilterActive, read-only

Beschreibung:

Diese Eigenschaft zeigt an, ob der interne Filter, der zu einem Datensatzsprung geführt hat, aktiv ist. Dieser wird z. B. ausgeführt, wenn man auf einen Favoriten-Datensatz springt.

Typ:

Bool

Beispiel VBScript:

```
If (cRM.CurrentProject.ActiveViews.ActiveView.RowGUIDFilterActive = True) Then
    Call cRM.DialogMessageBox("Derzeit ist ein interner Filter aktiv, der zu einem
Datensatzsprung geführt hat.", "View.RowGUIDFilterActive", vbOkOnly)
End If
```

Beispiel C#-Script:

```
if (cRM.CurrentProject.ActiveViews.ActiveView.RowGUIDFilterActive == true)
{
    cRM.DialogMessageBox("Derzeit ist ein interner Filter aktiv, der zu einem
Datensatzsprung geführt hat.", "View.RowGUIDFilterActive", 0);
}
```

TagID

Beschreibung:

Mit Hilfe dieser Eigenschaft können Sie eine eigene TagID (Identifiername) für eine aktive Ansicht übergeben bzw. auslesen. Ein Objekt vom Typ **View** (aktive Ansicht) kann anschließend über **ItemByTagID** erzeugt werden.

Typ:

String

Beispiel VBScript:

```
Call cRM.DialogMessageBox("Die TagID der Ansicht lautet: " &
cRM.CurrentProject.ActiveViews.ActiveView.TagID, "View.TagID", vbOkOnly)
```

Beispiel C#-Script:

```
cRM.DialogMessageBox("Die TagID der Ansicht lautet: " +
cRM.CurrentProject.ActiveViews.ActiveView.TagID, "View.TagID", 0);
```

ViewMode

Beschreibung:

Gibt den Modus der betreffenden Ansicht zurück oder setzt diesen.

Typ:

Integer

Wert	Beschreibung
1	Eingabemaske
2	Übersichtsliste
3	Web
4	Bericht
5	Vertikal geteilt
6	Horizontal geteilt
7	Landkarte

Beispiel VBScript:

```
If (cRM.CurrentProject.ActiveViews.ActiveView.ViewMode = 2) Then
```

```
    cRM.CurrentProject.ActiveViews.ActiveView.ViewMode = 1
End If
```

Beispiel C#-Script:

```
if (cRM.CurrentProject.ActiveViews.ActiveView.ViewMode == 2)
{
    cRM.CurrentProject.ActiveViews.ActiveView.ViewMode = 1;
}
```

3.47.2 Methoden

Activate

Beschreibung:

Aktiviert die Ansicht und bringt sie in den Vordergrund.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.CurrentProject.ActiveViews.ItemByName("Kontakte").Activate()
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ItemByName("Kontakte").Activate();
```

Close

Beschreibung:

Close versucht die Ansicht zu schließen.

Wichtig: Ruft man Close in der aktuellen Ansicht auf (in der das Script läuft), kann dies zu Problemen führen. Dies sollte daher vermeiden werden.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.CurrentProject.ActiveViews.ItemByName("Kontakte").Close()
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.CurrentProject.ActiveViews.ItemByName("Kontakte").Close();
```

CurrentInputForm

Beschreibung:

Diese Methode wechselt in der aktuellen Ansicht in den Bearbeitungsmodus und liefert das **InputForm** Objekt der aktuellen Eingabemaske zurück.

Parameter:

Parametername	Typ	Beschreibung
nMode	Long	0 = Wechsel in Bearbeiten-Modus (kompatibel zu bisher)

		1 = Neuanlage eines Datensatzes und Wechsel in Bearbeiten-Modus (kompatibel zu bisher) 2 = Beibehalten des aktuellen Modus

Typ:

InputForm

Beispiel VBScript:

```
Dim oInputForm : Set oInputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2)
' ...
Set oInputForm = Nothing
```

Beispiel C#-Script:

```
InputForm inputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2);
// ...
inputForm.Dispose();
```

CurrentRecordSet

Beschreibung:

Liefert ein Objekt vom Typ **RecordSet** zurück.

Rückgabewert:

RecordSet

Beispiel VBScript:

```
Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet
' ...
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
RecordSet recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet;
// ...
recordSet.Dispose();
```

CurrentRecordSetCopy

Beschreibung:

Liefert ein Objekt vom Typ **RecordSet** zurück, das die identischen Datensätze des aktiven Filters der Ansicht enthält, jedoch von der Ansicht unabhängig ist.

Hinweis: Wir empfehlen, nach Erzeugung eines **RecordSet**-Objektes zunächst mittles Aufruf der Methode "MoveFirst" die Existenz mindestens eines **Record**-Objektes zu überprüfen.

Wichtiger Hinweis für die Verwendung des Parameters CursorModel mit dem Wert 2 (forward-only) unter Microsoft SQL Server: Die Datensätze eines forward-only-RecordSets müssen nach dessen Erstellung direkt und unmittelbar über eine "GotoNext"-Schleife ohne Interaktion vollständig durchlaufen werden. Anderenfalls kann es, wenn das RecordSet viele Zeilen enthält, am Datenbankserver zu einem *ASYNC_NETWORK_IO*-Wartezustand kommen, der dann andere Abfragen (vor allem Änderungen) auf dieselbe Tabelle blockiert.

Parameter:

Parametername	Typ	Beschreibung

CursorModel	Long	Optional.
		Ermöglicht die Spezifikation des Datenbankcursormodells, das für den zurückgegebenen **RecordSet** genutzt werden soll.
		
		Werte:
		0 (Standardwert): Erzeugt ein **RecordSet** mit einem Datenbankcursormodell, welches innerhalb der combit CRM-Projektdatei spezifiziert werden kann:
		
		`...`
		`<!-- DATA -->`
		`<profile>`
		`<list name="">`
		`<list name="ExtendedSettings">`
		`<item name="COMRecordSetCursorDefaulT">2</item>`
		`</list>`
		`...`
		
		Wird in der combit CRM-Projektdatei keine Eigenschaft **COMRecordSetCursorDefault** gefunden, so wird immer ein fully-dynamic **RecordSet** erzeugt. Mögliche Werte für die Eigenschaft sind: 1 – fully-dynamic RecordSet, 2 – forward-only RecordSet.
		
		1: Erzeugt ein **RecordSet** mit fully-dynamic Datenbankcursor.
		
		2: Erzeugt ein **RecordSet** mit forward-only Datenbankcursor. Ermöglicht deutliche Performance-Gewinne, insbesondere bei großen Datenmengen und komplexen Filterausdrücken, erlaubt aber lediglich das einmalige Durchlaufen in Vorwärtsrichtung durch den **RecordSet**.
		
		Die Methoden **RecordSet.DialogSelectRecord**, **RecordSet.DialogSelectRecordMultiple**, **RecordSet.SendBulkMail** (bei anzuzeigendem integrierten Mail-Editor), **RecordSet.MovePrevious**, **RecordSet.MoveLast**, **InputForm.DialogSelectRecordDropDown** werden einen Scriptfehler werfen, wenn diese für einen forward-only **RecordSet** genutzt werden. Für diese Methoden muss der **RecordSet** explizit ohne forward-only (Werte 0 oder 1) erzeugt werden.

Rückgabewert:

RecordSet

Beispiel VBScript:

```
Dim oRecordSetCopy : Set oRecordSetCopy =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy
' ...
Set oRecordSetCopy = Nothing
```

Beispiel C#-Script:

```
RecordSet recordSetCopy =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy();
// ...
recordSetCopy.Dispose();
```

EditMailTemplate

Beschreibung:

Diese Methode öffnet eine Mailvorlage im Mail-Editor.

Parameter:

Parametername	Typ	Beschreibung
TemplateFileName	String	Pfad der zu öffnenden Mailvorlage. Wenn kein Pfad übergeben wird, dann erfolgt die Dateiauswahl interaktiv.

Beispiel VBScript:

```
Call cRM.CurrentProject.ActiveViews.ActiveView.EditMailTemplate("Pfad der
Mailvorlage")
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.EditMailTemplate("Pfad der
Mailvorlage");
```

EditReportTemplate

Beschreibung:

Diese Methode öffnet eine Druckvorlage im Druckvorlagen-Designer.

Parameter:

Parametername	Typ	Beschreibung
TemplateFileName	String	Pfad der zu öffnenden Druckvorlage. Wenn kein Pfad übergeben wird, dann erfolgt die Dateiauswahl interaktiv.

Beispiel VBScript:

```
Call cRM.CurrentProject.ActiveViews.ActiveView.EditReportTemplate("Pfad der
Druckvorlage")
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.EditReportTemplate("Pfad der
Druckvorlage");
```

InvokeMenu

Beschreibung:

Ruft einen Menüeintrag der Anwendung auf. Neben der ID des Menüeintrages wird angegeben, ob das Script solange warten soll, bis der Befehl abgearbeitet wurde (und evtl. Dialoge geschlossen wurden) oder ob das Script direkt weiterlaufen soll. Die Menü-IDs der Anwendung finden Sie im Kapitel **Menü-IDs**.

Hinweis: Es werden nur Menü-IDs von direkt sichtbaren Menüs unterstützt, d.h. Kontextmenüs können nicht verwendet werden. Sollte die Methode in einem asynchron ausgeführten Script ausgeführt werden, so ist der Rückgabewert immer True. Der Rückgabewert beschreibt, ob der Aufruf übermittelt werden konnte, nicht jedoch, ob in der aufzurufenden Funktion ggf. ein Problem festgestellt wurde.

Parameter:

Parametername	Typ	Beschreibung
MenuID	Long	Die ID des Menüeintrages.
Synchron	Bool	True: synchrone Ausführung False: asynchrone Ausführung

Rückgabewert:

Bool

Wert	Beschreibung
True	Befehl zum Aufrufen eines Menüeintrags wurde erfolgreich an combit CRM übermittelt.
False	Befehl zum Aufrufen eines Menüeintrags konnte nicht übermittelt werden. Dies kann z. B. der Fall sein, wenn der aufzurufende Menü-Befehl derzeit nicht zur Verfügung steht.

Beispiel VBScript:

```
Call cRM.CurrentProject.ActiveViews.ActiveView.InvokeMenu(32781, True) ' Neuen
Datensatz anlegen
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.InvokeMenu(32781, true); // Neuen
Datensatz anlegen
```

StartEditNew

Beschreibung:

Legt in der aktuellen Ansicht einen neuen Datensatz an und wechselt in den Änderungsmodus (analog zum Menüpunkt 'Neuer Datensatz anlegen').

Rückgabewert:

Bool

Wert	Beschreibung
True	Der Datensatz konnte angelegt werden.
False	Das Ausführen der Methode ist fehlgeschlagen.

Beispiel VBScript:

```
Call cRM.CurrentProject.ActiveViews.ActiveView.StartEditNew()
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.StartEditNew();
```

Update

Beschreibung:

Aktualisiert die aktuelle Ansicht. Dabei wird der aktuelle Datensatz neu dargestellt, und die Statusleiste wird ebenfalls aktualisiert. Es sollten zuvor sowohl alle etwaigen von der **View** geholten **RecordSet**-Objekte als auch alle etwaig geholten **Record**-Objekte auf **Nothing** gesetzt werden, keinesfalls dürfen sie anschließend noch benutzt werden.

Rückgabewert:

Bool

Beispiel VBScript:

```
Call cRM.CurrentProject.ActiveViews.ActiveView.Update()
```

Beispiel C#-Script:

```
cRM.CurrentProject.ActiveViews.ActiveView.Update();
```

3.48 ViewConfig Objekt

Hinweis: Aus Sicherheitsgründen ist der Zugriff auf **ViewConfig** Objekte von Ansichten, auf die der aktuelle Benutzer keine Zugriffsrechte besitzt, nicht möglich. Alle diese **ViewConfig** Objekte werden in der übergeordneten Collection (**ListViewConfigs**) für die Eigenschaften/Methoden *Count* und *Item* nicht angeboten. Bei einem versuchten Direktzugriff per *ItemByName* wird kein Objekt zurückgegeben – es erfolgt zudem eine Fehlerausgabe auf das Debug-Tool Debwin.

3.48.1 Eigenschaften

AddressInfos, read-only

Beschreibung:

Liefert ein Objekt vom Typ ListAddressInfos zurück

Typ:

ListAddressInfos

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oListAddressInfos : Set oListAddressInfos =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos
' ...
Set oListAddressInfos = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ListAddressInfos listaddressInfos =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").AddressInfos;
// ...
listaddressInfos.Dispose();
```

DBTableName, read-only

Beschreibung:

Liefert den Datenbank-Tabellennamen der Ansicht zurück.

Typ:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("Der Datenbank-Tabellenname der Ansicht ""Kontakte""
lautet: " & cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").DBTableName,
"ViewConfig.DBTableName", vbOkOnly)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox(@"Der Datenbank-Tabellenname der Ansicht ""Kontakte"" lautet
" + cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").DBTableName,
"ViewConfig.DBTableName", 0);
```

DBTableType, read-only

Beschreibung:

Liefert 0 wenn eine Datenbanktabelle als Basis für die Ansicht verwendet wird, oder 1 bei einer Datenbanksicht.

Typ:

Long

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

If (cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").DBTableType = 0) Then
  Call cRM.DialogMessageBox("Die Grundlage der Ansicht ""Kontakte"" bildet eine
Datenbanktabelle.", "ViewConfig.DBTableType", vbOkOnly)
Else
  Call cRM.DialogMessageBox("Die Grundlage der Ansicht ""Kontakte"" bildet eine
Datenbanksicht.", "ViewConfig.DBTableType", vbOkOnly)
End If
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

if (cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").DBTableType == 0)
{
    cRM.DialogMessageBox(@"Die Grundlage der Ansicht ""Kontakte"" bildet eine
Datenbanktabelle.", "ViewConfig.DBTableType", 0);
}
else
{
    cRM.DialogMessageBox(@"Die Grundlage der Ansicht ""Kontakte"" bildet eine
Datenbanksicht.", "ViewConfig.DBTableType", 0);
}
```

FamilyName, read-only

Beschreibung:

Liefert den definierten Familiennamen der Ansicht zurück.

Typ:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("Der Familienname der Ansicht ""Kontakte"" lautet: " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").FamilyName,
"ViewConfig.FamilyName", vbOkOnly)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox(@"Der Familienname der Ansicht ""Kontakte"" lautet " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").FamilyName,
"ViewConfig.FamilyName", 0);
```

FldCount, read-only

Beschreibung:

Liefert die Anzahl der Felder zurück.

Typ:

Long

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("In der Ansicht ""Kontakte"" wurden " &
CStr(cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").FldCount) & " Felder
konfiguriert.", "ViewConfig.FldCount", vbOkOnly)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox(@"In der Ansicht ""Kontakte"" wurden " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").FldCount.ToString() + "
Felder konfiguriert.", "ViewConfig.FldCount", 0);
```

Name, read-only

Beschreibung:

Liefert den Namen der definierten Ansicht zurück.

Typ:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("Folgender Name wird für die gewählte Ansicht verwendet:
" & cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").Name, "ViewConfig.Name",
vbOkOnly)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox("Folgender Name wird für die gewählte Ansicht verwendet " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").Name, "ViewConfig.Name", 0);
```

PrimaryKeyFldName, read-only

Beschreibung:

Liefert den Namen des Primärschlüsselfeldes, sofern der Primärschlüssel aus genau einem Feld besteht.

Typ:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("Der Name des Primärschlüsselfeldes der Ansicht
""Kontakte"" lautet " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").PrimaryKeyFldName,
"ViewConfig.PrimaryKeyFldName", vbOkOnly)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox(@"Der Name des Primärschlüsselfeldes der Ansicht ""Kontakte""
lautet " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").PrimaryKeyFldName,
"ViewConfig.PrimaryKeyFldName", 0);
```

Relations

Beschreibung:

Liefert ein Objekt vom Typ **ListRelations** zurück.

Typ:

ListRelations

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oListRelations : Set oListRelations =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").Relations
' ...
Set oListRelations = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ListRelations relations =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").Relations;
// ...
relations.Dispose();
```

SortOrderCount, read-only

Beschreibung:

Liefert die Anzahl der definierten Sortierungen zurück.

Typ:

Long

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("Die Anzahl der definierten Sortierungen in der Ansicht
""Kontakte"" beträgt: " &
CStr(cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").SortOrderCount),
"ViewConfig.SortOrderCount", vbOkOnly)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox(@"Die Anzahl der definierten Sortierungen in der Ansicht
""Kontakte"" beträgt: " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").SortOrderCount.ToString(),
"ViewConfig.SortOrderCount", 0);
```

SupportsRecycleBin, read-only

Beschreibung:

Liefert zurück, ob die Ansicht den Papierkorb unterstützt.

Typ:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

If (cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").SupportsRecycleBin =
True) Then
```

```
  Call cRM.DialogMessageBox("Die Ansicht ""Kontakte"" unterstützt die Papierkorb-
Funktion.", "ViewConfig.SupportsRecycleBin", vbOkOnly)
End If
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

if (cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").SupportsRecycleBin ==
true)
{
    cRM.DialogMessageBox(@"Die Ansicht ""Kontakte"" unterstützt die Papierkorb-
Funktion.", "ViewConfig.SupportsRecycleBin", 0);
}
```

3.48.2 Methoden

CodeDefinitions

Beschreibung:

Liefert für das übergebene Feld eine Collection **ListCodeDefinitions** der dafür hinterlegten Codedefinitionen.

Parameter:

Parametername	Typ	Beschreibung
FieldName	String	Physikalischer Feldname des gewünschten Feldes.

Rückgabewert:

ListCodeDefinitions

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oListCodeDefinitions : Set oListCodeDefinitions =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CodeDefinitions("Category")
' ...
Set oListCodeDefinitions = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ListCodeDefinitions codeDefinitions =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CodeDefinitions("Category");
// ...
codeDefinitions.Dispose();
```

CreateRecordSet

Beschreibung:

Erzeugt ein Objekt vom Typ **RecordSet**. Es wird daher keine Ansicht visuell geöffnet.

Hinweis: Aus Sicherheitsgründen ist der Zugriff auf **ViewConfig** Objekte von Ansichten, auf die der aktuelle Benutzer keine Zugriffsrechte besitzt, nicht möglich. Alle diese **ViewConfig** Objekte werden in der übergeordneten Collection (**ListViewConfigs**) für die Eigenschaften/Methoden *Count* und *Item* nicht angeboten. Bei einem versuchten Direktzugriff per *ItemByName* wird kein Objekt zurückgegeben – es erfolgt zudem eine Fehlerausgabe auf das Debug-Tool Debwin.

Wir empfehlen, nach Erzeugung eines **RecordSet**-Objektes zunächst mittles Aufruf der Methode "MoveFirst" **die Existenz mindestens eines Record-Objektes zu überprüfen.**

Wichtiger Hinweis für die Verwendung des Parameters CursorModel mit dem Wert 2 (forward-only) unter Microsoft SQL Server: Die Datensätze eines forward-only-RecordSets müssen nach dessen Erstellung direkt und unmittelbar über eine "GotoNext"-Schleife ohne Interaktion vollständig durchlaufen werden. Anderenfalls kann es, wenn das RecordSet viele Zeilen enthält, am Datenbankserver zu einem *ASYNC_NETWORK_IO*-Wartezustand kommen, der dann andere Abfragen (vor allem Änderungen) auf dieselbe Tabelle blockiert.

Parameter:

Parametername	Typ	Beschreibung
InitialCommand	String	Optional. Ermöglicht die Übergabe einer Sortierung mittels "SetSortOrder:n" (n = Nummer der gewünschten Sortierung, 0 = unsortiert, -n = invertierte Sortierung). Verhindert weitere Datenbankabfragen, die durch das Setzen der *SortOrder*-Eigenschaft ausgeführt werden müssten. Dieses Schlüsselwort muss als erstes angegeben werden, wenn es benutzt werden soll. Nachfolgende SetFilter*-Schlüsselworte können mit Leerzeichen oder Komma angehängt werden. Ein Filter kann mit Leerzeichen oder Komma getrennt dahinter aufgeführt werden. Der Filterausdruck muss dabei mit einer der folgenden Zeichenfolgen beginnen: "SetFilter: <Filterausdruck aus dem Filter Allgemein-Dialog>" "SetFilterByName:<Name für Scripte/Workflows des abgespeicherten Filters oder Pfad zu einer .crmshare-Datei mit enthaltenem Filterausdruck>" "SetFilterDirectSQL:<Freier SQL Filterausdruck>" "SetFilterByPrimaryKey:<Inhalt des Primärschlüsselfeldes>" Ersetzen Sie den Teil <...> durch den entsprechenden Wert.

CursorModel	Long	Optional.
		Ermöglicht die Spezifikation des Datenbankcursormodells, das für den zurückgegebenen **RecordSet** genutzt werden soll.
		
		Werte:
		0 (Standardwert): Erzeugt ein **RecordSet** mit einem Datenbankcursormodell, welches innerhalb der combit CRM-Projektdatei spezifiziert werden kann:
		
		`...`
		`<!-- DATA -->`
		`<profile>`
		`<list name="">`
		`<list name="ExtendedSettings">`
		`<item name="COMRecordSetCursorDefault">2</item>`
		`</list>`
		`...`

		Wird in der combit CRM-Projektdatei keine Eigenschaft **COMRecordSetCursorDefault** gefunden, so wird immer ein fully-dynamic **RecordSet** erzeugt. Mögliche Werte für die Eigenschaft sind: 1 – fully-dynamic RecordSet, 2 – forward-only RecordSet. 1: Erzeugt ein **RecordSet** mit fully-dynamic Datenbankcursor. 2: Erzeugt ein **RecordSet** mit forward-only Datenbankcursor. Ermöglicht deutliche Performance-Gewinne, insbesondere bei großen Datenmengen und komplexen Filterausdrücken, erlaubt aber lediglich das einmalige Durchlaufen in Vorwärtsrichtung durch den **RecordSet**. Die Methoden **RecordSet.DialogSelectRecord**, **RecordSet.DialogSelectRecordMultiple**, **RecordSet.SendBulkMail** (bei anzuzeigendem integrierten Mail-Editor), **RecordSet.MovePrevious**, **RecordSet.MoveLast**, **InputForm.DialogSelectRecordDropDown** werden einen Scriptfehler werfen, wenn diese für einen forward-only **RecordSet** genutzt werden. Für diese Methoden muss der **RecordSet** explizit ohne forward-only (Werte 0 oder 1) erzeugt werden.

Rückgabewert:

RecordSet

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet("SetFilterBy
Name:Kunden_aus_Deutschland")
' ...
Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

RecordSet recordSet =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").CreateRecordSet("SetFilterBy
Name:Kunden_aus_Deutschland");
// ...
recordSet.Dispose();
```

DupeCheckCriteria

Beschreibung:

Liefert die Felder, welche für die Dublettenprüfung verwenden werden, als TAB-separierte Liste zurück.

Rückgabewert:

String

Beispiel VBScript:

```
Dim sFieldsDupeCheck : sFieldsDupeCheck =
cRM.CurrentProject.ActiveViews.ActiveView.Config.DupeCheckCriteria
```

```
Dim dicFieldsDupeCheck : Set dicFieldsDupeCheck =
CreateObject("Scripting.Dictionary")
Dim aFieldsDupeCheck : aFieldsDupeCheck = Split(sFieldsDupeCheck, vbTab)
Dim sField, sUserInput

For Each sField in aFieldsDupeCheck
  sUserInput = cRM.DialogInputBox("Welcher Inhalt soll für das Feld """ & sField &
""" für den Dublettencheck verwendet werden?",
"RecordSet.FindRecordByDupeCheckCriteria")

  If (sUserInput <> "$CANCEL$") Then
      Call dicFieldsDupeCheck.Add(sField, sUserInput)
  Else
      Exit For
  End If
Next

Dim sUserInputInDictionary
Dim sFieldContent1 : sFieldContent1 = ""
Dim sFieldContent2 : sFieldContent2 = ""
Dim sFieldContent3 : sFieldContent3 = ""
Dim sFieldContent4 : sFieldContent4 = ""
Dim sFieldContent5 : sFieldContent5 = ""
Dim sFieldContent6 : sFieldContent6 = ""
Dim sFieldContent7 : sFieldContent7 = ""
Dim sFieldContent8 : sFieldContent8 = ""
Dim nCounter : nCounter = 0

If (dicFieldsDupeCheck.Count > 0) Then
  For Each sUserInputInDictionary in dicFieldsDupeCheck.Keys
      nCounter = nCounter + 1

      Select Case nCounter
          Case 1
              sFieldContent1 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
          Case 2
              sFieldContent2 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
          Case 3
              sFieldContent3 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
          Case 4
              sFieldContent4 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
          Case 5
              sFieldContent5 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
          Case 6
              sFieldContent6 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
          Case 7
              sFieldContent7 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
          Case 8
              sFieldContent8 = dicFieldsDupeCheck.Item(sUserInputInDictionary)
     End Select
  Next
End If

Set dicFieldsDupeCheck = Nothing

Dim oRecordSet : Set oRecordSet =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy
Call oRecordSet.FindRecordByDupeCheckCriteria("", sFieldContent1, sFieldContent2,
sFieldContent3, sFieldContent4, sFieldContent5, sFieldContent6, sFieldContent7,
sFieldContent8, False)

If (Not oRecordSet Is Nothing) Then

  If (oRecordSet.RecCount > 0) Then

      Call cRM.DialogMessageBox("Es wurde mindestens eine Dublette gefunden.",
"RecordSet.FindRecordByDupeCheckCriteria", vbOkOnly)

  End If
```

```
End If

Set oRecordSet = Nothing
```

Beispiel C#-Script:

```
string fieldsDupeCheckCriteria =
cRM.CurrentProject.ActiveViews.ActiveView.Config.DupeCheckCriteria;
System.Collections.Generic.Dictionary<string, string> dicFieldsDupeCheck = new
System.Collections.Generic.Dictionary<string, string>();
string[] fieldsDupeCheck = fieldsDupeCheckCriteria.Split('\t');
string userInput = null;

foreach (string field in fieldsDupeCheck)
{
    userInput = cRM.DialogInputBox(@"Welcher Inhalt soll für das Feld """ + field
+ @""" für den Dublettencheck verwendet werden?",
"RecordSet.FindRecordByDupeCheckCriteria");

    if (userInput != "$CANCEL$")
    {
        dicFieldsDupeCheck.Add(field, userInput);
    }
    else
    {
        break;
    }
}

int counter = 0;
string fieldContent1 = null;
string fieldContent2 = null;
string fieldContent3 = null;
string fieldContent4 = null;
string fieldContent5 = null;
string fieldContent6 = null;
string fieldContent7 = null;
string fieldContent8 = null;
RecordSet recordSet = null;

if (dicFieldsDupeCheck.Count > 0)
{
    foreach (var item in dicFieldsDupeCheck)
    {
        counter++;

        switch (counter)
        {
            case 1:
                fieldContent1 = item.Value;
                break;
            case 2:
                fieldContent2 = item.Value;
                break;
            case 3:
                fieldContent3 = item.Value;
                break;
            case 4:
                fieldContent4 = item.Value;
                break;
            case 5:
                fieldContent5 = item.Value;
                break;
            case 6:
                fieldContent6 = item.Value;
                break;
            case 7:
                fieldContent7 = item.Value;
                break;
            case 8:
                fieldContent8 = item.Value;
```

```
                break;
            default:
                break;
        }
    }

    recordSet = cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSetCopy();
    recordSet.FindRecordByDupeCheckCriteria("", fieldContent1, fieldContent2,
fieldContent3, fieldContent4, fieldContent5, fieldContent6, fieldContent7,
fieldContent8, false);

    if (recordSet != null)
    {
        if (recordSet.RecCount > 0)
        {
            cRM.DialogMessageBox("Es wurde mindestens eine Dublette gefunden.",
"RecordSet.FindRecordByDupeCheckCriteria", 0);
            recordSet.Dispose();
        }
    }
}
```

FldAlias

Beschreibung:

Liefert den Aliasnamen des Feldes zurück, das als fortlaufende Nummer übergeben wurde. Ist der Aliasname leer, wird der entsprechende physikalische Feldname zurückgegeben.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Die fortlaufende Nummer des gewünschten Feldes in der Datenbankstruktur von 1 bis FldCount.

Rückgabewert:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
Dim nFldCount : nFldCount = oViewConfig.FldCount
Dim nCounter : nCounter = 0

Dim sFldAlias : sFldAlias = ""
Dim nFldDec : nFldDec = 0
Dim bFldGDPRActive : bFldGDPRActive = False
Dim sFldName : sFldName = ""
Dim nFldLen : nFldLen = 0
Dim bFldReadOnly : bFldReadOnly = False
Dim nFldType : nFldType = 0
Dim nFldTypePhys : nFldTypePhys = 0
Dim sFldTypePhysNative : sFldTypePhysNative = ""

For nCounter = 1 To nFldCount

    sFldAlias = oViewConfig.FldAlias(nCounter)
    nFldDec = oViewConfig.FldDec(nCounter)
    bFldGDPRActive = oViewConfig.FldGDPRActive(nCounter)
    sFldName = oViewConfig.FldName(nCounter)
    nFldLen = oViewConfig.FldLen(nCounter)
    bFldReadOnly = oViewConfig.FldReadOnly(nCounter)
    nFldType = oViewConfig.FldType(nCounter)
    nFldTypePhys = oViewConfig.FldTypePhys(nCounter)
    sFldTypePhysNative = oViewConfig.FldTypePhysNative(nCounter)
```

```
    If (bFldGDPRActive = True) Then
        Call cRM.DialogMessageBox("Das physikalische Feld " & sFldName & " (Alias:
" & sFldAlias & ", Länge: " & CStr(nFldLen) & ") hat folgende Typen: " &
CStr(nFldType) & " - " & CStr(nFldTypePhys) & " - " & sFldTypePhysNative,
"ViewConfig", vbOkOnly)
    End If

Next

Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
long fldCount = viewConfig.FldCount;
int counter = 0;

string fldAlias = null;
long fldDec = 0;
bool fldGDPRActive = false;
string fldName = null;
long fldLen = 0;
bool fldReadOnly = false;
long fldType = 0;
long fldTypePhys = 0;
string fldTypePhysNative = null;

for (counter = 1; counter <= fldCount; counter++)
{
    fldAlias = viewConfig.FldAlias(counter);
    fldDec = viewConfig.FldDec(counter);
    fldGDPRActive = viewConfig.FldGDPRActive(counter);
    fldName = viewConfig.FldName(counter);
    fldLen = viewConfig.FldLen(counter);
    fldReadOnly = viewConfig.FldReadOnly(counter);
    fldType = viewConfig.FldType(counter);
    fldTypePhys = viewConfig.FldTypePhys(counter);
    fldTypePhysNative = viewConfig.FldTypePhysNative(counter);

    if (fldGDPRActive == true)
    {
        cRM.DialogMessageBox("Das physikalische Feld " + fldName + " (Alias " +
fldAlias + ", Länge: " + fldLen.ToString() + ") hat folgende Typen " +
fldType.ToString() + " - " + fldTypePhys.ToString() + " - " + fldTypePhysNative,
"ViewConfig", 0);

    }
}

viewConfig.Dispose();
```

FldAliasByName

Beschreibung:

Liefert den Aliasnamen des Feldes zurück, dessen physikalischer Feldname übergeben wurde. Ist der Aliasname leer, wird der entsprechende physikalische Feldname zurückgegeben.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Physikalischer Feldname des gewünschten Feldes.

Rückgabewert:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("Der Alias des Feldnamens ""ZIP_Private"" der Ansicht
""Kontakte"" lautet " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").FldAliasByName("ZIP_Private"
), "ViewConfig.FldAliasByName", vbOkOnly)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox(@"Der Alias des Feldnamens ""ZIP_Private"" der Ansicht
""Kontakte"" lautet " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").FldAliasByName("ZIP_Private"
), "ViewConfig.FldAliasByName", 0);
```

FldCaseSensitive

Beschreibung:

Überprüft, ob bei einem übergebenen Feld die Groß- und Kleinschreibung beachtet wird. Ist der Rückgabewert False kann innerhalb von Scripten bei dynamischen Filterausdrücken auf UPPER für das jeweilige Feld verzichtet werden. Dies ermöglicht eine Verbesserung der Performance.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Die fortlaufende Nummer des gewünschten Feldes in der Datenbankstruktur von 1 bis FldCount.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim bReturn : bReturn =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").FldCaseSensitive(nIndex)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

bool return =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").FldCaseSensitive(index);
```

FldDec

Beschreibung:

Liefert die Anzahl der Dezimalstellen des Feldes zurück, das als fortlaufende Nummer übergeben wurde.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Die fortlaufende Nummer des gewünschten Feldes in der Datenbankstruktur von 1 bis FldCount.

Rückgabewert:

Long

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution
```

```
Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
Dim nFldCount : nFldCount = oViewConfig.FldCount
Dim nCounter : nCounter = 0

Dim sFldAlias : sFldAlias = ""
Dim nFldDec : nFldDec = 0
Dim bFldGDPRActive : bFldGDPRActive = False
Dim sFldName : sFldName = ""
Dim nFldLen : nFldLen = 0
Dim bFldReadOnly : bFldReadOnly = False
Dim nFldType : nFldType = 0
Dim nFldTypePhys : nFldTypePhys = 0
Dim sFldTypePhysNative : sFldTypePhysNative = ""

For nCounter = 1 To nFldCount

    sFldAlias = oViewConfig.FldAlias(nCounter)
    nFldDec = oViewConfig.FldDec(nCounter)
    bFldGDPRActive = oViewConfig.FldGDPRActive(nCounter)
    sFldName = oViewConfig.FldName(nCounter)
    nFldLen = oViewConfig.FldLen(nCounter)
    bFldReadOnly = oViewConfig.FldReadOnly(nCounter)
    nFldType = oViewConfig.FldType(nCounter)
    nFldTypePhys = oViewConfig.FldTypePhys(nCounter)
    sFldTypePhysNative = oViewConfig.FldTypePhysNative(nCounter)

    If (bFldGDPRActive = True) Then
        Call cRM.DialogMessageBox("Das physikalische Feld " & sFldName & " (Alias:
" & sFldAlias & ", Länge: " & CStr(nFldLen) & ") hat folgende Typen: " &
CStr(nFldType) & " - " & CStr(nFldTypePhys) & " - " & sFldTypePhysNative,
"ViewConfig", vbOkOnly)
    End If

Next

Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
long fldCount = viewConfig.FldCount;
int counter = 0;

string fldAlias = null;
long fldDec = 0;
bool fldGDPRActive = false;
string fldName = null;
long fldLen = 0;
bool fldReadOnly = false;
long fldType = 0;
long fldTypePhys = 0;
string fldTypePhysNative = null;

for (counter = 1; counter <= fldCount; counter++)
{
    fldAlias = viewConfig.FldAlias(counter);
    fldDec = viewConfig.FldDec(counter);
    fldGDPRActive = viewConfig.FldGDPRActive(counter);
    fldName = viewConfig.FldName(counter);
    fldLen = viewConfig.FldLen(counter);
    fldReadOnly = viewConfig.FldReadOnly(counter);
    fldType = viewConfig.FldType(counter);
    fldTypePhys = viewConfig.FldTypePhys(counter);
    fldTypePhysNative = viewConfig.FldTypePhysNative(counter);

    if (fldGDPRActive == true)
    {
```

```
        cRM.DialogMessageBox("Das physikalische Feld " + fldName + " (Alias " +
fldAlias + ", Länge: " + fldLen.ToString() + ") hat folgende Typen " +
fldType.ToString() + " - " + fldTypePhys.ToString() + " - " + fldTypePhysNative,
"ViewConfig", 0);

    }
}

viewConfig.Dispose();
```

FldExists

Beschreibung:

Überprüft, ob ein Feld in der Ansicht existiert.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Übergabe des zu überprüfenden Feldnamens.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

If (cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").FldExists("Name") =
True) Then
    Call cRM.DialogMessageBox("Das Feld mit dem physikalischen Feldnamen ""Name""
existiert innerhalb der ""Kontakte""-Ansicht.", "ViewConfig.FldExists", vbOkOnly)
End If
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

if (cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").FldExists("Name") ==
true)
{
    cRM.DialogMessageBox(@"Das Feld mit dem physikalischen Feldnamen ""Name""
existiert innerhalb der ""Kontakte""-Ansicht.", "ViewConfig.FldExists", 0);
}
```

FldGDPRActive

Beschreibung:

Liefert den Datenschutz-Status des Feldes zurück, das als fortlaufende Nummer übergeben wurde.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Die fortlaufende Nummer des gewünschten Feldes in der Datenbankstruktur von 1 bis FldCount.

Rückgabewert:

Bool

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
Dim nFldCount : nFldCount = oViewConfig.FldCount
Dim nCounter : nCounter = 0
```

```
Dim sFldAlias : sFldAlias = ""
Dim nFldDec : nFldDec = 0
Dim bFldGDPRActive : bFldGDPRActive = False
Dim sFldName : sFldName = ""
Dim nFldLen : nFldLen = 0
Dim bFldReadOnly : bFldReadOnly = False
Dim nFldType : nFldType = 0
Dim nFldTypePhys : nFldTypePhys = 0
Dim sFldTypePhysNative : sFldTypePhysNative = ""

For nCounter = 1 To nFldCount

    sFldAlias = oViewConfig.FldAlias(nCounter)
    nFldDec = oViewConfig.FldDec(nCounter)
    bFldGDPRActive = oViewConfig.FldGDPRActive(nCounter)
    sFldName = oViewConfig.FldName(nCounter)
    nFldLen = oViewConfig.FldLen(nCounter)
    bFldReadOnly = oViewConfig.FldReadOnly(nCounter)
    nFldType = oViewConfig.FldType(nCounter)
    nFldTypePhys = oViewConfig.FldTypePhys(nCounter)
    sFldTypePhysNative = oViewConfig.FldTypePhysNative(nCounter)

    If (bFldGDPRActive = True) Then
        Call cRM.DialogMessageBox("Das physikalische Feld " & sFldName & " (Alias:
" & sFldAlias & ", Länge: " & CStr(nFldLen) & ") hat folgende Typen: " &
CStr(nFldType) & " - " & CStr(nFldTypePhys) & " - " & sFldTypePhysNative,
"ViewConfig", vbOkOnly)
    End If

Next

Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
long fldCount = viewConfig.FldCount;
int counter = 0;

string fldAlias = null;
long fldDec = 0;
bool fldGDPRActive = false;
string fldName = null;
long fldLen = 0;
bool fldReadOnly = false;
long fldType = 0;
long fldTypePhys = 0;
string fldTypePhysNative = null;

for (counter = 1; counter <= fldCount; counter++)
{
    fldAlias = viewConfig.FldAlias(counter);
    fldDec = viewConfig.FldDec(counter);
    fldGDPRActive = viewConfig.FldGDPRActive(counter);
    fldName = viewConfig.FldName(counter);
    fldLen = viewConfig.FldLen(counter);
    fldReadOnly = viewConfig.FldReadOnly(counter);
    fldType = viewConfig.FldType(counter);
    fldTypePhys = viewConfig.FldTypePhys(counter);
    fldTypePhysNative = viewConfig.FldTypePhysNative(counter);

    if (fldGDPRActive == true)
    {
        cRM.DialogMessageBox("Das physikalische Feld " + fldName + " (Alias " +
fldAlias + ", Länge: " + fldLen.ToString() + ") hat folgende Typen " +
fldType.ToString() + " - " + fldTypePhys.ToString() + " - " + fldTypePhysNative,
"ViewConfig", 0);

    }
```

```
}

viewConfig.Dispose();
```

FldIndex

Beschreibung:

Liefert den Index des Feldes zurück.

Parameter:

Parametername	Typ	Beschreibung
FieldName	String	Physikalischer Name des Feldes.

Rückgabewert:

Long

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
Dim nFldIndex : nFldIndex = oViewConfig.FldIndex("Firstname")
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
long fldIndex = viewConfig.FldIndex("Firstname");
```

FldName

Beschreibung:

Liefert den Namen des Feldes zurück, das als fortlaufende Nummer übergeben wurde.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Die fortlaufende Nummer des gewünschten Feldes in der Datenbankstruktur von 1 bis FldCount.

Rückgabewert:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
Dim nFldCount : nFldCount = oViewConfig.FldCount
Dim nCounter : nCounter = 0

Dim sFldAlias : sFldAlias = ""
Dim nFldDec : nFldDec = 0
Dim bFldGDPRActive : bFldGDPRActive = False
Dim sFldName : sFldName = ""
Dim nFldLen : nFldLen = 0
Dim bFldReadOnly : bFldReadOnly = False
Dim nFldType : nFldType = 0
Dim nFldTypePhys : nFldTypePhys = 0
Dim sFldTypePhysNative : sFldTypePhysNative = ""

For nCounter = 1 To nFldCount
```

```
    sFldAlias = oViewConfig.FldAlias(nCounter)
    nFldDec = oViewConfig.FldDec(nCounter)
    bFldGDPRActive = oViewConfig.FldGDPRActive(nCounter)
    sFldName = oViewConfig.FldName(nCounter)
    nFldLen = oViewConfig.FldLen(nCounter)
    bFldReadOnly = oViewConfig.FldReadOnly(nCounter)
    nFldType = oViewConfig.FldType(nCounter)
    nFldTypePhys = oViewConfig.FldTypePhys(nCounter)
    sFldTypePhysNative = oViewConfig.FldTypePhysNative(nCounter)

    If (bFldGDPRActive = True) Then
        Call cRM.DialogMessageBox("Das physikalische Feld " & sFldName & " (Alias:
" & sFldAlias & ", Länge: " & CStr(nFldLen) & ") hat folgende Typen: " &
CStr(nFldType) & " - " & CStr(nFldTypePhys) & " - " & sFldTypePhysNative,
"ViewConfig", vbOkOnly)
    End If

Next

Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
long fldCount = viewConfig.FldCount;
int counter = 0;

string fldAlias = null;
long fldDec = 0;
bool fldGDPRActive = false;
string fldName = null;
long fldLen = 0;
bool fldReadOnly = false;
long fldType = 0;
long fldTypePhys = 0;
string fldTypePhysNative = null;

for (counter = 1; counter <= fldCount; counter++)
{
    fldAlias = viewConfig.FldAlias(counter);
    fldDec = viewConfig.FldDec(counter);
    fldGDPRActive = viewConfig.FldGDPRActive(counter);
    fldName = viewConfig.FldName(counter);
    fldLen = viewConfig.FldLen(counter);
    fldReadOnly = viewConfig.FldReadOnly(counter);
    fldType = viewConfig.FldType(counter);
    fldTypePhys = viewConfig.FldTypePhys(counter);
    fldTypePhysNative = viewConfig.FldTypePhysNative(counter);

    if (fldGDPRActive == true)
    {
        cRM.DialogMessageBox("Das physikalische Feld " + fldName + " (Alias " +
fldAlias + ", Länge: " + fldLen.ToString() + ") hat folgende Typen " +
fldType.ToString() + " - " + fldTypePhys.ToString() + " - " + fldTypePhysNative,
"ViewConfig", 0);

    }
}

viewConfig.Dispose();
```

FldLen

Beschreibung:

Liefert die Zeichenlänge des Feldes zurück, das als fortlaufende Nummer übergeben wurde.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Die fortlaufende Nummer des gewünschten Feldes in der Datenbankstruktur von 1 bis FldCount.

Rückgabewert:

Long

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
Dim nFldCount : nFldCount = oViewConfig.FldCount
Dim nCounter : nCounter = 0

Dim sFldAlias : sFldAlias = ""
Dim nFldDec : nFldDec = 0
Dim bFldGDPRActive : bFldGDPRActive = False
Dim sFldName : sFldName = ""
Dim nFldLen : nFldLen = 0
Dim bFldReadOnly : bFldReadOnly = False
Dim nFldType : nFldType = 0
Dim nFldTypePhys : nFldTypePhys = 0
Dim sFldTypePhysNative : sFldTypePhysNative = ""

For nCounter = 1 To nFldCount

    sFldAlias = oViewConfig.FldAlias(nCounter)
    nFldDec = oViewConfig.FldDec(nCounter)
    bFldGDPRActive = oViewConfig.FldGDPRActive(nCounter)
    sFldName = oViewConfig.FldName(nCounter)
    nFldLen = oViewConfig.FldLen(nCounter)
    bFldReadOnly = oViewConfig.FldReadOnly(nCounter)
    nFldType = oViewConfig.FldType(nCounter)
    nFldTypePhys = oViewConfig.FldTypePhys(nCounter)
    sFldTypePhysNative = oViewConfig.FldTypePhysNative(nCounter)

    If (bFldGDPRActive = True) Then
        Call cRM.DialogMessageBox("Das physikalische Feld " & sFldName & " (Alias:
" & sFldAlias & ", Länge: " & CStr(nFldLen) & ") hat folgende Typen: " &
CStr(nFldType) & " - " & CStr(nFldTypePhys) & " - " & sFldTypePhysNative,
"ViewConfig", vbOkOnly)
    End If

Next

Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
long fldCount = viewConfig.FldCount;
int counter = 0;

string fldAlias = null;
long fldDec = 0;
bool fldGDPRActive = false;
string fldName = null;
long fldLen = 0;
```

```
bool fldReadOnly = false;
long fldType = 0;
long fldTypePhys = 0;
string fldTypePhysNative = null;

for (counter = 1; counter <= fldCount; counter++)
{
    fldAlias = viewConfig.FldAlias(counter);
    fldDec = viewConfig.FldDec(counter);
    fldGDPRActive = viewConfig.FldGDPRActive(counter);
    fldName = viewConfig.FldName(counter);
    fldLen = viewConfig.FldLen(counter);
    fldReadOnly = viewConfig.FldReadOnly(counter);
    fldType = viewConfig.FldType(counter);
    fldTypePhys = viewConfig.FldTypePhys(counter);
    fldTypePhysNative = viewConfig.FldTypePhysNative(counter);

    if (fldGDPRActive == true)
    {
        cRM.DialogMessageBox("Das physikalische Feld " + fldName + " (Alias " +
fldAlias + ", Länge: " + fldLen.ToString() + ") hat folgende Typen " +
fldType.ToString() + " - " + fldTypePhys.ToString() + " - " + fldTypePhysNative,
"ViewConfig", 0);

    }
}

viewConfig.Dispose();
```

FldReadOnly

Beschreibung:

Überprüft, ob ein Feld gesetzt (verändert) werden darf.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Die fortlaufende Nummer des gewünschten Feldes in der Datenbankstruktur von 1 bis FldCount.

Rückgabewert:

Bool

Wert	Beschreibung
True	Das Feld ist schreibgeschützt und kann nicht überschrieben werden.
False	Das Feld ist nicht schreibgeschützt und kann überschrieben werden.

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
Dim nFldCount : nFldCount = oViewConfig.FldCount
Dim nCounter : nCounter = 0

Dim sFldAlias : sFldAlias = ""
Dim nFldDec : nFldDec = 0
Dim bFldGDPRActive : bFldGDPRActive = False
Dim sFldName : sFldName = ""
Dim nFldLen : nFldLen = 0
Dim bFldReadOnly : bFldReadOnly = False
Dim nFldType : nFldType = 0
```

```
Dim nFldTypePhys : nFldTypePhys = 0
Dim sFldTypePhysNative : sFldTypePhysNative = ""

For nCounter = 1 To nFldCount

    sFldAlias = oViewConfig.FldAlias(nCounter)
    nFldDec = oViewConfig.FldDec(nCounter)
    bFldGDPRActive = oViewConfig.FldGDPRActive(nCounter)
    sFldName = oViewConfig.FldName(nCounter)
    nFldLen = oViewConfig.FldLen(nCounter)
    bFldReadOnly = oViewConfig.FldReadOnly(nCounter)
    nFldType = oViewConfig.FldType(nCounter)
    nFldTypePhys = oViewConfig.FldTypePhys(nCounter)
    sFldTypePhysNative = oViewConfig.FldTypePhysNative(nCounter)

    If (bFldGDPRActive = True) Then
        Call cRM.DialogMessageBox("Das physikalische Feld " & sFldName & " (Alias:
" & sFldAlias & ", Länge: " & CStr(nFldLen) & ") hat folgende Typen: " &
CStr(nFldType) & " - " & CStr(nFldTypePhys) & " - " & sFldTypePhysNative,
"ViewConfig", vbOkOnly)
    End If

Next

Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
long fldCount = viewConfig.FldCount;
int counter = 0;

string fldAlias = null;
long fldDec = 0;
bool fldGDPRActive = false;
string fldName = null;
long fldLen = 0;
bool fldReadOnly = false;
long fldType = 0;
long fldTypePhys = 0;
string fldTypePhysNative = null;

for (counter = 1; counter <= fldCount; counter++)
{
    fldAlias = viewConfig.FldAlias(counter);
    fldDec = viewConfig.FldDec(counter);
    fldGDPRActive = viewConfig.FldGDPRActive(counter);
    fldName = viewConfig.FldName(counter);
    fldLen = viewConfig.FldLen(counter);
    fldReadOnly = viewConfig.FldReadOnly(counter);
    fldType = viewConfig.FldType(counter);
    fldTypePhys = viewConfig.FldTypePhys(counter);
    fldTypePhysNative = viewConfig.FldTypePhysNative(counter);

    if (fldGDPRActive == true)
    {
        cRM.DialogMessageBox("Das physikalische Feld " + fldName + " (Alias " +
fldAlias + ", Länge: " + fldLen.ToString() + ") hat folgende Typen " +
fldType.ToString() + " - " + fldTypePhys.ToString() + " - " + fldTypePhysNative,
"ViewConfig", 0);

    }
}

viewConfig.Dispose();
```

FldType

Beschreibung:

Liefert den internen Typ des Feldes zurück, das als fortlaufende Nummer übergeben wurde.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Die fortlaufende Nummer des gewünschten Feldes in der Datenbankstruktur von 1 bis FldCount.

Rückgabewert:

Long

Interne Feldtypen der Anwendung:

Wert	Feldtyp intern
0	Unbekannt
1	Logisch
2	Zeichen
3	Notizen
4	Numerisch
5	Datum mit Zeit
6	Datum
7	Zeit
8	Numerisch binary
9	Telefon
10	Mobiltelefon
11	Internet
12	E-Mail
13	Code
14	Global eindeutige ID
15	Postleitzahl
16	Straße
17	Ort
18	Land
19	Bundesland
20	Bankleitzahl
21	Datensatz-ID
22	Grafikverweis
23	Dateiverweis
24	Automatische Nr.
-	reserviert
27	Telefax
28	Postfach Postleitzahl
29	Postfach
30	Erfassungsbenutzer
31	Erfassungsdatum
32	Änderungsbenutzer
33	Änderungsdatum
34	Autosequenz
35	Symbol
-	Internal
37	Eingebettete Datei
38	Eingebettete Grafik

39	DMS Dokument
40	intern
41	Belegverweis (FM) – Format: fmw://<Mandant-name>/<Belegtyp>/<BelegNummer>"
43	Notizen formatiert
44	Papierkorb-ID

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
Dim nFldCount : nFldCount = oViewConfig.FldCount
Dim nCounter : nCounter = 0

Dim sFldAlias : sFldAlias = ""
Dim nFldDec : nFldDec = 0
Dim bFldGDPRActive : bFldGDPRActive = False
Dim sFldName : sFldName = ""
Dim nFldLen : nFldLen = 0
Dim bFldReadOnly : bFldReadOnly = False
Dim nFldType : nFldType = 0
Dim nFldTypePhys : nFldTypePhys = 0
Dim sFldTypePhysNative : sFldTypePhysNative = ""

For nCounter = 1 To nFldCount

    sFldAlias = oViewConfig.FldAlias(nCounter)
    nFldDec = oViewConfig.FldDec(nCounter)
    bFldGDPRActive = oViewConfig.FldGDPRActive(nCounter)
    sFldName = oViewConfig.FldName(nCounter)
    nFldLen = oViewConfig.FldLen(nCounter)
    bFldReadOnly = oViewConfig.FldReadOnly(nCounter)
    nFldType = oViewConfig.FldType(nCounter)
    nFldTypePhys = oViewConfig.FldTypePhys(nCounter)
    sFldTypePhysNative = oViewConfig.FldTypePhysNative(nCounter)

    If (bFldGDPRActive = True) Then
        Call cRM.DialogMessageBox("Das physikalische Feld " & sFldName & " (Alias:
" & sFldAlias & ", Länge: " & CStr(nFldLen) & ") hat folgende Typen: " &
CStr(nFldType) & " - " & CStr(nFldTypePhys) & " - " & sFldTypePhysNative,
"ViewConfig", vbOkOnly)
    End If

Next

Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
long fldCount = viewConfig.FldCount;
int counter = 0;

string fldAlias = null;
long fldDec = 0;
bool fldGDPRActive = false;
string fldName = null;
long fldLen = 0;
bool fldReadOnly = false;
long fldType = 0;
long fldTypePhys = 0;
string fldTypePhysNative = null;

for (counter = 1; counter <= fldCount; counter++)
{
    fldAlias = viewConfig.FldAlias(counter);
    fldDec = viewConfig.FldDec(counter);
```

```
        fldGDPRActive = viewConfig.FldGDPRActive(counter);
        fldName = viewConfig.FldName(counter);
        fldLen = viewConfig.FldLen(counter);
        fldReadOnly = viewConfig.FldReadOnly(counter);
        fldType = viewConfig.FldType(counter);
        fldTypePhys = viewConfig.FldTypePhys(counter);
        fldTypePhysNative = viewConfig.FldTypePhysNative(counter);

        if (fldGDPRActive == true)
        {
            cRM.DialogMessageBox("Das physikalische Feld " + fldName + " (Alias " +
fldAlias + ", Länge: " + fldLen.ToString() + ") hat folgende Typen " +
fldType.ToString() + " - " + fldTypePhys.ToString() + " - " + fldTypePhysNative,
"ViewConfig", 0);

        }
    }

    viewConfig.Dispose();
```

FldTypePhys

Beschreibung:

Liefert den physikalischen Feldtyp zurück.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Die fortlaufende Nummer des gewünschten Feldes in der Datenbankstruktur von 1 bis FldCount.

Rückgabewert:

Long

Wert	Feldtyp
0	Unbekannt
1	Logisch
2	Ganzzahl kurz
3	Ganzzahl lang
4	Rationale Zahl
5	Numerisch
6	Datum mit Zeit
7	Zeichen
8	Bytes
9	Binär lang
10	Zeichen lang
11	BLOB
12	CLOB
15	Ganzzahl kurz (ohne Vorzeichen)
16	Ganzzahl lang (ohne Vorzeichen)
18	Ganzzahl 64-Bit
19	Ganzzahl 64-Bit (ohne Vorzeichen)

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
Dim nFldCount : nFldCount = oViewConfig.FldCount
Dim nCounter : nCounter = 0
```

```
Dim sFldAlias : sFldAlias = ""
Dim nFldDec : nFldDec = 0
Dim bFldGDPRActive : bFldGDPRActive = False
Dim sFldName : sFldName = ""
Dim nFldLen : nFldLen = 0
Dim bFldReadOnly : bFldReadOnly = False
Dim nFldType : nFldType = 0
Dim nFldTypePhys : nFldTypePhys = 0
Dim sFldTypePhysNative : sFldTypePhysNative = ""

For nCounter = 1 To nFldCount

    sFldAlias = oViewConfig.FldAlias(nCounter)
    nFldDec = oViewConfig.FldDec(nCounter)
    bFldGDPRActive = oViewConfig.FldGDPRActive(nCounter)
    sFldName = oViewConfig.FldName(nCounter)
    nFldLen = oViewConfig.FldLen(nCounter)
    bFldReadOnly = oViewConfig.FldReadOnly(nCounter)
    nFldType = oViewConfig.FldType(nCounter)
    nFldTypePhys = oViewConfig.FldTypePhys(nCounter)
    sFldTypePhysNative = oViewConfig.FldTypePhysNative(nCounter)

    If (bFldGDPRActive = True) Then
        Call cRM.DialogMessageBox("Das physikalische Feld " & sFldName & " (Alias:
" & sFldAlias & ", Länge: " & CStr(nFldLen) & ") hat folgende Typen: " &
CStr(nFldType) & " - " & CStr(nFldTypePhys) & " - " & sFldTypePhysNative,
"ViewConfig", vbOkOnly)
    End If

Next

Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
long fldCount = viewConfig.FldCount;
int counter = 0;

string fldAlias = null;
long fldDec = 0;
bool fldGDPRActive = false;
string fldName = null;
long fldLen = 0;
bool fldReadOnly = false;
long fldType = 0;
long fldTypePhys = 0;
string fldTypePhysNative = null;

for (counter = 1; counter <= fldCount; counter++)
{
    fldAlias = viewConfig.FldAlias(counter);
    fldDec = viewConfig.FldDec(counter);
    fldGDPRActive = viewConfig.FldGDPRActive(counter);
    fldName = viewConfig.FldName(counter);
    fldLen = viewConfig.FldLen(counter);
    fldReadOnly = viewConfig.FldReadOnly(counter);
    fldType = viewConfig.FldType(counter);
    fldTypePhys = viewConfig.FldTypePhys(counter);
    fldTypePhysNative = viewConfig.FldTypePhysNative(counter);

    if (fldGDPRActive == true)
    {
        cRM.DialogMessageBox("Das physikalische Feld " + fldName + " (Alias " +
fldAlias + ", Länge: " + fldLen.ToString() + ") hat folgende Typen " +
fldType.ToString() + " - " + fldTypePhys.ToString() + " - " + fldTypePhysNative,
"ViewConfig", 0);
```

```
    }
}

viewConfig.Dispose();
```

FldTypePhysNative

Beschreibung:

Liefert den physikalischen Namen des Feldtyps des übergebenen Feldes zurück.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Die fortlaufende Nummer des gewünschten Feldes in der Datenbankstruktur von 1 bis FldCount.

Rückgabewert:

String

Microsoft SQL

Feldtyp
bigint
binary
bit
char
date
datetime
datetime2
datetimeoffset
decimal
float
image
int
money
nchar
ntext
numeric
nvarchar
NVARCHARMAX
real
smallint
sql_variant
text
timestamp
tinyint
Unbekannt
uniqueidentifier
varbinary
VARBINARYMAX
varchar
VARCHARMAX

PostgreSQL

Feldtyp

abstime
aclitem
bigint
binary
bit
bit
bool
bytea
char
cid
cidr
circle
date
datetime
datetime2
datetimeoffset
decimal
float
float4
float8
image
inet
int
int2
int4
int8
interval
macaddr
money
nchar
ntext
numeric
nvarchar
nvarchar(max)
oid
path
polygon
real
smallint
sql_variant
text
text[]
time
time with time zone
timestamp
timestamptz
tinyint
Unbekannt
uniqueidentifier
varbinary
varbinary(max)
varbit
varchar

varchar(max)

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Dim oViewConfig : Set oViewConfig =
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte")
Dim nFldCount : nFldCount = oViewConfig.FldCount
Dim nCounter : nCounter = 0

Dim sFldAlias : sFldAlias = ""
Dim nFldDec : nFldDec = 0
Dim bFldGDPRActive : bFldGDPRActive = False
Dim sFldName : sFldName = ""
Dim nFldLen : nFldLen = 0
Dim bFldReadOnly : bFldReadOnly = False
Dim nFldType : nFldType = 0
Dim nFldTypePhys : nFldTypePhys = 0
Dim sFldTypePhysNative : sFldTypePhysNative = ""

For nCounter = 1 To nFldCount

    sFldAlias = oViewConfig.FldAlias(nCounter)
    nFldDec = oViewConfig.FldDec(nCounter)
    bFldGDPRActive = oViewConfig.FldGDPRActive(nCounter)
    sFldName = oViewConfig.FldName(nCounter)
    nFldLen = oViewConfig.FldLen(nCounter)
    bFldReadOnly = oViewConfig.FldReadOnly(nCounter)
    nFldType = oViewConfig.FldType(nCounter)
    nFldTypePhys = oViewConfig.FldTypePhys(nCounter)
    sFldTypePhysNative = oViewConfig.FldTypePhysNative(nCounter)

    If (bFldGDPRActive = True) Then
        Call cRM.DialogMessageBox("Das physikalische Feld " & sFldName & " (Alias:
" & sFldAlias & ", Länge: " & CStr(nFldLen) & ") hat folgende Typen: " &
CStr(nFldType) & " - " & CStr(nFldTypePhys) & " - " & sFldTypePhysNative,
"ViewConfig", vbOkOnly)
    End If

Next

Set oViewConfig = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

ViewConfig viewConfig = cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte");
long fldCount = viewConfig.FldCount;
int counter = 0;

string fldAlias = null;
long fldDec = 0;
bool fldGDPRActive = false;
string fldName = null;
long fldLen = 0;
bool fldReadOnly = false;
long fldType = 0;
long fldTypePhys = 0;
string fldTypePhysNative = null;

for (counter = 1; counter <= fldCount; counter++)
{
    fldAlias = viewConfig.FldAlias(counter);
    fldDec = viewConfig.FldDec(counter);
    fldGDPRActive = viewConfig.FldGDPRActive(counter);
    fldName = viewConfig.FldName(counter);
    fldLen = viewConfig.FldLen(counter);
    fldReadOnly = viewConfig.FldReadOnly(counter);
    fldType = viewConfig.FldType(counter);
    fldTypePhys = viewConfig.FldTypePhys(counter);
```

```
        fldTypePhysNative = viewConfig.FldTypePhysNative(counter);

        if (fldGDPRActive == true)
        {
            cRM.DialogMessageBox("Das physikalische Feld " + fldName + " (Alias " +
fldAlias + ", Länge: " + fldLen.ToString() + ") hat folgende Typen " +
fldType.ToString() + " - " + fldTypePhys.ToString() + " - " + fldTypePhysNative,
"ViewConfig", 0);

        }
    }

    viewConfig.Dispose();
```

RowIDFldName

Beschreibung:

Liefert die echte RowID-Spalte zurück.

Rückgabewert:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("Der Name des Feldes der RowID-Spalte lautet " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").RowIDFldName,
"ViewConfig.RowIDFldName", vbOkOnly)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox("Der Name des Feldes der RowID-Spalte lautet " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").RowIDFldName,
"ViewConfig.RowIDFldName", 0);
```

SortOrderName

Beschreibung:

Liefert anhand der übergebenen Sortierung den Namen der Sortierung zurück.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Übergabe der Sortierung 0 = <ohne Sortierung>

Rückgabewert:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

Call cRM.DialogMessageBox("Der Name der Sortierung mit dem Index 2 lautet " &
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").SortOrderName(2),
"ViewConfig.SortOrderName", vbOkOnly)
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der Kontakte-Ansicht einer combit_Large-Solution

cRM.DialogMessageBox("Der Name der Sortierung mit dem Index 2 lautet " +
cRM.CurrentProject.ViewConfigs.ItemByName("Kontakte").SortOrderName(2),
"ViewConfig.SortOrderName", 0);
```

3.49 WebElement Objekt

Bietet Zugriff auf ein Web-Element.

3.49.1 Eigenschaften

ID

Beschreibung:

Liefert die ID des Web-Elements zurück.

Typ:

String

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der WebElemente-Ansicht einer combit_Large-Solution

<!--#pragma keepeditmode-->
Dim oListWebElements : Set oListWebElements =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).WebElements
Dim oWebElement : Set oWebElement = oListWebElements.ItemByName("{64BBFFD7-EA32-
4358-BBFC-744D6A94291D}")
Dim sID : sID = oWebElement.ID
Dim oInternetExplorer : Set oInternetExplorer = oWebElement.IE()
Call oInternetExplorer.Navigate("https://www.combit.net")
Set oInternetExplorer = Nothing
Set oWebElement = Nothing
Set oListWebElements = Nothing
```

Beispiel C#-Script:

```
' Dieses Beispiel basiert auf der WebElemente-Ansicht einer combit_Large-Solution

//<!--#pragma keepeditmode-->
ListWebElements listWebElements =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).WebElements;
WebElement webElement = listWebElements.ItemByName("{64BBFFD7-EA32-4358-BBFC-
744D6A94291D}");
string id = webElement.ID;
webElement.IE().Navigate("https://www.combit.net");
webElement.Dispose();
listWebElements.Dispose();
```

3.49.2 Methoden

IE

Beschreibung:

Liefert das Interface des Internet Explorers als Objekt vom Typ **InternetExplorer** zurück.

Siehe http://msdn.microsoft.com/en-us/library/aa752084%28v=vs.85%29.aspx

Rückgabewert:

InternetExplorer

Beispiel VBScript:

```
' Dieses Beispiel basiert auf der WebElemente-Ansicht einer combit_Large-Solution

<!--#pragma keepeditmode-->
Dim oListWebElements : Set oListWebElements =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).WebElements
```

```
Dim oWebElement : Set oWebElement = oListWebElements.ItemByName("{64BBFFD7-EA32-
4358-BBFC-744D6A94291D}")
Dim oInternetExplorer : Set oInternetExplorer = oWebElement.IE()
Call oInternetExplorer.Navigate("https://www.combit.net")
Set oInternetExplorer = Nothing
Set oWebElement = Nothing
Set oListWebElements = Nothing
```

Beispiel C#-Script:

```
// Dieses Beispiel basiert auf der WebElemente-Ansicht einer combit_Large-Solution

//<!--#pragma keepeditmode-->
ListWebElements listWebElements =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2).WebElements;
WebElement webElement = listWebElements.ItemByName("{64BBFFD7-EA32-4358-BBFC-
744D6A94291D}");
webElement.IE().Navigate("https://www.combit.net");
webElement.Dispose();
listWebElements.Dispose();
```

Hinweis: combit kann nicht gewährleisten, dass alle Eigenschaften und Methoden des Internet Explorers im Kontext der Verwendung als Control innerhalb der Eingabemaske (hier als Web-Element) wirklich unterstützt werden. Es wird lediglich das Interface nach außen zur Verfügung gestellt.

3.50 WScript Objekt

Das **WScript**-Objekt beinhaltet Eigenschaften und Methoden der Scripting-Engine sowie von der Datenbank oder dem aktuellen Status unabhängige Funktionalität.

Das Objekt kann nicht von außen instanziert werden und steht daher in dieser Form nur in Scripten zur Verfügung, die innerhalb des Programms aufgerufen werden.

Bei der Verwendung im C# Script verwenden Sie als Typ für die Eigenschaften und Methoden ‚dynamic'.

3.50.1 Eigenschaften

ClipboardText

Beschreibung:

Setzt oder liest den Textinhalt der Zwischenablage.

Typ:

String (VBScript) / dynamic (C#)

Beispiel VBScript:

```
' Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

Dim oCompanyInfo : Set oCompanyInfo = cRM.CurrentProject.CompanyInfo
Dim dicCompanyInfo : Set dicCompanyInfo = CreateObject("Scripting.Dictionary")

Call dicCompanyInfo.Add("AccountNo", oCompanyInfo.AccountNo)
Call dicCompanyInfo.Add("Bank", oCompanyInfo.Bank)
Call dicCompanyInfo.Add("BankCode", oCompanyInfo.BankCode)
Call dicCompanyInfo.Add("City", oCompanyInfo.City)
Call dicCompanyInfo.Add("Company", oCompanyInfo.Company)
Call dicCompanyInfo.Add("Company2", oCompanyInfo.Company2)
Call dicCompanyInfo.Add("Company3", oCompanyInfo.Company3)
Call dicCompanyInfo.Add("Country", oCompanyInfo.Country)
Call dicCompanyInfo.Add("Email", oCompanyInfo.Email)
Call dicCompanyInfo.Add("Extra1", oCompanyInfo.Extra1)
Call dicCompanyInfo.Add("Extra2", oCompanyInfo.Extra2)
Call dicCompanyInfo.Add("Fax", oCompanyInfo.Fax)
```

```
Call dicCompanyInfo.Add("IBAN", oCompanyInfo.IBAN)
Call dicCompanyInfo.Add("Internet", oCompanyInfo.Internet)
Call dicCompanyInfo.Add("Logo", oCompanyInfo.Logo)
Call dicCompanyInfo.Add("Phone", oCompanyInfo.Phone)
Call dicCompanyInfo.Add("Street", oCompanyInfo.Street)
Call dicCompanyInfo.Add("VatID", oCompanyInfo.VatID)
Call dicCompanyInfo.Add("ZIP", oCompanyInfo.ZIP)

Dim oListCompanyInfoUserDefined : Set oListCompanyInfoUserDefined =
oCompanyInfo.UserDefinedFields
Dim oCompanyInfoUserDefinedItem
Dim nCount : nCount = 0

For nCount = 1 To oListCompanyInfoUserDefined.Count

    Set oCompanyInfoUserDefinedItem = oListCompanyInfoUserDefined.Item(nCount)

    If (oCompanyInfoUserDefinedItem.Type = 2) Then ' Wenn es eine Textinformation
ist
        Call dicCompanyInfo.Add(oCompanyInfoUserDefinedItem.Key,
oCompanyInfoUserDefinedItem.Value)
    End If

    Set oCompanyInfoUserDefinedItem = Nothing

Next

Set oListCompanyInfoUserDefined = Nothing

Dim sFullCompanyInfo : sFullCompanyInfo = ""
Dim dicItem

For Each dicItem In dicCompanyInfo
    If (Len(dicCompanyInfo(dicItem)) > 0) Then
        sFullCompanyInfo = sFullCompanyInfo & dicItem & ": " &
dicCompanyInfo(dicItem) & vbCrlf
    End If
Next

WScript.ClipboardText = sFullCompanyInfo

Call cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die
Zwischenablage übernommen.", "CompanyInfo", vbOkOnly)

Set dicCompanyInfo = Nothing
Set oCompanyInfo = Nothing
```

Beispiel C#-Script:

```
// Sammelt alle Firmenstammdaten und schreibt diese in die Windows-Zwischenablage

CompanyInfo companyInfo = cRM.CurrentProject.CompanyInfo;
System.Collections.Generic.Dictionary<string, string> dictionaryCompanyInfo = new
System.Collections.Generic.Dictionary<string, string>();

dictionaryCompanyInfo.Add("AccountNo", companyInfo.AccountNo);
dictionaryCompanyInfo.Add("Bank", companyInfo.Bank);
dictionaryCompanyInfo.Add("BankCode", companyInfo.BankCode);
dictionaryCompanyInfo.Add("City", companyInfo.City);
dictionaryCompanyInfo.Add("Company", companyInfo.Company);
dictionaryCompanyInfo.Add("Company2", companyInfo.Company2);
dictionaryCompanyInfo.Add("Company3", companyInfo.Company3);
dictionaryCompanyInfo.Add("Country", companyInfo.Country);
dictionaryCompanyInfo.Add("Email", companyInfo.Email);
dictionaryCompanyInfo.Add("Extra1", companyInfo.Extra1);
dictionaryCompanyInfo.Add("Extra2", companyInfo.Extra2);
dictionaryCompanyInfo.Add("Fax", companyInfo.Fax);
dictionaryCompanyInfo.Add("IBAN", companyInfo.IBAN);
dictionaryCompanyInfo.Add("Internet", companyInfo.Internet);
dictionaryCompanyInfo.Add("Logo", companyInfo.Logo);
dictionaryCompanyInfo.Add("Phone", companyInfo.Phone);
```

```
dictionaryCompanyInfo.Add("Street", companyInfo.Street);
dictionaryCompanyInfo.Add("VatID", companyInfo.VatID);
dictionaryCompanyInfo.Add("ZIP", companyInfo.ZIP);

ListCompanyInfoUserDefined listCompanyInfoUserDefined =
companyInfo.UserDefinedFields;

foreach (CompanyInfoUserDefinedItem companyInfoUserDefinedItem in
listCompanyInfoUserDefined)
{
    if (companyInfoUserDefinedItem.Type == 2)
    {
        dictionaryCompanyInfo.Add(companyInfoUserDefinedItem.Key,
companyInfoUserDefinedItem.Value);
    }
}

string fullCompanyInfo = string.Empty;

foreach (var dictionaryItem in dictionaryCompanyInfo)
{
    if (dictionaryItem.Value != "")
    {
        fullCompanyInfo = fullCompanyInfo + dictionaryItem.Key + ": " +
dictionaryItem.Value + System.Environment.NewLine;
    }
}

WScript.ClipboardText = fullCompanyInfo;
cRM.DialogMessageBox("Die kompletten Firmenstammdaten wurden in die Zwischenablage
übernommen.", "CompanyInfo", 0);

companyInfo.Dispose();
```

Event

Beschreibung:

Das WScript.Event-Objekt und entsprechende Unterobjekte sind global verfügbar. Die im jeweils aktuellen Script-Kontext nicht verfügbaren Unterobjekte sind Nothing bzw. null. WScript.Event.Project ist immer verfügbar und entspricht cRM.CurrentProject.

Etwaige gesetzte WScript.Event-Unterobjekte werden nicht in per Project.ExecuteScriptBy*-Methoden gestartete Scripte weitergereicht.

Der Aufruf in C# Script unterscheidet sich vom Aufruf innerhalb von VBScript:

WScript.Event (VBScript)
Event (C# Script)

Ausführliche Informationen dazu erhalten Sie im Kapitel **Event Objekt**.

Typ:

Event

Beispiel VBScript:

```
Dim oEvent, oTimeManager, oAppointment
Set oEvent = WScript.Event
Set oTimeManager = cRM.CurrentProject.timemanager
Set oAppointment = oTimeManager.Appointments.ItemByUniqueID(CStr(oEvent.Data))
MsgBox oAppointment.Start & ": " & oAppointment.Subject
```

Beispiel C#-Script:

```
object appointmentSaved = Event;
TimeManager timeManager = cRM.CurrentProject.TimeManager;
Appointment appointment =
timeManager.Appointments.ItemByUniqueID(appointmentSaved.Data);
```

```
MessageBox.Show(appointment.Start.ToString() + ": " + appointment.Subject,
cRM.AppTitle, MessageBoxButtons.OK);
```

Priority

Beschreibung:

Legt die Priorität des Scripts fest oder gibt diese zurück. Je höher die Priorität, desto häufiger bekommt das Script Rechenzeit zugeteilt und umgekehrt. Die möglichen Werte liegen zwischen –2 und +2, wobei –2 die niedrigste und +2 die höchste Priorität darstellt.

Hinweis: Kann nur in asynchronen Scripts verwendet werden.

Typ:

Long (VBScript) / dynamic (C#)

RightClicked, read-only

Beschreibung:

Gibt True zurück, wenn das Script in der Eingabemaske über einen Button ausgeführt wurde und dieser mit der rechten Maustaste geklickt wurde. Ermöglicht die Ausführung zweier verschiedener Funktionen über einen Button.

Typ:

Bool (VBScript) / dynamic (C#)

Terminate, read-only

Beschreibung:

Gibt True zurück, wenn die Anwendung gerade beendet wird bzw. das aktive Script zum Beenden aufgefordert wird. Die Eigenschaft kann und sollte unbedingt in Verbindung mit asynchronen Scripts und bei der Ausführung von Scripts im Workflow Server verwendet werden und ermöglicht ein sauberes Verlassen des Scripts.

Bitte beachten Sie dabei, dass die noch laufenden, asynchronen Scripte insgesamt 4 Sekunden Zeit haben, sich zu beenden, bevor die Beendigung erzwungen wird. Sleeps und andere Aktionen im Script sollten also keinesfalls länger als 3 Sekunden dauern, bevor wieder die .Terminate Eigenschaft geprüft und das Script dann ggf. sich so schnell wie möglich kontrolliert beenden muss.

Typ:

Bool (VBScript) / dynamic (C#)

3.50.2 Methoden

ActivateTitle

Beschreibung:

Aktiviert ein Windows-Fenster über dessen Titel.

Parameter:

Parametername	Typ	Beschreibung
WindowTitle	String	Der Titel (Name) des zu aktivierenden Fensters. Wenn Sie als erstes Zeichen ein "$" übergeben, muss der übergebene Titel lediglich enthalten sein, ansonsten genau übereinstimmen.

ActivateTitleByClassname

Beschreibung:

Aktiviert ein Windows-Fenster über dessen Klassennamen.

Parameter:

Parametername	Typ	Beschreibung
Classname	String	Der Klassenname des zu aktivierenden Fensters.

Beep

Beschreibung:

Gibt den Standard Windows Warnton aus.

CheckAbortedWaitDlg

Beschreibung:

Die Methode muss zwischen StartWaitDlg und EndWaitDlg aufgerufen werden und gibt zurück, ob der Benutzer die Schaltfläche "Abbrechen" des Wartedialoges betätigt hat, sofern StartWaitDlg mit Abbruch-Button angezeigt wurde. Die Methode wird vorzugsweise in Ausführungsschleifen verwendet.

Rückgabewert:

Bool (VBScript) / dynamic (C#)

EndWaitDlg

Beschreibung:

Blendet einen zuvor mit StartWaitDlg angezeigten Wartedialog aus.

GetGlobalProperty

Beschreibung:

Liest eine globale Script-Einstellung aus, die zuvor mit SetGlobalProperty gesetzt wurde oder gibt den Standardwert zurück.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der Einstellung.
Default	String	Der Standardwert der Einstellung, der zurückgegeben wird, wenn keine Einstellung mit dem angegebenen Namen existiert.

Rückgabewert:

String (VBScript) / dynamic (C#)

GetUserProperty

Beschreibung:

Liest eine benutzerspezifische Script-Einstellung aus, die zuvor mit **SetUserProperty** gesetzt wurde oder gibt den Standardwert zurück.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der Einstellung.
Default	String	Der Standardwert der Einstellung, der zurückgegeben wird, wenn keine Einstellung mit dem angegebenen Namen existiert.

Rückgabewert:

String (VBScript) / dynamic (C#)

SetGlobalProperty

Beschreibung:

Setzt eine globale Script-Einstellung. Die Speicherung erfolgt in der Datei "global.ini" in der "cmbt_Files" Tabelle.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der Einstellung.
String	String	Der (neue) Wert der Einstellung.

Rückgabewert:

Bool (VBScript) / dynamic (C#)

SetUserProperty

Beschreibung:

Setzt eine benutzerspezifische Script-Einstellung. Die Speicherung erfolgt in der Datei "<BENUTZER>\user_scriptvars.ini" in der "cmbt_Files" Tabelle.

Parameter:

Parametername	Typ	Beschreibung
Name	String	Der Name der Einstellung.
String	String	Der (neue) Wert der Einstellung.

Rückgabewert:

Bool (VBScript) / dynamic (C#)

SetWaitDlgText

Beschreibung:

Hiermit kann der Text in einem per StartWaitDlg gestarteten Wartedialog nachträglich geändert werden.

Parameter:

Parametername	Typ	Beschreibung
Text	String	Der anzuzeigende Informationstext.

Sleep

Beschreibung:

Hält die Ausführung des Scripts für eine anzugebende Zeit an.

Parameter:

Parametername	Typ	Beschreibung
nTime	Long	Anzahl Millisekunden, die das Script angehalten werden soll. `Sleep 1000` hält die Ausführung bspw. für 1 Sekunde an.

StartWaitDlg

Beschreibung:

Zeigt einen Wartedialog mit einer Fortschrittsanimation sowie einem zu übergebenden Informationstext an. Die Ausführung des Scripts wird während der Anzeige fortgeführt. Die Ausblendung erfolgt mit **EndWaitDlg**. Über den zweiten Parameter kann bestimmt werden, ob der Dialog eine "Abbrechen"-Schaltfläche enthalten soll. Wenn ja, dann kann mit **CheckAbortedWaitDlg** geprüft werden, ob die Schaltfläche betätigt wurde. Reserviert: Über den dritten Parameter kann die Verzögerung bis zur Anzeige des Dialogs eingestellt werden. Standardmäßig wird eine Verzögerung von 3 Sekunden verwendet, um zu verhindern, dass schnell geöffnete und geschlossene Dialoge flackern.

Parameter:

Parametername	Typ	Beschreibung
sText	String	Der anzuzeigende Informationstext.
bCancel	Bool	Anzeige einer Schaltfläche "Abbrechen"
nInitialDelay	Long	Reserviert, derzeit immer 0. (Optional. Verzögerung bis zur Anzeige des Dialogs in Sekunden. 0 = unmittelbare Anzeige. Der vorbelegte Wert beträgt 3 Sekunden.)

WaitTitle

Beschreibung:

Methode wartet so lange, bis das Fenster mit dem übergebenen Titel gefunden wurde. Anschließend wird das gefundene Fenster aktiviert.

Parameter:

Parametername	Typ	Beschreibung
sTitel	String	Der Titel (Name) des zu aktivierenden Fensters. Wenn Sie als erstes Zeichen ein "$" übergeben, muss der übergebene Titel lediglich enthalten sein, ansonsten genau übereinstimmen.
nTime	Long	Max. Wartezeit in Sekunden. -1 bedeutet unbegrenzte Wartezeit.

Rückgabewert:

Bool (VBScript) / dynamic (C#)

Wert	Beschreibung
True	Fenster wurde gefunden.
False	Kein Fenster gefunden.

4 Ereignisse

Die Ereignis Verwaltung ermöglicht es auf verschiedene Ereignisse mit automatischen Aktionen zu reagieren. Hierzu können für jedes der angebotenen Ereignisse Scripte oder Workflows hinterlegt werden. Scripte und Workflows können per Datei verknüpft werden, alternativ können Scripte auch direkt im Projekt eingebettet werden.

Einschränkungen für diese Ereignis-Scripte:

Das Recht "Darf Scripte ausführen" wird bei diesen Scripten nicht abgefragt (ist also nicht wirksam), damit eine über Ereignis-Scripte konfigurierte Anwendungslogik in jedem Fall durchlaufen wird.

Es erfolgt keine Statusanzeige (Animation) in der Statusleiste.

Es gibt projektspezifische und ansichtsspezifische Ereignisse sowie Ereignisse für Termine und Aufgaben. Die nachfolgenden Ereignisse stehen Ihnen im Projekt unter Konfigurieren >Projekt > Ereignisse zur Verfügung.

Wichtig: Bei aufwändigen Scripten (lange Laufzeit) empfiehlt es sich, asynchrone Scripte für die Ereignisse zu verwenden, da ansonsten die Anwendung zu lange blockiert wird.

Ausgenommen hiervon sind jedoch die beiden Ereignisse "Projekt wird geschlossen" und "Ansicht wird geschlossen", hier muss ein synchrones Script verwendet werden. Generell müssen Sie bei asynchronen Scripten unbedingt berücksichtigen, dass der Anwender in der Zwischenzeit weiterarbeiten (und zum Beispiel den aktuellen Datensatz wechseln) kann. Dies kann je nach Script unvorhersehbare Effekte haben!

4.1 Event Objekt

Bei den ausgeführten Scripten steht das Objekt Event zur Verfügung, welches Informationen über das aktuelle Ereignis zur Verfügung stellt. Auf dieses Objekt wird über das WScript Objekt zugegriffen:

```
WScript.Event
```

Hinweis: Bei der Verwendung von C# Scripten verwenden Sie lediglich Event, nicht WScript.Event. Ein Aufruf könnte hierbei wie folgt aussehen:

```
dynamic Record = Event.Record1;
```

Bei der Verwendung im C# Script verwenden Sie als Typ für die Eigenschaften immer ‚dynamic'.

Das WScript.Event-Objekt und entsprechende Unterobjekte sind global verfügbar. Die im jeweils aktuellen Script-Kontext nicht verfügbaren Unterobjekte sind Nothing bzw. null. WScript.Event.Project ist immer verfügbar und entspricht cRM.CurrentProject.

Etwaige gesetzte WScript.Event-Unterobjekte werden nicht in per Project.ExecuteScriptBy*-Methoden gestartete Scripte weitergereicht.

4.1.1 Eigenschaften

Cancel

Beschreibung:

Sofern der das Ereignis auslösende Vorgang einen Abbruch unterstützt, kann dieser durch Setzen dieser Eigenschaft auf True vorgenommen werden. Eine genaue Beschreibung finden Sie in diesem Fall bei den jeweiligen Ereignissen.

Typ:

Bool (VBScript) / dynamic (C#)

ChangedFields

Beschreibung:

Das Objekt ist für das Ereignis "Feldänderung" gültig.

Bei den Ereignissen "Feldänderung" und "Zeitüberschreitung" ist Record1 der betroffene Datensatz.

Typ:

ListEventFieldChange (VBScript) / dynamic (C#)

Data

Beschreibung:

Enthält ggf. eine Zusatzinformation zum Ereignis.

Typ:

String (VBScript) / dynamic (C#)

Project

Beschreibung:

Enthält das Project Objekt um das es im Ereignis geht.

Typ:

Project (VBScript) / dynamic (C#)

Record1

Beschreibung:

Dieses Objekt zeigt ggf. auf den am Ereignis beteiligten Datensatz. Eine genaue Beschreibung finden Sie in diesem Fall bei den jeweiligen Ereignissen.

Typ:

Record (VBScript) / dynamic (C#)

Record2

Beschreibung:

Dieses Objekt zeigt ggf. auf den am Ereignis beteiligten Datensatz. Eine genaue Beschreibung finden Sie in diesem Fall bei den jeweiligen Ereignissen.

Typ:

Record (VBScript) / dynamic (C#)

View

Beschreibung:

Enthält das View Objekt, um das es im Ereignis geht (sofern beim Ereignis verfügbar).

Typ:

View (VBScript) / dynamic (C#)

4.2 Projektspezifische Ereignisse

4.2.1 Projekt wurde geöffnet

Das Ereignis wird ausgelöst, wenn das Projekt geöffnet wurde.

4.2.2 Projekt wird geschlossen

Das Ereignis wird ausgelöst, bevor die Anwendung sich die aktuell offenen Ansichten merkt (um sie beim nächsten Start wiederherstellen zu können).

4.2.3 Eingehender Anruf

Das Ereignis wird ausgelöst, wenn ein Anruf eingeht.

- Data: enthält eingehende Nummer (ohne Sonderzeichen).

Das Benachrichtigungsfenster über einen eingehenden Anruf kann mittels WScript.Event.Cancel = true (C#: Event.RecordSet = True;) geschlossen werden.

4.2.4 Eingehender Anruf wurde gesucht

Wird erst nach der durchgeführten Rufnummernsuche (phone manager) ausgelöst. Falls keine Datensätze gefunden wurden, hat WScript.Event.RecordSet (Event.RecordSet bei C# Script) den Wert Nothing. Falls ein derartiges Ereignis eingerichtet wurde, wird seitens der Anwendung keine "Datensatz wurde nicht gefunden" Meldung mehr ausgegeben, um Automatisierungen zu ermöglichen.

Achtung: Das RecordSet ist unter Umständen nicht visuell.

- Data: enthält eingehende Nummer (ohne Sonderzeichen).

4.2.5 Menübefehl wird ausgeführt

Das Ereignis wird ausgeführt, wenn ein Menübefehl ausgeführt wird.

- Data: <MenuID> (siehe Kapitel Menü-IDs für die entsprechenden Menü-IDs, mehrere Menü-IDs können semikolonsepariert angegeben werden)

4.2.6 Menübefehl verstecken

Das Ereignis versteckt einen Menübefehl.

- Data: <MenuID> (siehe Kapitel Menü-IDs für die entsprechenden Menü-IDs, mehrere Menü-IDs können semikolonsepariert angegeben werden)

Beispiel VBScript:

```
' Verstecke 'Start > Suche' und 'Fenster > Wechseln zu > Info-Zentrale'
WScript.Event.Data = "32810;33131"
```

4.3 Ereignisse für Termine und Aufgaben

4.3.1 Gemeinsame Ereignisse

Diese Ereignisse sollten nur einmal entweder für Termine oder Aufgaben definiert werden.

4.3.1.1 Menübefehl wird ausgeführt

Das Ereignis wird ausgeführt, wenn ein Menübefehl ausgeführt wird.

- Data: <MenuID> (siehe Kapitel Menü-IDs für die entsprechenden Menü-IDs, mehrere Menü-IDs können semikolonsepariert angegeben werden)

4.3.1.2 Menübefehl verstecken

Das Ereignis versteckt einen Menübefehl.

- Data: <MenuID> (siehe Kapitel Menü-IDs für die entsprechenden Menü-IDs, mehrere Menü-IDs können semikolonsepariert angegeben werden)

4.3.2 Termine

4.3.2.1 Termin wurde gespeichert

► Data: enthält die ID des Termins. Zugriff auf den Termin erhält man mit der Methode ItemByUniqueID des time manager Appointment Objekts.

Beispiel VBScript:

```
Dim oEvent, oTimeManager, oAppointment
Set oEvent = WScript.Event
Set oTimeManager = cRM.CurrentProject.timemanager
Set oAppointment = oTimeManager.Appointments.ItemByUniqueID(CStr(oEvent.Data))
MsgBox oAppointment.Start & ": " & oAppointment.Subject
```

Beispiel C#-Script:

```
object appointmentSaved = Event;
TimeManager timeManager = cRM.CurrentProject.TimeManager;
Appointment appointment =
timeManager.Appointments.ItemByUniqueID(appointmentSaved.Data);
MessageBox.Show(appointment.Start.ToString() + ": " + appointment.Subject,
cRM.AppTitle, MessageBoxButtons.OK);
```

4.3.3 Aufgaben

4.3.3.1 Aufgabe wurde gespeichert / Aufgabe wurde erledigt

► Data: enthält die ID der Aufgabe. Zugriff auf die Aufgabe erhält man mit der Methode ItemByUniqueID des time manager ToDo Objekts.

Beispiel VBScript:

```
Dim oEvent, oTimeManager, oToDo
Set oEvent = WScript.Event
Set oTimeManager = cRM.CurrentProject.timemanager
Set oToDo = oTimeManager.ToDos.ItemByUniqueID(CStr(oEvent.Data))
MsgBox oToDo.Start & ": " & oToDo.Subject
```

Beispiel C#-Script:

```
object toDoSaved = Event;
TimeManager timeManager = cRM.CurrentProject.TimeManager;
ToDo toDo = timeManager.ToDos.ItemByUniqueID(toDoSaved.Data);
MessageBox.Show(toDo.Start.ToString() + ": " + toDo.Subject, cRM.AppTitle,
MessageBoxButtons.OK);
```

4.4 Ansichtenspezifische Ereignisse

Wichtig: Für ansichtenspezifische Ereignisse verwenden Sie WScript.Event.View (Event.View unter C#), um auf die passende Ansicht zugreifen zu können, da ein ActiveView Objekt nicht zwangsläufig die gewünschte Ansicht zurückliefert, insbesondere, wenn beim Programmstart gleichzeitig weitere Ansichten geöffnet, beziehungsweise wiederhergestellt werden, die Ansichten durch eine Benutzereingabe schnell gewechselt werden oder die Info-Zentrale und/oder die Termin- und Aufgabenverwaltung zum gleichen Zeitpunkt geöffnet werden.

4.4.1 Ansicht wurde geöffnet

Die Ansicht, in der das Ereignis konfiguriert wurde, wurde geöffnet.

4.4.2 Ansicht wird geschlossen

Die Ansicht, in der das Ereignis konfiguriert wurde, wird geschlossen.

4.4.3 Datensatz wird aus Papierkorb wiederhergestellt

Der Datensatz wird aus dem Papierkorb wiederhergestellt.

- Data: Enthält mittels "|" getrennt den Ansichtennamen und die Datensatz-ID des betroffenen Datensatzes aus der Ansicht.

Hinweis: Das Ereignis wird vor der eigentlichen Aktion ausgelöst. Somit kann das Wiederherstellen/Endgültig-Löschen per WScript.Event.Cancel = true verhindert werden.

4.4.4 Datensatz wird gespeichert

Der Datensatz wird gespeichert (betrifft das interaktive Speichern eines Datensatzes in der Eingabemaske).

- Data: Wenn "0" dann ist es ein bestehender Datensatz, wenn "1" dann ist es ein neuer Datensatz.

Hinweis: WScript.Event.Record1 (Event.Record1 im C# Script) enthält den Datensatz kurz bevor er gespeichert wird. Lese- und Schreibzugriffe auf die Daten (GetContents…, SetContents…) dürfen im Kontext dieses Events nicht über die Eingabemaske (InputForm) stattfinden, sondern ausschließlich über WScript.Event.Record1 (Event.Record1 im C# Script). Lock(), Unlock(), Save() darf für den Record nicht aufgerufen werden. Die Ansicht darf im Kontext dieses Events nicht per Script geschlossen werden (View.Close).

Das Speichern eines Datensatzes kann durch Setzen der Eigenschaft Cancel des Event Objekts auf "True" unterbunden werden.

Wichtig: Ein Schreibzugriff auf CurrentRecord ist in diesem Event nicht möglich! Der Event wird nicht beim direkten Ändern in Übersichtsliste/Container ausgelöst.

4.4.5 Datensatz wird in Papierkorb endgültig gelöscht

Der Datensatz wird endgültig aus dem Papierkorb der betreffenden Ansicht gelöscht.

- Data: Enthält mittels "|" getrennt den Ansichtennamen und die Datensatz-ID des betroffenen Datensatzes aus der Ansicht.

Hinweis: Das Ereignis wird vor der eigentlichen Aktion ausgelöst. Somit kann das Wiederherstellen/Endgültig-Löschen per WScript.Event.Cancel = true verhindert werden.

4.4.6 Datensatz wurde gespeichert

Der Datensatz wurde gespeichert (betrifft das interaktive Speichern eines Datensatzes in der Eingabemaske).

- Data: Wenn "0" dann ist es ein bestehender Datensatz, wenn "1" dann ist es ein neuer Datensatz.

Hinweis: Das Event wird nicht beim direkten Ändern in Übersichtsliste/Container ausgelöst. Zudem darf die Ansicht im Kontext dieses Events nicht per Script geschlossen werden (View.Close).

4.4.7 Datensatz wird zusammengeführt

Der Datensatz wird zusammengeführt.

- Data: Record1 stellt den Primär-, Record2 den Sekundär-Datensatz dar.

Hinweis: Das Speichern eines Datensatzes kann durch Setzen der Eigenschaft Cancel des Event Objekts auf "True" unterbunden werden.

4.4.8 Datensatz wurde zusammengeführt

Der Datensatz wurde zusammengeführt.

- Data: Record1 stellt den Ergebnis-Datensatz dar.

4.4.9 Datensatzbearbeitung wird begonnen

Innerhalb der Eingabemaske wird mit der Bearbeitung eines bestehenden Datensatzes begonnen.

- Data: Record1 stellt den Datensatz dar.

Hinweis: Das Speichern eines Datensatzes kann durch Setzen der Eigenschaft Cancel des Event Objekts auf "True" unterbunden werden.

Das analoge Ereignis für die Bearbeitung eines neuen Datensatzes ist Neuer Datensatz wird bearbeitet.

4.4.10 Datensatzbearbeitung wurde begonnen

Innerhalb der Eingabemaske wurde mit der Bearbeitung eines Datensatzes begonnen.

- Data: 0 – Bearbeitung eines bestehenden Datensatzes, 1 – Bearbeitung eines neuen Datensatzes
- Record1 stellt den Datensatz dar.

Hinweis: Das Ereignis wird nicht ausgelöst, wenn ein Script die Datensatzbearbeitung ausgelöst (z. B. via CurrentInputForm(0)), diese aber auch direkt gleich wieder beendet (z. B. via CurrentInputForm.Save) hat.

4.4.11 Feldänderung

Ein Feld wurde geändert. Siehe die Eigenschaft ChangedFields im Event Objekt.

- Data: Record1 stellt den betroffenen Datensatz dar.

4.4.12 Zeitüberschreitung

Eine Zeitüberschreitung ist eingetreten.

- Data: Record1 stellt den betroffenen Datensatz dar.

4.4.13 Neuer Datensatz wird bearbeitet

Wird ausgelöst, sobald in der Eingabemaske ein neuer Datensatz anlegt wird. Kann z. B. zum Vorbelegen von Feldern bei neuen Datensätzen per Script verwendet werden.

Beispiel VBScript:

```
Option Explicit
Dim oProject, oActiveView, oCurrentInputForm
' Aktives cRM Projekt
Set oProject = cRM.CurrentProject
```

```
Set oActiveView = oProject.ActiveViews.ActiveView
Set oCurrentInputForm = oActiveView.CurrentInputForm(0)
' Feld "ABC" in der Ansicht 'Firmen' auf "A" setzen
oCurrentInputForm.SetContentsByName CStr("ABC"), CStr("A")
```

Beispiel C#-Script:

```
// Aktives Projekt
Project currentProject = cRM.CurrentProject;
combit.cRM.COM.View activeView = currentProject.ActiveViews.ActiveView;
InputForm currentInputForm = activeView.CurrentInputForm(0);

// Feld "ABC" in der Ansicht "Firmen" auf "A" setzen
currentInputForm.SetContentsByName("ABC", "A");
```

4.4.14 Menübefehl wird ausgeführt

Das Ereignis wird ausgeführt, wenn ein Menübefehl ausgeführt wird.

> Hinweis: Das Ereignis erfordert das etwaige Beenden des Bearbeiten-Modus (die Rückfrage nach dem Speichern etwaiger Änderungen). Wählt der Benutzer bei der Rückfrage "Abbrechen", so wird das Ereignis nicht ausgeführt. Scripte für dieses Event unterstützen jedoch <!--#pragma keepeditmode--> und <!--#pragma autosave-->.

Menü:

- Data: <MenuID> (siehe Kapitel Menü-IDs für die entsprechenden Menü-IDs, mehrere Menü-IDs können semikolonsepariert angegeben werden)

Relationen-Container:

- Data: <MenuID>|<RelationFieldName#RelationContainerID>|<FieldName>|<FieldContent> (siehe Kapitel Menü-IDs für die entsprechenden Menü-IDs, mehrere Menü-IDs können semikolonsepariert angegeben werden)

> Hinweis: <FieldName> und <FieldContent> werden nur dann relevant und gefüllt, wenn der Container-(Kontext-)Menüpunkt im Zusammenhang mit dem konkret angeklickten Feld steht. Also bspw., wenn ein Anwender per Rechtsklick auf ein im Container angezeigtes Dokument klickt und "Dokument mit verknüpfter Anwendung öffnen" ausführt, oder bei einem Rechtsklick auf eine Telefonnummer und dort "Anrufen" ausgeführt wird.
>
> Führt der Anwender einen Doppelklick auf einen Container-Datensatz aus oder öffnet diesen mit ENTER, dann werden folgende Data-Elemente befüllt: <MenuID> und <RelationFieldName#RelationContainerID>. <FieldName> und <FieldContent> bleiben leer.

Termin-Container:

- Data: <MenuID>|:AppointmentsContainer (siehe Kapitel Menü-IDs für die entsprechenden Menü-IDs, mehrere Menü-IDs können semikolonsepariert angegeben werden)

Aufgaben-Container:

- Data: <MenuID>|:TodosContainer (siehe Kapitel Menü-IDs für die entsprechenden Menü-IDs, mehrere Menü-IDs können semikolonsepariert angegeben werden)

Funktionsdefinitionen:

- Data: <MenuID>|<RelationFieldName#RelationContainerID>

> Hinweis: Hierbei handelt es sich um die Funktionsdefinitionen, welche als Schaltflächen im Eingabemaskendesigner platziert werden können - bspw. "Container.Löschen" oder "Funktionsdefinition Übernahmemaske". Dabei werden diese Funktionsdefinitionen auf entsprechend passende Menübefehle interpretiert - bspw. wird für die Funktionsdefinition "Container.Löschen" der Menüpunkt "Kontextmenü - Container-Datensatz: Löschen" (ID: 32847) verwendet. Siehe Kapitel Menü-IDs für die entsprechenden Menü-IDs.

Beispiel zum Verhindern des Löschen-Menübefehls (Menü-ID 32774)

VBScript:

```
MsgBox "Löschen ist nicht erlaubt!"
WScript.Event.Cancel = True ' Verhindern des Befehls
```

C#-Script:

```
MessageBox.Show("Löschen ist nicht erlaubt!", cRM.AppTitle, MessageBoxButtons.OK);
Event.Cancel = true; // Verhindern des Befehls
```

Beispiel zum Verhindern einer Aktion eines Menüpunkts aus dem Container-Kontextmenü

VBScript:

```
Option Explicit
Dim dataItems
dataItems = Split(WScript.Event.Data, "|", -1, 1)
Dim menuID
menuID = dataItems(0)
Dim relationContainerID
relationContainerID = dataItems(1)
If (relationContainerID = "ID.Aktivitäten.CompanyID#{45F5CAD2-73A1-4D48-B21E-
F38D27093D12}") Then
    MsgBox "Sie dürfen hier im Aktivitäten-Container keinen neuen Datensatz
anlegen. Daher wird die Aktion nun abgebrochen.", vbOKOnly + vbInformation,
"Hinweis/Info"
    WScript.Event.Cancel = True
End If
```

C#-Script:

```
string[] dataItems = Event.Data.Split('|');
if (dataItems.Length > 1)
{
    string menuID = dataItems[0];
    string relationContainerID = dataItems[1];

    if (relationContainerID == "ID.Aktivitäten.CompanyID#{45F5CAD2-73A1-4D48-B21E-
F38D27093D12}")
    {
        MessageBox.Show("Sie dürfen im Aktivitäten-Container keinen neuen
Datensatz anlegen. Daher wird die Aktion nun abgebrochen.", cRM.AppTitle + " -
Hinweis/Info" , MessageBoxButtons.OK, MessageBoxIcon.Information);
        Event.Cancel = true;
    }
}
```

4.4.15 Menübefehl verstecken

Das Ereignis versteckt einen Menübefehl.

Menü:

- Data: <MenuID> (siehe Kapitel **Menü-IDs** für die entsprechenden Menü-IDs, mehrere Menü-IDs können semikolonsepariert angegeben werden)

Beispiel zum Verstecken des Suchen-Menübefehls (Menü-ID 32810)

VBScript:

```
WScript.Event.Data = "32810"
```

C#-Script:

```
Event.Data = "32810";
```

4.5 EventFieldChange Objekt

Einzelne Feldänderung.

4.5.1 Eigenschaften

ContentNew, read-only

Beschreibung:

Liefert den neuen Inhalt des geänderten Feldes zurück.

Typ:

Variant

ContentOld, read-only

Beschreibung:

Liefert den alten Inhalt des geänderten Feldes zurück.

Typ:

Variant

Fieldname, read-only

Beschreibung:

Liefert den Feldnamen des geänderten Feldes zurück.

Typ:

String

4.6 ListEventFieldChange Objekt

Liste von Feldänderungen.

4.6.1 Eigenschaften

Count, read-only

Beschreibung:

Liefert die Anzahl Feldänderungen zurück.

Typ:

Long

4.6.2 Methoden

Item

Beschreibung:

Liefert eine einzelne Feldänderung vom Typ EventFieldChange zurück.

Typ:

EventFieldChange

4.7 Beispiele

4.7.1 Ereignis 'Feldänderung' per Script verarbeiten

Siehe Script 'EventFieldChange.vbs' im Script-Verzeichnis der mitgelieferten Solutions.

4.7.2 Ereignis 'Zeitüberschreitung' per Script verarbeiten

Siehe Script 'EventTimeout.vbs' im Script-Verzeichnis der mitgelieferten Solutions.

5 E-Mail-Autopilot

Der E-Mail-Autopilot startet bei einer gefundenen E-Mail ein konfiguriertes Script. Damit mit Hilfe des gestarteten Scripts das cRM-Objekt verwendet werden kann, können Sie entweder das konfigurierte cRM-Objekt vom Autopilot verwenden oder selbst im Script ein eigenes cRM-Objekt erstellen.

Eine ausführliche Beschreibung der Autopilot-Konfiguration finden Sie im Abschnitt "E-Mail-Autopilot" im Kapitel "Workflows und Ereignisse" des Handbuchs.

> Wichtig: Der E-Mail-Autopilot steht erst ab der Professional-Edition zur Verfügung.
>
> Bitte beachten Sie, dass bei Übergabe von String-Parametern diese immer mit CStr() formatiert werden müssen.
>
> Beispiel: `Call Mail.Forward(CStr(<Neuer Betreff>))`
>
> Hinweis: Wenn in einem Script die Platzhalter PRJDIR und/oder APPDIR verwendet werden (beispielsweise für ein Include-Statement), muss zwingend das cRM-COM-Objekt als Script-Objekt konfiguriert werden.

5.1 Attachment Objekt

5.1.1 Eigenschaften

Name

Beschreibung:

Dateiname des Dateianhangs.

> Hinweis: Der Dateiname wird bei Simple MAPI im 8.3-Format zurückgeliefert.

Typ:

String

5.1.2 Methoden

SaveAs

Beschreibung:

Speichert die Datei im angegebenen Verzeichnis.

Parameter:

Parametername	Typ	Beschreibung
sPath	String	Speicherpfad

Rückgabewert:

Bool

5.2 Attachments Objekt

5.2.1 Eigenschaften

Count

Beschreibung:

Gibt die Anzahl an gefundenen Anhängen zurück.

Typ:

Int

5.2.2 Methoden

Item

Beschreibung:

Gibt das entsprechende Attachment-Objekt zurück. Es muss die Index-Nummer des Eintrages übergeben werden. Der Index geht von 0 bis Count-1.

Parameter:

Parametername	Typ	Beschreibung
Index	Long	Index-Nummer.

Typ:

Attachment

5.3 HostApp Objekt

Bietet Zugriff auf die Applikation und das Mail-Objekt.

5.3.1 Eigenschaften

Mail

Beschreibung:

Gibt die aktuelle E-Mail als Mail-Objekt zurück.

Typ:

Mail

StopFurtherAction

Beschreibung:

Wird diese Eigenschaft auf True gesetzt, werden keine weiteren Betreffzeilen ausgewertet. Das Programm fängt dann mit dem Bearbeiten der nächsten Mail an.

Rückgabewert:

Bool

Version

Beschreibung:

Gibt die Version zurück.

Rückgabewert:

Long

5.3.2 Methoden

AddProtocolEntry

Beschreibung:

Über diese Methode des HostApp-Objektes können eigene Ausgaben in die Protokolldatei des E-Mail-Autopiloten hinzugefügt werden. Jeder hinzugefügte Eintrag in der Protokolldatei erhält den Präfix "Custom: ", um die eigenen Einträge identifizieren zu können.

Hinweis: Hierfür muss die Protokollierung im E-Mail-Autopiloten aktiviert sein.

Parameter:

Parametername	Typ	Beschreibung
sLogEntry	String	Neuer Eintrag für die Protokolldatei.

Beispiel VBScript:

```
Call HostApp.AddProtocolEntry(CStr("<Eigener Protokolleintrag>"))
```

Beispiel C#-Script:

```
HostApp.AddProtocolEntry("<Eigener Protokolleintrag>");
```

SetOutputString

Beschreibung:

Dient zur Ausgabe von Zwischenschritten innerhalb eines Scripts. Der Text wird in der Hauptansicht des Programms angezeigt.

Hinweis: Der bisherige Ausgabetext wird gelöscht!

Parameter:

Parametername	Typ	Beschreibung
sOutputString	String	Statusanzeige im Programm

SetOutputStringAppend

Beschreibung:

Dient zur Ausgabe von Zwischenschritten innerhalb eines Scripts. Der Text wird in der Hauptansicht des Programms angezeigt. Der Statustext wird an die bestehende Ausgabe anhängt.

Parameter:

Parametername	Typ	Beschreibung
sOutputString	String	Statusanzeige im Programm.

5.4 Mail Objekt

Bietet Zugriff auf eine vom E-Mail-Autopilot erkannte E-Mail.

5.4.1 Eigenschaften

Attachments

Beschreibung:

Liefert einen ggf. vorhandenen Mailanhang als Attachments-Objekt zurück.

Um zu überprüfen, ob die E-Mail-Anhänge hat, muss das angelegte Objekt mit `Is Nothing` auf Gültigkeit überprüft werden und die Anzahl der Anhänge mit Hilfe der Count-Eigenschaft des zurückgelieferten Attachments-Objekts abgefragt werden.

Typ:

Attachments

BodyMessage

Beschreibung:

Liefert den kompletten Nachrichtentext zurück.

Rückgabewert:

String

CCRecipients

Beschreibung:

Liefert eine semikolon-separierte Liste der E-Mail-Empfänger im CC zurück.

Rückgabewert:

String

Recipients

Beschreibung:

Liefert eine semikolon-separierte Liste der E-Mail-Empfänger zurück.

Rückgabewert:

String

SendDateTime

Beschreibung:

Gibt das Sendedatum inkl. Uhrzeit der E-Mail zurück. Das Format wird gemäß den Windows Systemeinstellungen zurückgegeben, d.h. bei einer Regionseinstellung "Deutsch" erfolgt die Ausgabe umgerechnet in die lokale Zeit im Format: 16.07.2012 11:09:04

Rückgabewert:

String

Sender

Beschreibung:

Name des Absenders inklusive E-Mail-Adresse.

Rückgabewert:

String

Beispiel VBScript:

```
Set oMailObject = HostApp.Mail
sMailSubject = oMailObject.Subject
sMailSender = oMailObject.Sender
HostApp.SetOutputString("Betreff der E-Mail: " & sMailSubject)
HostApp.SetOutputStringAppend("Absender der E-Mail: " & sMailSender)
```

Beispiel C#-Script:

```
Mail mailObject = HostApp.Mail;
string mailSubject = mailObject.Subject;
string mailSender = mailObject.Sender;
HostApp.SetOutputString("Betreff der E-Mail: " + mailSubject);
HostApp.SetOutputStringAppend("Absender der E-Mail: " + mailSender);
```

SenderAddressResolved

Beschreibung:

E-Mail-Adresse des Absenders (wird nach RFC822 Standard für Internet-Mails ermittelt).

Rückgabewert:

String

Subject

Beschreibung:

Gibt den Betreff der E-Mail zurück.

Rückgabewert:

String

5.4.2 Methoden

Delete

Beschreibung:

Löscht die E-Mail physikalisch aus dem Postfach.

Forward

Beschreibung:

Leitet die gesamte E-Mail an den konfigurierten E-Mail-Empfänger weiter. Ausführliche Informationen zur Konfiguration des E-Mail-Betreffs finden Sie unter " E-Mail-Betreff" im Abschnitt "E-Mail-Autopilot" im Kapitel "Workflows und Ereignisse" des Handbuchs.

Hinweis: Wenn keine Parameter angeben sind, so werden die im Programm konfigurierten Werte verwendet.

Parameter:

Parametername	Typ	Beschreibung
sSubject	String	Neuer Betreff der zu weiterleitenden E-Mail (optional).
sRecipients	String	E-Mail-Adresse des Empfängers (optional).

Rückgabewert:

Bool

SaveAs

Beschreibung:

Speichert die gesamte E-Mail inkl. Anhang als Datei.

Parameter:

Parametername	Typ	Beschreibung
sFile	String	Pfad und Dateiname der Maildatei. Als Dateiendung muss *.eml* verwendet werden!

Rückgabewert:

Bool

SetMarkAsRead

Beschreibung:

Über diese Methode des Mail-Objektes können E-Mails bei Simple MAPI, Extended MAPI und IMAP als gelesen markiert werden. Im Konfigurationsdialog ist hierfür eine Option ("E-Mail als gelesen markieren") vorhanden, um das Standard-Verhalten steuern zu können.

Hinweis: Für SMTP/POP3 ist diese Methode irrelevant.

Beispiel VBScript:

```
Call HostApp.Mail.SetMarkAsRead()
```

Beispiel C#-Script:

```
HostApp.Mail.SetMarkAsRead();
```

5.5 WScript Objekt

Das Objekt kann nicht von außen instanziert werden und steht daher in dieser Form nur in Scripten zur Verfügung, die innerhalb des Programms aufgerufen werden.

5.5.1 Methoden

Sleep

Beschreibung:

Hält die Ausführung des Scripts für eine anzugebende Zeit an.

Parameter:

Parametername	Typ	Beschreibung
Time	Long	Anzahl Millisekunden, die das Script angehalten werden soll. `Sleep 1000` hält die Ausführung bspw. für 1 Sekunde an.

6 Info-Zentrale, Web-Ansichten, Web-Elemente, Web-Panel

Sie können für die Info-Zentrale, in Web-Ansichten (in jeder Ansicht über Ansicht > Web), in Web-Elementen in der Eingabemaske oder im Web-Panel eine Internetseite oder eine eigene HTML-Seite hinterlegen. Die HTML-Seiten können hierbei mit jedem beliebigen HTML-Editor erstellt werden.

6.1 Allgemeine Hinweise

Ausführliche Informationen zur jeweiligen Konfiguration finden Sie unter "Projekteigenschaften" im Kapitel "Konfiguration Projekt" bzw. unter "Allgemein" im Kapitel "Konfiguration Ansichten" des Handbuchs.

Die Anzeige von dynamischen Informationen (z. B. aus der Anwendung) innerhalb einer HTML-Seite geschieht über eine Scriptfunktion. Die Scriptfunktion ermittelt den dynamischen Inhalt und gibt ihn über das Document Object Model (DOM) des Webbrowsers mittels document.write bzw. getElementById(…).innerHTML Kommando aus. Dabei wird das Objektmodel des combit CRM nahezu vollumfänglich unterstützt.

> Hinweis: C#-Scripte stehen jeweils nicht zur Verfügung.
>
> Tipp: Um Informationen in der HTML-Ansicht zu "speichern" oder über mehrere HTML-Seiten hinweg zu erhalten, stehen Ihnen im Project-Objekt der Anwendung die Methoden SetUserProperty() und GetUserProperty() zur Verfügung. Hierüber können Werte persistent gespeichert und geladen werden. Dies wird z. B. für die Speicherung/Abfrage des Menü-Navigationsstatus in der Datei "script.js" verwendet.

6.2 Hinweise zu Web-Elementen

Im Eigenschaftsdialog des Web-Elements im Eingabemaskendesigner geben Sie dazu für die Eigenschaft "URL" die Adresse einer Webseite an und übergeben ggf. weitere Parameter.

Am Beispiel des Web-Elements "PDF" in der mitgelieferten Large Solution sähe dies wie folgt aus:

```
%PRJDIR%\Web-
Elemente\showPDF\showPDF.html?pdfFile=%PRJDIR%\Mailvorlagen\Teambuilding
Hochseilgarten.pdf
```

Der Zugriff auf ein Web-Element in einem Script (innerhalb oder außerhalb von combit CRM aufgerufen) erfolgt über die WebElements Eigenschaft des InputForm-Objekts. Siehe die entsprechenden Kapitel ListWebElements Objekt und WebElement Objekt.

> Wichtig: Beachten Sie hierbei, dass der Zugriff auf das View-Objekt nur auf dem vorgenannten Weg durchgeführt werden darf, nicht aber über das ActiveView-Objekt des Application-Objekts der Anwendung, da ein Web-Element seinen Inhalt immer asynchron lädt und ausführt.
>
> Wichtig: Bitte stellen Sie sicher, dass Web-Elemente bei Verwendung bestimmter Methoden (z. B. PrintLabel) prüfen, ob sich der aktuelle Datensatz im Bearbeitungsmodus befindet und in diesem Fall nichts weiter machen.

6.3 Zugriff auf cRM-Objekte

Der Zugriff auf die cRM-Objekte unterscheidet sich je nach verwendeter Web-Runtime.

6.3.1 Microsoft Chromium Edge Runtimes (WebView2)

> Hinweis: Die "Microsoft Chromium Edge Runtimes (WebView2)" Web-Runtime kann für die im combit CRM bereitgestellten Funktionen Info-Zentrale, Web-Ansichten und Web-Elemente verwendet werden. Für das Web-Panel wird diese immer verwendet.

Wichtig: Der Quellcode wird asynchron ausgeführt. Dies bedeutet, dass bei der Verwendung von cRM-Methoden und -Eigenschaften mit "await" gewartet werden muss, bevor der nächste Code-Bestandteil abgearbeitet werden kann.

Beispiel: oProject = await cRM.CurrentProject();

Bei der Verwendung von Objekten muss darauf geachtet werden, dass diese Objekte zum frühestmöglichen Zeitpunkt wieder freigegeben werden. Hierfür wird die Methode "Dispose" angeboten. Wenn die Objekte nicht eigenständig freigegeben werden, so passiert dies zu einem undefinierten Zeitpunkt durch die "Garbage Collection". Beachten Sie auch, dass in einem JavaScript-Programm keine COM-Objekte als globale Variablen gespeichert werden dürfen. Dies und das Freigeben von Objekten durch die "Garbage Collection" kann sonst unerwartetes Verhalten hervorrufen.

Der Zugriff auf das Application-Objekt der Anwendung erfolgt innerhalb der Webseite über das window.chrome.webview.hostObjects.combitCRM Objekt.

```
let oApplication = window.chrome.webview.hostObjects.combitCRM;
```

Der Zugriff auf das "zugehörige" View-Objekt der Anwendung erfolgt innerhalb der Webseite über das window.chrome.webview.hostObjects.combitCRM_Context.View Objekt.

```
let oView = window.chrome.webview.hostObjects.combitCRM_Context.View;
```

Der Zugriff auf den Namen des "zugehörigen" View-Objekts der Anwendung oder die ID eines Webseite erfolgt innerhalb der Web-Ansicht über das window.chrome.webview.hostObjects.combitCRM_Context.Name Objekt.

```
var sViewName = window.chrome.webview.hostObjects.combitCRM_Context.Name;
```

Das Schließen einer Info-Zentrale, einer Web-Ansicht oder eines Web-Elements wird nun im Ereignis CRM_WebViewClose abgebildet.

```
async function CRM_WebViewClose() { alert('WebView is closing...'); }
```

6.3.1.1 Beispiel: Name des Projekts anzeigen

```
<html>
<head>
<title>Beispiel</title>
<script>
    document.addEventListener('DOMContentLoaded', function () {
        (async () => {
            let sProjectName = await
window.chrome.webview.hostObjects.combitCRM.CurrentProject().Name;
            let mainPart = document.getElementById('mainPart');
            mainPart.innerHTML = sProjectName;
        })();
    });
</script>
</head>
<body>
<h1>Aktuelles Projekt:</h1>
<p id='mainPart'></p>
</body>
</html>
```

6.3.2 Microsoft Internet Explorer Control (veraltet)

Hinweis: Die "Microsoft Internet Explorer Control (veraltet)" Web-Runtime kann für Info-Zentrale, Web-Ansichten und Web-Elemente verwendet werden. Das Web-Panel ist mit dieser nicht kompatibel. Wir empfehlen die Verwendung der "Microsoft Chromium Edge Runtimes (WebView2)" Web-Runtime.

Wichtig: Die Webseiten werden dabei in einem Internet Explorer Control angezeigt. Dieses läuft intern im Kompatibilitätsmodus (siehe auch "<meta http-equiv="X-UA-Compatible" content="IE=edge" />") wodurch es

Unterschiede/Einschränkungen insbesondere bei der Darstellung von CSS im Vergleich zur Internet Explorer Standalone-Anwendung geben kann.

Bei Problemen mit der Darstellung einer Webseite, besonders wenn es sich um eine Seite handelt, welche den Inhalt per Script dynamisch erzeugt (siehe Beispiele in den Referenzlösungen), kann es hilfreich sein, die Option "Scriptfehler anzeigen" in den Eigenschaften des entsprechenden Web-Element Objekts zu aktivieren.

Der Zugriff auf das Application-Objekt der Anwendung erfolgt innerhalb der Webseite über das **window.external.Application** Objekt.

```
var oApplication = window.external.Application
```

Der Zugriff auf das "zugehörige" View-Objekt der Anwendung erfolgt innerhalb der Webseite über das **window.external.View** Objekt.

```
var oView = window.external.View
```

Der Zugriff auf den Namen des "zugehörigen" View-Objekts der Anwendung oder die ID eines Web-Elements erfolgt innerhalb der Webseite über das **window.external.Name** Objekt.

```
var sViewName = window.external.Name
```

6.3.2.1 Beispiel: Name des Projekts anzeigen

```
<html>
<head>
<title>Beispiel</title>
<script language="VBScript">
<!--
Function Display_Project
    Dim ocRM, oProject
    Set ocRM = window.external
    Set oProject = ocRM.CurrentProject
    document.write oProject.Name
     Set oProject = Nothing
     Set ocRM = Nothing
End Function
//-->
</script>
</head>
<body>
<h1>Aktuelles Projekt:</h1>
<p><script language="VBScript">Display_Project</script></p>
</body>
</html>
```

7 Empfohlene Vorgehensweisen

7.1 Arbeiten mit Rückgabewerten und Fehlerhandling

7.1.1 Was sind Rückgabewerte?

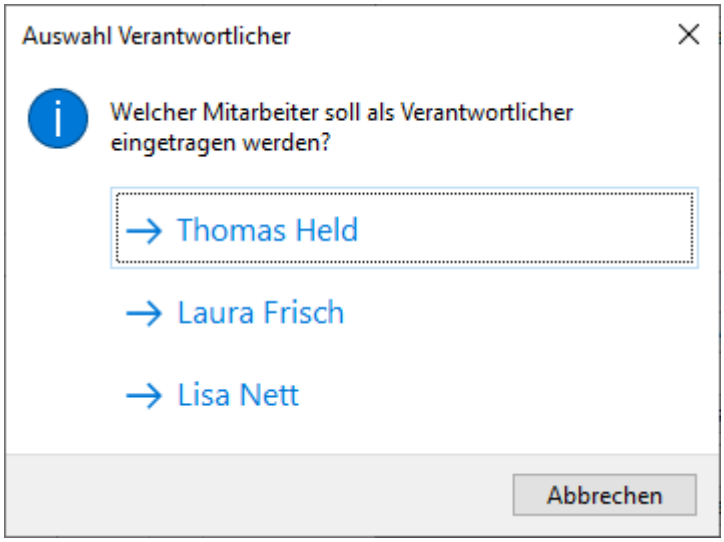
In der Programmierung ist der Rückgabewert der Wert, der von einer Eigenschaft oder Methode nach der Ausführung zurückgegeben wird. Innerhalb der SDK-Dokumentation findet sich zu jeder Eigenschaft oder Methode, die einen Rückgabewert besitzen, eine entsprechende Information.

7.1.2 Wofür können Rückgabewerte verwendet werden?

Ein einfaches Beispiel zur Verwendung von Rückgabewerten gibt es bei der Benutzerinteraktion. So kann beispielsweise mittels *cRM.DialogChoiceMessageBox()*-Methode eine Rückfrage an den combit CRM Anwender formuliert und eine Antwort erhalten werden.

```
' Anzeige eines Auswahldialogs mit drei möglichen Antworten und einer Abbrechen-
Schaltfläche
nResult = cRM.DialogChoiceMessageBox("Welcher Mitarbeiter soll als
Verantwortlicher eingetragen werden?", "Auswahl Verantwortlicher", "Thomas Held" &
vbTab & "Laura Frisch" & vbTab & "Lisa Nett", 1, true)
```

Dem Anwender wird mit der Ausführung dieses Scriptcodes folgender Dialog angezeigt:



Der Rückgabewert für die Methode *cRM.DialogChoiceMessageBox()* ist laut SDK-Dokumentation wie folgt beschrieben:

Long bzw. uint (der Rückgabewert entspricht den Konstanten einer MsgBox unter VBScript). Wenn der Nutzer eine Auswahl getroffen hat, dann wird ein OffSet von 100 einberechnet, d.h. bei Klick der zweiten Auswahl ist der Rückgabewert 102.

Wir erkennen hierbei, dass der Rückgabewert vom Typ „Long“ ist und bei erfolgreicher Wahl einen Wert von 101 bis 103 erhalten kann. Klickt der Nutzer auf die Schaltfläche „Abbrechen“ wird der Wert 2 zurückgeliefert, was der MessageBox-Konstante in VBScript „vbCancel“ entspricht. Siehe auch Kapitel Script-Konstanten.

Der Rückgabewert wird im Beispiel in der Variable nResult gespeichert, welche ausgewertet werden kann, nachdem der Anwender mit dem Dialog interagiert hat.

```
' Anzeige eines Auswahldialogs mit drei möglichen Antworten und einer Abbrechen-
Schaltfläche
nResult = cRM.DialogChoiceMessageBox("Welcher Mitarbeiter soll als
Verantwortlicher eingetragen werden?", "Auswahl Verantwortlicher", "Thomas Held" &
vbTab & "Laura Frisch" & vbTab & "Lisa Nett", 1, true)
```

```
' Wenn der Rückgabewert einem der erwarteten Werte entspricht, informiere den
Nutzer über seine Auswahl
If (nResult = 101) Then
    Call cRM.DialogMessageBox("Thomas Held ist nun für diesen Datensatz
verantwortlich.", "Erfolgreiche Auswahl", vbInformation)
ElseIf (nResult = 102) Then
    Call cRM.DialogMessageBox("Laura Frisch ist nun für diesen Datensatz
verantwortlich.", "Erfolgreiche Auswahl", vbInformation)
ElseIf (nResult = 103) Then
    Call cRM.DialogMessageBox("Lisa Nett ist nun für diesen Datensatz
verantwortlich.", "Erfolgreiche Auswahl", vbInformation)
ElseIf (nResult = 2) Then
    Call cRM.DialogMessageBox("Die Bearbeitung wurde abgebrochen.", "Bearbeitung
abgebrochen", vbInformation)
End If
```

7.1.3 Einfaches Fehlerhandling

Das Beispiel könnte jetzt noch durch ein einfaches Fehlerhandling ergänzt werden. Hierbei ergänzen wir zu den erwarteten Rückgabewerten einen Else-Zweig in die If-Bedingung, um Werte außerhalb des erwartbaren Bereichs „abfangen" zu können und demnach zu verhindern, dass diese im weiteren Scriptverlauf zu Problemen führen können.

```
' Anzeige eines Auswahldialogs mit drei möglichen Antworten und einer Abbrechen-
Schaltfläche
nResult = cRM.DialogChoiceMessageBox("Welcher Mitarbeiter soll als
Verantwortlicher eingetragen werden?", "Auswahl Verantwortlicher", "Thomas Held" &
vbTab & "Laura Frisch" & vbTab & "Lisa Nett", 1, true)

' Wenn der Rückgabewert einem der erwarteten Werte entspricht, informiere den
Nutzer über seine Auswahl
If (nResult = 101) Then
    Call cRM.DialogMessageBox("Thomas Held ist nun für diesen Datensatz
verantwortlich.", "Erfolgreiche Auswahl", vbInformation)
ElseIf (nResult = 102) Then
    Call cRM.DialogMessageBox("Laura Frisch ist nun für diesen Datensatz
verantwortlich.", "Erfolgreiche Auswahl", vbInformation)
ElseIf (nResult = 103) Then
    Call cRM.DialogMessageBox("Lisa Nett ist nun für diesen Datensatz
verantwortlich.", "Erfolgreiche Auswahl", vbInformation)
ElseIf (nResult = 2) Then
    Call cRM.DialogMessageBox("Die Bearbeitung wurde abgebrochen.", "Bearbeitung
abgebrochen", vbInformation)
' Wenn der Rückgabewert einem unerwarteten Wert entspricht, informiere den Nutzer
und beende das Script
Else
    Call cRM.DialogMessageBox("Es wurde ein nicht erwarteter Rückgabewert
erkannt." & vbCrLf & vbCrLf &_
    "Bitte wenden Sie sich an Ihren Administrator." & vbCrLf & vbCrLf &_
    "Das Script wird nun beendet.", "Unerwarteter Rückgabewert", vbCritical)
    Call WScript.Quit()
End If
```

7.1.4 Beispiel für das Bearbeiten eines Datensatzes

Das nachfolgende Beispiel zeigt wie eine Datensatzbearbeitung durchgeführt werden kann. Dabei werden die Rückgabewerte der unterschiedlichen Methoden (*Record.Lock()*, *Record.Save()*, *Record.Unlock()*) überprüft und der Anwender bei einem negativen Rückgabewert darüber informiert, dass ein bestimmter Vorgang nicht durchgeführt werden konnte.

```
' Zugriff auf den aktuell dargestellten Datensatz (einer Firmen-Ansicht)
Dim oRecord : Set oRecord =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentRecordSet.CurrentRecord

' Prüfen, ob der Zugriff möglich war
If (Not oRecord Is Nothing) Then
```

```
    ' Prüfen, ob der Datensatz für die Bearbeitung durch andere Nutzer gesperrt
werden kann
    If (oRecord.Lock() = True) Then

        ' Datensatz bearbeiten
        Call oRecord.SetContentsValueByName("Company", "combit Software GmbH")

        ' Prüfen, ob der Datensatz gespeichert werden kann
        If (oRecord.Save() = True) Then

            ' Prüfen, ob der Datensatz wieder freigegeben werden kann
            If (oRecord.Unlock() = True) Then

                ' Ab hier hat die Bearbeitung des Datensatzes funktioniert und das
Script kann beendet werden

            Else
                  ' Nutzer informieren, Script beenden
                Call cRM.DialogMessageBox("Der aktuelle Datensatz konnte nach der
Bearbeitung des Feldes ""Firma"" und dem Speichern nicht für die Bearbeitung durch
andere Nutzer freigegeben werden.", "Freigabe nicht möglich", vbExclamation &
vbOkOnly)
                Set oRecord = Nothing
                Call WScript.Quit()
            End If

        Else
             ' Nutzer informieren, Script beenden
            Call cRM.DialogMessageBox("Der aktuelle Datensatz konnte nach der
Bearbeitung des Feldes ""Firma"" nicht gespeichert werden.", "Speichern nicht
möglich", vbExclamation & vbOkOnly)
            Set oRecord = Nothing
            Call WScript.Quit()
        End If
    Else
        ' Nutzer informieren, Script beenden
        Call cRM.DialogMessageBox("Der aktuelle Datensatz konnte nicht für die
Bearbeitung durch andere Nutzer gesperrt werden.", "Sperren nicht möglich",
vbExclamation & vbOkOnly)
        Set oRecord = Nothing
        Call WScript.Quit()
    End If
Else

    ' Nutzer informieren, Script beenden
    Call cRM.DialogMessageBox("Der Zugriff auf den aktuellen Datensatz war nicht
möglich.", "Kein Zugriff auf Datensatz", vbExclamation & vbOkOnly)
    Set oRecord = Nothing
    Call WScript.Quit()

End If

Set oRecord = Nothing
```

7.2 Arbeiten mit vorgefertigten Methoden von combit

Innerhalb jedes Scripts-Unterordners einer der mitgelieferten combit CRM-Solutions befindet sich eine VBScript-Datei welche sich über Include-Befehle in neue und bestehende Scripte inkludieren lässt und Zugriff auf viele von combit vorbereitete Script-Funktionen bietet – mit entsprechenden Rückgabewerten und verlässlichen Methoden, inklusive Fehlerhandling: BasicCollection.vbs.

Das Inkludieren kann über folgenden Befehl durchgeführt werden (Achtung: Der Pfad kann sich in Ihrer Umgebung noch ändern):

```
<!--#include-once file="%PRJDIR%\Scripts\BasicCollection.vbs"-->
```

Die BasicCollection bietet hierbei eine reichhaltige Auswahl von vorgefertigten Methoden, welche Ihnen einiges an manueller Arbeit abnehmen können. Folgende Methoden stehen derzeit zur Verfügung:

BuildRelativePath	CheckForCriticalEventLogEntries	CheckAbortedWaitDlgAndSet-TextWithStatus
CheckAndLockRecord	CheckFile	CRMCheck
CheckGDPRColumn	CheckSolution	CheckViewConfigsObject
CreateContactDisplay	CreateDeleteRecordFeedback	CreateEventLogRecord
CreateNewRecordByRecordSet	cRMMsgBox	DeleteRecord
DialogChoiceMessageBoxByRecor-dSet	ExecuteSQLCommandRaw	ExportRecordSet
GDPRColumnsExists	GetActivityByUser	GetAddressByUser
GetCalendarWeek	GetCalendarWeekYear	GetCompaniesOrContactsByUser
GetCompanyOrContactSe-archTermByUser	GetContainerByInputForm	GetContainerRecordByInputForm
GetContentsValueByNameWith-NullSafe	GetCurrentLogGroupID	GetDateTimeByUser
GetEmailAddressByUser	GetFldInformationByViewNameAn-dFldName	GetIDByNameInRecordOrInput-Form
GetInputFormByActiveView	GetNextLogGroupID	GetProjectViewNameByUser
GetRecordByActiveView	GetRecordSetByViewConfig	GetRecordSetByViewName
GetViewConfigByViewConfigs	GetVisibleRecordSetByViewName	GoToRecordByEventLog
IsInt	MakeValidFilenameString	MaskSQLSpecialChars
OpenFile	OpenRecordByViewAndID	OpenViewByName
PrintListAndLabel	ReplaceRelativePath	ResetColumnInInputForm
ResetCompanyInformationInDictio-nary	ResetContactInformationInDictio-nary	SaveAndUnlockRecord
SelectFile	SetCompanyInformationInDictio-nary	SetCompanyOrContactFilterBySe-archTerm
SetContainerFilterByName	SetContactInformationInDictionary	SetContentsByNameInRecordOrIn-putForm
SetContentsByNameInRecord-ToNull	SetDBSystemTypeInDictionary	SetFilterDirectSQLProcess
UpdateActiveCompaniesAnd-ContactsViews	UpdateAllViews	UpdateView
UpdateViews		

Weitere Informationen zu den einzelnen Methoden erhalten Sie innerhalb der BasicCollection.vbs – öffnen Sie die Datei einfach in Ihrem bevorzugten Texteditor – hier befinden sich für jede Methode eine entsprechende Beschreibung, eine Übersicht der Parameter und des Rückgabewertes.

7.3 Ausführen eines Filters

Das nachfolgende Beispiel zeigt die von uns empfohlene Vorgehensweise bei der Ausführung von Filterausdrücken. Mit Hilfe von *RecordSet.MoveFirst()* nach dem Aufruf von *RecordSet.SetFilter…()* kann überprüft werden, ob mindestens ein Datensatz dem Filterausdruck entspricht und zugewiesen werden kann. Sollte dem so sein, dann kann mittels *RecordSet.HasMultipleRecords* geprüft werden, ob mehr als ein Datensatz dem Filterausdruck entspricht, sodass man seinen nachfolgenden Quellcode entsprechend anpassen kann (z. B. bei nur einem Datensatz: *Record.SendSingleMailDialog()*, bei mehr als einem Datensatz: *RecordSet.SendBulkMail()*).

Beispiel VBScript:

```
If (oRecordSet.SetFilter...()) Then
    If (oRecordSet.MoveFirst()) Then
        If (oRecordSet.HasMultipleRecords = True) Then
            ' ... mehrere Treffer
        Else
            ' ... ein Treffer
        End If
    Else
        ' ... keine Treffer
    End If
Else
    ' ... Fehler
End If
```

Beispiel C#-Script:

```
if (oRecordSet.SetFilter...())
{
    if (oRecordSet.MoveFirst())
    {
        if (oRecordSet.HasMultipleRecords = true)
        {
            // ... mehrere Treffer
        }
        else
        {
            // ... ein Treffer
        }
    }
    else
    {
        // ... keine Treffer
    }
}
else
{
    // ... Fehler
}
```

7.4 Datensatzinhalte Ausführen eines Filters

Das nachfolgende Beispiel zeigt die von uns empfohlene Vorgehensweise bei der Ausführung von Filterausdrücken. Mit Hilfe von *RecordSet.MoveFirst()* nach dem Aufruf von *RecordSet.SetFilter…()* kann überprüft werden, ob mindestens ein Datensatz dem Filterausdruck entspricht und zugewiesen werden kann. Sollte dem so sein, dann kann mittels *RecordSet.HasMultipleRecords* geprüft werden, ob mehr als ein Datensatz dem Filterausdruck entspricht, sodass man seinen nachfolgenden Quellcode entsprechend anpassen kann (z. B. bei nur einem Datensatz: *Record.SendSingleMailDialog()*, bei mehr als einem Datensatz: *RecordSet.SendBulkMail()*).

Beispiel VBScript:

```
If (oRecordSet.SetFilter...()) Then
    If (oRecordSet.MoveFirst()) Then
        If (oRecordSet.HasMultipleRecords = True) Then
            ' ... mehrere Treffer
        Else
            ' ... ein Treffer
        End If
    Else
        ' ... keine Treffer
    End If
Else
    ' ... Fehler
End If
```

Beispiel C#-Script:

```
if (oRecordSet.SetFilter...())
{
    if (oRecordSet.MoveFirst())
    {
        if (oRecordSet.HasMultipleRecords = true)
```

```
            {
                // ... mehrere Treffer
            }
            else
            {
                // ... ein Treffer
            }
        }
        else
        {
            // ... keine Treffer
        }
    }
    else
    {
        // ... Fehler
    }
```

7.5 Datensatzinhalte verändern während Bearbeitung

In Beispielen wird häufig darauf eingegangen, dass man Datensatzinhalte innerhalb eines Record-Objekts bearbeiten kann. Die dafür vorgesehenen Methoden *Record.GetContents(Value)By…()* oder *Record.SetContents(Value)By…()* werden daher ebenso häufig in den Beispielen verwendet.

Es kann nun jedoch sein, dass Datensatzinhalte verändert werden sollen, obwohl der Datensatz, auf den man üblicherweise zugreifen würde, noch im Bearbeitungsmodus ist bzw. sich sogar in der Neuanlage befindet, sodass entweder die Methoden *Record.SetContents(Value)By…()* und *Record.Save()* das interaktive Speichern dieses Datensatzes erfordern und so den Nutzer durch Hinweismeldungen aus dem Tritt bringen können, oder dass das Instanziieren des Record-Objekt fehlerhaft ist und die Record-Objektvariable möglicherweise leer bleibt.

Als Lösung bietet sich hierbei die Arbeit mit dem InputForm-Objekt an. Dieses Objekt repräsentiert die Eingabemaske, die dem Nutzer zum Zeitpunkt der Scriptausführung angezeigt wird und ermöglicht mit den Methoden *InputForm.GetContents(Value)By…()* und *InputForm.SetContents(Value)By…()* ähnliche Bearbeitungsmöglichkeiten, wie aus dem Record-Objekt bekannt.

Das Bearbeiten eines Feldinhalts kann wie folgt aussehen:

```
' Beibehalten des Bearbeitungsmodus
<!--pragma keepeditmode-->

' Zuweisen des InputForm-Objekt über die Methode CurrentInputForm() des View-
Objekts
Dim oInputForm : Set oInputForm =
cRM.CurrentProject.ActiveViews.ActiveView.CurrentInputForm(2) ' Wert 2:
Beibehalten des aktuellen Bearbeitungsmodus

' Beschreiben des Feldes "Firma" mit dem Wert "combit Software GmbH" innerhalb der
Eingabemaske
Call oInputForm.SetContentsValueByName("Firma", "combit Software GmbH")

' Speichern der Änderungen innerhalb der Eingabemaske ohne interaktive Rückmeldung
an den Anwender
Call oInputForm.Save()
```

7.6 HTTP-Requests ausführen

Die nachfolgenden Beispiele zeigen, wie ein HTTP POST-Request mit Hilfe von combit CRM-Methoden durchgeführt werden kann. Dabei werden Informationen im JSON-Format angefordert. Weitere Informationen finden Sie auch in den Methodenbeschreibungen der HTTP*-Methoden innerhalb des Kapitels Application/cRM Objekt.

Mit den Informationen für URL, Header und Data wird die cRM.HTTPPost Methode aufgerufen, um Daten eines Webservice abzurufen. Die Antwort des Webservers wird in einer Variablen gespeichert und mit Methoden aus dem inkludierten Scripten v_Script und v_JSON verarbeitet. Beide Scripte befinden sich im Scripts-Unterordner Ihrer combit CRM-Installation. Die aufbereitete Antwort wird in separaten Messageboxen mit dem Statuscode und der Antwort des Webservers angezeigt.

Beispiel VBScript:

```
<!--#include-once file="...\Scripts\v_JSON\v_Script.vbs"-->
<!--#include-once file="...\Scripts\v_JSON\v_JSON.vbs"-->

Const sURL = "https://httpbin.org/post"
Const sHeader = "[{""key"": ""cumstom_key"", ""value"": ""custom_value""},
{""key"": ""Content-Type"", ""value"": ""application/json""}]"
Const sData = "[{""key"": ""test_key"", ""value"": ""test_value""}]"
Dim sAnswer : sAnswer = cRM.HTTPPost(sURL, sHeader, sData)

Set json = New v_JSON
Call json.FromString(sAnswer)

MsgBox "Statuscode: " & CStr(json.Item("status"))
MsgBox "Response: " & json.Item("response")
```

Beispiel C#-Script:

```
//<!--#include-once file=@"...\Scripts\v_JSON\v_Script.vbs"-->
//<!--#include-once file=@"...\Scripts\v_JSON\v_JSON.vbs"-->

using System.Windows.Forms;
using combit.cRM.COM;
using Newtonsoft.Json;

cRMApplication cRM = new cRMApplication(EApplicationStartType.GetActiveobject);
string url = "https://httpbin.org/post";
string header = @"[{""key"": ""cumstom_key"", ""value"": ""custom_value""},
{""key"": ""Content-Type"", ""value"": ""application/json""}]";
string data = @"[{""key"": ""test_key"", ""value"": ""test_value""}]";
string answer = cRM.HTTPPost(url, header, data);

var responseDefinition = new { status = 0, response = ""};
var response = JsonConvert.DeserializeAnonymousType(answer, responseDefinition);
MessageBox.Show(response.status.ToString());
MessageBox.Show(response.response);
```

7.7 Arbeiten mit JavaScript

Bei der Arbeit mit JavaScript innerhalb der Info-Zentrale, in Web-Ansichten, den Web-Elementen oder dem Web-Panel gibt es konzeptuell bestimmte Punkte zu beachten.

Zunächst ist es wichtig zu wissen, dass die Bereitstellung des combit CRM-/Application-Objects nur innerhalb der Info-Zentrale, in Web-Ansichten, den Web-Elementen oder dem Web-Panel möglich ist. Dies bedeutet im Umkehrschluss, dass in einem externen Chromium-Browser nicht auf das Objektmodel von combit CRM zugegriffen werden kann.

Bitte beachten Sie zudem, dass bei der Arbeit mit Objekten der Aufruf des gewünschten Objektes synchron mit dem Befehl "await" durchgeführt wird und erzeugte Objekte in umgekehrter Reihenfolge der Erzeugung mittels ".Dispose"-Methode wieder aufgeräumt werden müssen. Der Einsatz des Garbage Collectors birgt hierbei die Gefahr, dass nicht die korrekte Reihenfolge für das Aufräumen der Objekte eingehalten wird und so Fehler und unerwartete Zustände auftreten können.

Beispiele für den praktischen Einsatz liefern die für die "Large"-Solution bereitgestellten Scripte, die in den Unterordnern "InfoBoard", "InfoZentrale" und "Web-Elemente" gefunden werden können.

Weitere Informationen finden Sie auch im Kapitel **Microsoft Chromium Edge Runtimes (WebView2)**.

8 Protokollhandler crm://

Über das crm://-Protokoll können verschiedene Aktionen, auch außerhalb der cRM-Applikation, initiiert werden.

8.1 Grundlegender Aufbau

crm://<Kommando>?<Parameter>&<Parameter>&{Parameter}& …

Dabei sind Kommandos und Parameter zwingend in spitzen Klammern zu setzen. Parameter in geschweiften Klammern können optional definiert werden. Für jeden Parameter gibt es eine lange und eine kurze Schreibweise, welche beide verwendet werden können. Es wird zudem eine HTML-Codierung verwendet. Kommandos und Parameter können miteinander kombiniert werden.

8.2 Übersicht der Kommandos

Kommando	Parameter	Beschreibung
execute	Path oder P	Führt ein Script aus. Beispiel: crm://execute?P=%APPDIR%\Scripts\cRMTaschenrechner.vbs
filter	Name oder N Desc oder D Optional: View oder V Type oder T	Ruft einen Filter anhand des übergebenen Scriptnamens (Name / N) oder der Beschreibung (Desc / D) auf. Optional können eine Ansicht (View / V) und ein Verknüpfungstyp (Type / T, mögliche Werte: "AND", "OR") definiert werden. So ist es auch möglich verschiedene Filter miteinander zu verknüpfen. Beispiel: crm://filter?View=Firmen&D=Alle%20Kunden%20auswählen
findrecordbymail	Mail oder M oder EMail oder E Optional: View oder V Reduce oder R Current oder C oder Type oder Ty	Sucht alle Datensätze, bei denen in einem E-Mail-Feld eine bestimmte E-Mail-Adresse (Mail / M / EMail / E) vorhanden ist. Optional können folgende Parameter verwendet werden: - View / V: Ansicht - Reduce / R: Wenn True, wird die E-Mail-Adresse schrittweise in bis zu vier Durchläufen durch Auslassen von Top-Level-Domain, Domain, Lokalteil durchsucht, mögliche Werte: "True", "False" - Current / C / Type / Ty: Suche innerhalb des aktuellen Filters, mögliche Werte: "True", "False" Beispiel: crm://findrecordbymail?View=Firmen&Mail=info%40luna-aventuras.net
findrecordbyphone	Phone oder P Optional: View oder V Reduce oder R	Sucht alle Datensätze, bei denen in einem Telefon-Feld eine bestimmte Telefonnummer (Phone / P) vorhanden ist. Optional können folgende Parameter verwendet werden: - View / V: Ansicht - Reduce / R: Wenn True, dann wird die Telefonnummer schrittweise um bis zu 4 Stellen verkürzt, falls es nicht bereits Treffer gab, mögliche Werte: "True", "False" - Current / C / Type / Ty: Suche innerhalb des aktuellen Filters, mögliche Werte: "True", "False"

	Current oder C oder Type oder Ty	Beispiel: crm://findrecordbyphone?View=Firmen&Phone=07531%2F0999999-0
goto	Ref oder R	Öffnet einen Datensatz. Folgender Aufbau wird für den Datensatzverweis verwendet: <Kommando> = <Projekt ID> | <Ansichtsname> | <Familienname der Ansicht> | <Primärschlüsselfeld> | <Primärschlüssel> | <Datensatzbeschreibung> Beispiel: crm://goto?Ref=76b5fce6e6684105a6ef8bb16a15b03a%7cFirmen%7c%7cID%7c54ef9e5a0dc14fdc9c690333ecc81ec5%7cWild-%20und%20Freizeitpark%20Stockach
goto	View oder V Optional: ViewMode oder M	Öffnet eine Ansicht, optional kann der Modus der Ansicht definiert werden. Beispiel: crm://goto?View=Kontakte&ViewMode=2 Als mögliche Werte für den Modus der Ansicht können die im View-Objekt definierten Werte der Eigenschaft ViewMode verwendet werden.
instantreport	Name oder N Desc oder D Optional: Media oder M Output oder O Silent oder S	Ruft einen Sofortbericht anhand des übergebenen Scriptnamens (Name / N) oder der Beschreibung (Desc / D) auf. Beispiel: crm://instantreport?D=Firmen%20-%20Branchenverteilung crm://instantreport?D=Firmen%20-%20Branchenverteilung&M=PDF&C:\Export-Ergebnisse\Firmen%20-%20Branchenverteilung.pdf
manflt	FilePath oder F Optional: View oder V	Führt einen manuellen Filter anhand eines Pfads zu einer .tag-Datei auf. Optional kann eine Ansicht definiert werden, die statt der aktuellen Ansicht verwendet wird. Beispiel: crm://manflt?F=C:\Manuelle%20Filter\Verkaufschancen.tag&V=Verkaufschancen
phone	Number oder N	Ruft die übergebene Rufnummer über den phone manager an. Beispiel: crm://phone?Number=07531906010
phone	Ref oder R und Field oder F	Ruft die Rufnummer eines Datensatzes an. Beispiel: crm://phone?Ref=76b5fce6e6684105a6ef8bb16a15b03a%7cFirmen%7c%7cID%7c54ef9e5a0dc14fdc9c690333ecc81ec5%7cWild-%20und%20Freizeitpark%20Stockach&Field=Telefon

search	Text oder T Optional: View oder V Fields oder F Mode oder M Type oder Ty Case oder C	Führt eine Suche anhand eines übergebenen Textes aus. Optional können die Ansicht (View / V), die zu durchsuchenden Felder (Fields / F, komma- oder semikolonsepariert), der Modus (Mode / M, "normal", "exact", "wildcard", "phonetic", "contains"), der Typ (Type / Ty, "AND", "OR") und ob Case-Sensitiv (Case / C, "0", "1") definiert werden. Beispiel: crm://search?Text=Maier crm://search?F=Name,Vorname&M=wildcard&T=L*&Ty=OR

9 Export-Optionen für Print-Methoden

In allen Print-Methoden ist es möglich Export-Optionen zu definieren, die über die Benutzeroberfläche unter Umständen nicht gewählt werden können.

Aufbau:
Print-Methode(<Export-Format>:<Export-Format der Export-Option>|<Export-Option>=<Wert der Export-Option>;<Export-Format der Export-Option2>|<Export-Option2>=<Wert der Export-Option2>;...)

Beispiel:
PrintReport("PDF:PDF|PDF.Author=combit Software GmbH;PDF|PDF.Conformance=pdf17",...)

9.1 Unterstützte Optionen

Export.ShowResult: Spezifizert, ob im Anschluss an den Export die mit der Dateiendung verknüpfte Anwendung automatisch gestartet werden soll.

Wert	Bedeutung
0	Keine Anzeige des Ergebnisses
1	Führt ein *ShellExecute()* für die erstellte Export-Datei aus, so dass die mit der Dateiendung verknüpfte Anwendung automatisch gestartet wird.
Default	0

PRN.Copies: Damit kann bestimmt werden, wie viele Kopien gedruckt werden sollen. Bei Etiketten handelt es sich um die jeweilige Anzahl der Etiketten, die auch in der Vorschau und in Exportformaten berücksichtigt wird. Bei Listen und Karteikarten hingegen handelt es sich um die Anzahl der Exemplare, die der Druckertreiber beim Druck erstellen soll. Das Verhalten dieser Option entspricht der Eigenschaft "Exemplare" im Druckoptionendialog.

Beispiel:
PrintLabel("PRN:PRN|PRN.Copies=4",...)

PDF.Title: Spezifiziert den Titel des zu generierenden PDF-Dokuments.

PDF.Subject: Spezifiziert das Thema des zu generierenden PDF-Dokuments. Default: leer.

PDF.Keywords: Spezifiziert die Stichwörter des zu generierenden PDF-Dokuments. Default: leer.

PDF.Conformance: Wenn dieser Parameter gesetzt ist, kann die zu verwendende PDF-Version eingestellt werden. Bei aktivierter Verschlüsselung (siehe *PDF.Encryption.EncryptFile*) ergibt sich die Verschlüsselungsstärke automatisch. Es stehen diverse Optionen zur Verfügung, die im Folgenden erläutert werden.

Wert	Bedeutung
pdf10	PDF Version 1.0
pdf11	PDF Version 1.1
pdf12	PDF Version 1.2
pdf13	PDF Version 1.3
pdf14	PDF Version 1.4 (entspricht Acrobat 5)
pdf15	PDF Version 1.5
pdf16	PDF Version 1.6 (entspricht Acrobat 7)
pdf17	PDF Version 1.7 (ISO 32000-1)
pdf20	PDF Version 2.0 (ISO 32000-2)
pdfa1b	PDF/A-1b (ISO 19005-1, Level B Konformität)
pdfa1a	PDF/A-1a (ISO 19005-1, Level A Konformität)
pdfa2b	PDF/A-2b (ISO 19005-2, Level B Konformität)
pdfa2u	PDF/A-2u (ISO 19005-2, Level U Konformität)
pdfa2a	PDF/A-2a (ISO 19005-2, Level A Konformität)
pdfa3b	PDF/A-3b (ISO 19005-3, Level B Konformität)
pdfa3u	PDF/A-3u (ISO 19005-3, Level U Konformität)
pdfa3a	PDF/A-3a (ISO 19005-3, Level A Konformität)
Default	pdf17

PDF.Encryption.EncryptFile: Wenn dieser Parameter gesetzt ist, wird die Ergebnisdatei verschlüsselt. Die Verschlüsselungsart ergibt sich automatisch anhand der ausgewählte PDF-Version (siehe *PDF.Conformance*). Dann stehen diverse weitere Optionen zur Verfügung, die im Folgenden erläutert werden.

Wert	Bedeutung
0	Datei nicht verschlüsseln
1	Datei verschlüsseln Wichtige Hinweise zur Verschlüsselung der ausgewählten PDF-Version: pdf10, pdfa[x]: keine Verschlüsselung pdf11, pdf12, pdf13: RC4 mit einer Schlüssellänge von 40 pdf14: RC4 mit einer Schlüssellänge von 128 pdf15, pdf16, pdf17: AES mit einer Schlüssellänge von 128 pdf20: AES mit einer Schlüssellänge von 256
Default	0

PDF.Encryption.EnablePrinting: Wenn dieser Parameter gesetzt ist, kann die Datei trotz Verschlüsselung gedruckt werden. Nur wirksam, wenn PDF.Encryption.EncryptFile auf "1" gesetzt wird.

Wert	Bedeutung
0	Drucken ist nicht möglich
1	Drucken ist möglich
Default	0

PDF.Encryption.EnableChanging: Wenn dieser Parameter gesetzt ist, kann die Datei trotz Verschlüsselung bearbeitet werden. Nur wirksam, wenn PDF.Encryption.EncryptFile auf "1" gesetzt wird.

Wert	Bedeutung
0	Bearbeiten ist nicht möglich
1	Bearbeiten ist möglich
Default	0

PDF.Encryption.EnableCopying: Wenn dieser Parameter gesetzt ist, können Teile der Datei trotz Verschlüsselung in die Zwischenablage übernommen werden. Nur wirksam, wenn PDF.Encryption.EncryptFile auf "1" gesetzt wird.

Wert	Bedeutung
0	Kopieren ist nicht möglich
1	Kopieren ist möglich
Default	0

PDF.OwnerPassword: Das Besitzerpasswort für die verschlüsselte Datei. Dieses wird benötigt, um die Datei bearbeiten zu können. Wenn kein Passwort angegeben wird, wird die Datei mit einem zufälligen Passwort verschlüsselt. Wir empfehlen, immer ein geeignetes Passwort explizit zu wählen.

PDF.UserPassword: Das Benutzerpasswort für die verschlüsselte Datei. Dieses wird benötigt, um auf eine verschlüsselte Datei zugreifen zu können. Wenn kein Passwort angegeben wird, ist der Zugriff ohne Passwort möglich (evtl. mit Einschränkungen, s. o.).

PDF.Author: Setzt das Author-Tag in der PDF-Datei. Default: leer.

PDF.ZUGFeRDConformanceLevel: Legt das ZUGFeRD Conformance Level fest.

Wert	Bedeutung
BASIC	Einfache Rechnungen mit strukturierten Daten. Weitere Informationen als Freitext möglich.
EXTENDED	Branchenübergreifender Rechnungsaustausch mit erweiterten strukturierten Daten.
COMFORT	Vollautomatische Rechnungsverarbeitung mit strukturierten Daten.Nur für ZUGFeRD 1.0 relevant. Für ZUGFeRD 2.0/2.1 bitte "EN 16931" verwenden.
EN 16931	Vollautomatische Rechnungsverarbeitung mit strukturierten Daten. Nur für ZUGFeRD 2.0/2.1 gültig. Identisch mit XRechnung 1.0.
BASIC WL	Das Profil ist aus Gründen der Übereinstimmung beider Standards ZUGFeRD und Factur-X auch in ZUGFeRD 2.0/2.1 enthalten, stellt in Deutschland jedoch keine vollwertige Rechnung im Sinne des UStG dar.

Wert	Bedeutung
MINIMUM	Das Profil ist aus Gründen der Übereinstimmung beider Standards ZUGFeRD und Factur-X auch in ZUGFeRD 2.0/2.1 enthalten, stellt in Deutschland jedoch keine vollwertige Rechnung im Sinne des UStG dar.
XRECHNUNG	Das Profil erfüllt die spezifischen Anforderungen der öffentlichen Verwaltung in Deutschland. Es rrfüllt die Vorgaben der europäischen Norm EN16931 und auch die nationalen Geschäftsregeln und verwaltungsspezifischen Bestimmungen des Standards XRechnung.
Default	BASIC

PDF.ZUGFeRDVersion: Legt die ZUGFeRD Version fest.

Wert	Bedeutung
1.0	Es wird ZUGFeRD in der Version 1.0 forciert (Dateiname: ZUGFeRD-invoice.xml).
2.0	Es wird ZUGFeRD in der Version 2.0 forciert (Dateiname: ZUGFeRD-invoice.xml).
2.1	Es wird Factur-x forciert, was ZUGFeRD in der Version 2.1 entspricht. (Dateiname: factur-x.xml)
Default	2.0

PDF.ZUGFeRDXmlPath: Gibt den Pfad einer ZUGFeRD konformen XML-Datei an, die in das Ergebnis-PDF eingebettet werden soll. Der Dateinamen muss dabei der eingestellten ZUGFeRD-Version entsprechen (siehe PDF.ZUGFeRDVersion). Die XML-Datei muss zuvor von der Anwendung selber erstellt worden sein.

LL.FAX.RecipName: Definiert den Empfängernamen.

LL.FAX.RecipNumber: Definiert die Empfängerfaxnummer.

LL.FAX.SenderName: Definiert den Absendername.

LL.FAX.SenderCompany: Definiert die Absenderfirma.

LL.FAX.SenderDepartment: Definiert die Absenderabteilung.

LL.FAX.SenderBillingCode: Definiert den Absenderverrechnungscode.

9.2 ZUGFeRD-Export Beispiel

Beispiel VBScript:

```
Dim sZUGFeRDXmlPath
sZUGFeRDXmlPath = "C:\temp\ZUGFeRD-invoice.xml"

Dim sExportTarget, sExportOptions
sExportTarget = "PDF"
sExportOptions = "PDF|PDF.ZUGFeRDXmlPath=" & sZUGFeRDXmlPath

Call oCurrentRecord.PrintReport(sExportTarget & ":" & sExportOptions,
sProjectFile, bSilent, sExportPath & sExportFileName, bUseModalPreviewWindow)
```

Beispiel C#-Script:

```
string ZUGFeRDXmlPath = @"C:\temp\ZUGFeRD-invoice.xml";

string exportTarget = "PDF";
string exportOptions = "PDF|PDF.ZUGFeRDXmlPath=" + ZUGFeRDXmlPath;

currentRecord.PrintReport(exportTarget + ":" + exportOptions, projectFile, silent,
exportPath + exportFileName, useModalPreviewWindow);
```

10 Methodenübergreifende Parameter

10.1 Ausgabe-Medium

Zur Verfügung stehen folgende Parameterwerte:

Wert	Beschreibung
PRN	Drucker
PRV	Vorschau
PDF	Adobe PDF-Format
XHTML	XHTML/CSS-Format
MHTML	Multi-Mime HTML-Format
XLS	Microsoft Excel-Format
DOCX	Microsoft Word-Format
RTF	Rich Text-Format (RTF)
XPS	Microsoft XPS-Format
PICTURE_MULTITIFF	Multi-TIFF-Grafik
PICTURE_TIFF	TIFF-Grafik
PICTURE_PNG	PNG-Grafik
PICTURE_PNG_CROP	Automatisch um überflüssigen weißen Rahmen beschnittene PNG-Grafik
PICTURE_JPEG	JPEG--Grafik
PICTURE_JPEG_CROP	Automatisch um überflüssigen weißen Rahmen beschnittene JPEG-Grafik
PICTURE_BMP	Bitmap-Grafik
PICTURE_EMF	Enhanced Metafile-Grafik (EMF)
PICTURE_EMF_CROP	Automatisch um überflüssigen weißen Rahmen beschnittene Enhanced Metafile-Grafik (EMF)
FILE	Datei
HTML	HTML-Format
JQM	HTML jQuery Mobile-Format (nur für Listen)
TTY	Nadeldrucker-Format (TTY)
PPTX	Microsoft PowerPoint-Format
SVG	Scalable Vector Graphics Format
TXT	Text-Format (CSV)
TXT_LAYOUT	Text-Format (Layout-erhaltend)
XML	XML-Format

11 Dokumentenmanagementsysteme integrieren

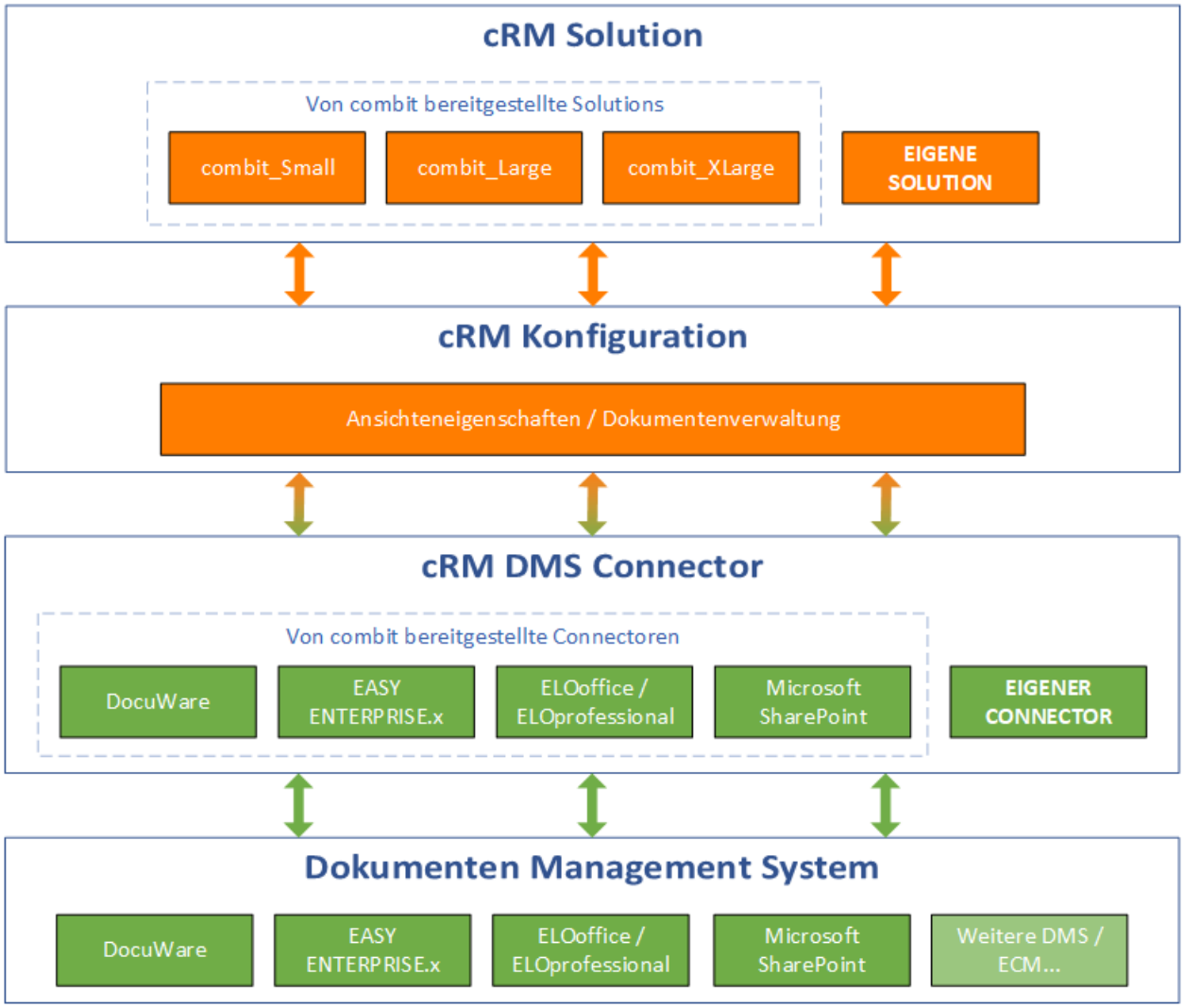
Neben der integrierten Dokumentenverwaltung (Dateiverweise oder eingebettete Dokumente) bietet combit CRM die Möglichkeit auch externe Dokumentenmanagementsysteme (DMS) als Dokumentenverwaltung zu verwenden. Eine aktuelle Liste der verfügbaren, und bereits von combit CRM unterstützen DMS, finden Sie auch in den Systemvoraussetzungen im Handbuch.

Sollte nun jedoch ein bisher unbekanntes oder spezielles, eigenes DMS für die Dokumentenverwaltung des cRMs verwendet werden, steht eine gesonderte Programmierschnittstelle zur Verfügung. Es können durch diese einheitliche Schnittstelle auch nicht out-of-the-box unterstützte Dokumentenmanagementsysteme angebunden werden.

Die Implementierung eines derartigen DMS Connectors wird in der Regel durch einen Solution- oder Integrations-Partner oder Inhouse-IT-Entwickler durchgeführt. Es erfolgt dadurch eine Integration des angebundenen DMS in die combit CRM Programmoberfläche. Das heißt die gewohnten Arbeitsabläufe in der CRM-Software und im DMS bleiben erhalten und werden miteinander verbunden.

11.1 Grundlegender Aufbau

Der grundlegende Aufbau der in combit CRM enthaltenen DMS-Integration oder die Implementierung eines eigenen DMS-Connectors sieht wie folgt aus:



Die Implementierung eines DMS-Connectors kann in C#/VB.NET aber auch C++ oder Delphi erfolgen. Die notwendige Beschreibung der Schnittstelle erhalten Sie auf Anfrage direkt von combit.

12 E-Mail-Tools integrieren

Neben dem integrierten E-Mail-Versand bietet combit CRM auch die Möglichkeit externe E-Mail-Versandsysteme, die für professionelles und leistungsfähiges Kampagnenmanagement ausgelegt sind, zu verwenden.

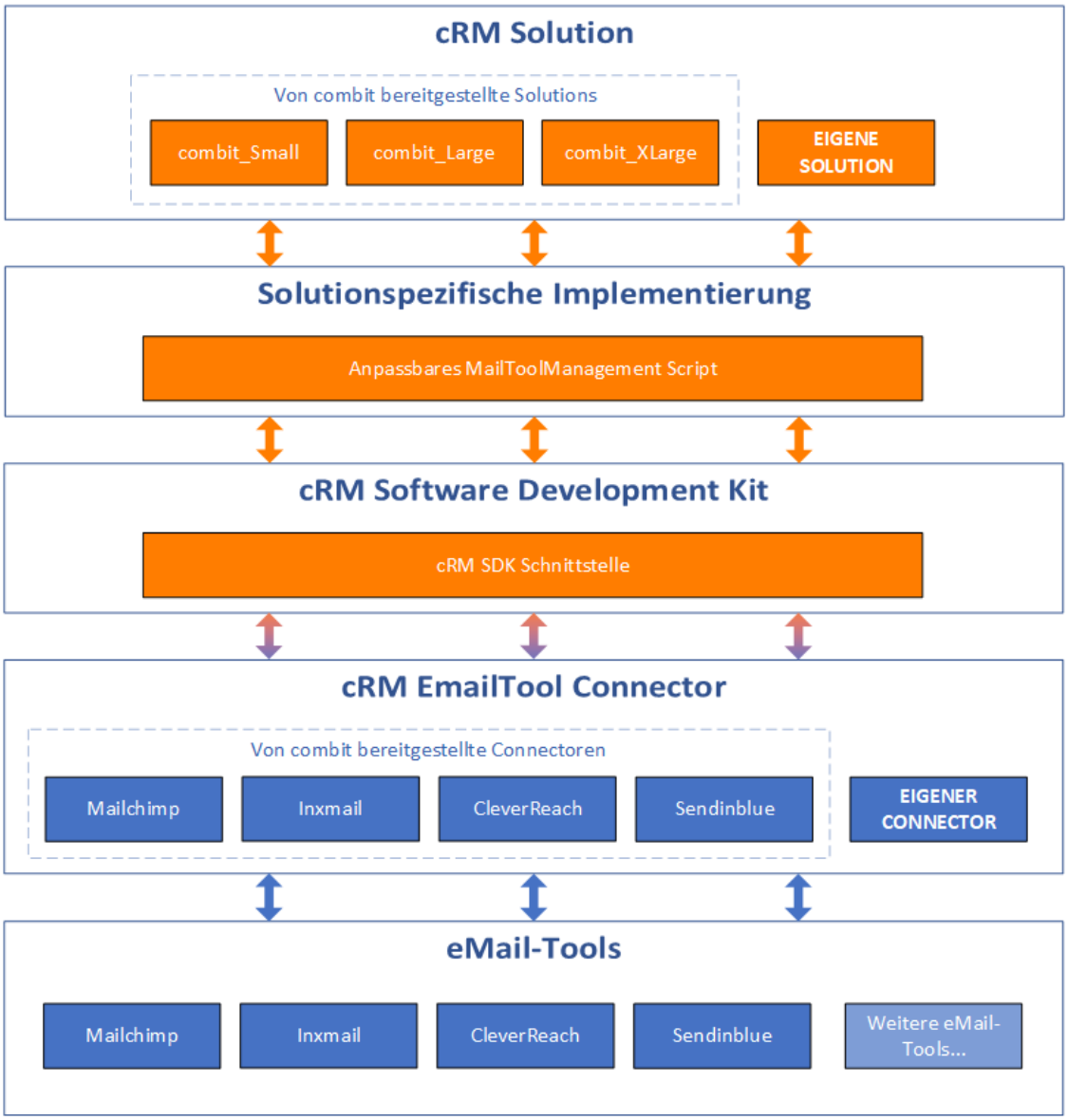
Eine aktuelle Liste der verfügbaren, und bereits von combit CRM unterstützen Systeme, finden Sie auch im EmailTool-Objekt dieser SDK-Dokumentation.

Sollte nun jedoch ein bisher unbekanntes oder spezielles eigenes E-Mail-Versandsystem verwendet werden, steht eine gesonderte Programmierschnittstelle zur Verfügung. Es können durch diese einheitliche Schnittstelle auch nicht out-of-the-box unterstützte E-Mail-Versandsysteme angebunden werden. Beachten Sie jedoch, dass immer auch eine solutionspezifische Integration über das SDK (siehe EmailTool-Objekte) erforderlich ist.

Die Implementierung eines derartigen EmailTool Connectors wird in der Regel durch einen Solution- oder Integrations-Partner oder Inhouse-IT-Entwickler durchgeführt. Es erfolgt dadurch eine Integration des angebundenen E-Mail-Versandsystems in das combit CRM SDK.

12.1 Grundlegender Aufbau

Der grundlegende Aufbau der in combit CRM enthaltenen EmailTool-Integration oder die Implementierung eines eigenen EmailTool-Connectors sieht wie folgt aus:



Die Implementierung eines EmailTool Connectors kann in C#/VB.NET erfolgen. Die notwendige Beschreibung der Schnittstelle erhalten Sie auf Anfrage direkt von combit.

12.2 Zugriff auf cRM-Objekt

Bitte beachten Sie, dass, um Scripte außerhalb der Anwendung starten zu können, das Application-Objekt "cRM.Application" benötigt wird. Siehe auch Beispiel "CreatecRMApplicationInstance.vbs" sowie den entsprechenden Eintrag im Kapitel "Scripte und Programmierreferenz" des Handbuchs. In der Regel sind die Scripte nur innerhalb der Anwendung lauffähig.

13 Menü-IDs

Diese Menü-IDs können zum Aufruf des jeweiligen Befehls über die InvokeMenu Methoden der Application, phonemanager und View Objekte verwendet werden. Außerdem können die Befehle in den Projekt-Ereignissen verwendet werden.

13.1 Menüband

13.1.1 Projekt-Menü

Befehl in Menüband	ID
Datei > Speichern	57603
Datei > Speichern unter	57604
Datei > Öffnen > Öffnen	57601
Datei > Öffnen > Aktualisieren	33417
Datei > Öffnen > Start-Center	33246
Datei > Neu	57600
Datei > Verwalten > Eigenschaften	32905
Datei > Verwalten > Ereignisse	32987
Datei > Verwalten > Aktionen	31002
Datei > Verwalten > Firmenstammdaten	33129
Datei > Verwalten > Navigationsstruktur > Ansichten	33100
Datei > Verwalten > Navigationsstruktur > Eigene Sofortberichte	33533
Datei > Verwalten > Navigationsstruktur > Globale Sofortberichte	33532
Datei > Verwalten > Navigationsstruktur > Eigene Filter	33531
Datei > Verwalten > Navigationsstruktur > Globale Filter	33530
Datei > Verwalten > Automatische Zähler	32897
Datei > Verwalten > Datenbankstruktur	33016
Datei > Verwalten > Neue Ansicht erstellen	32800
Datei > Verwalten > Projektimport	33243
Datei > Verwalten > Reorganisieren	33201
Datei > Verwalten > Bereitstellung für WebAccess	33206
Datei > Verwalten > Datensynchronisation > Exchange-Adressbuchssynchronisation	33548
Datei > Verwalten > Datensynchronisation > Exchange-Terminsynchronisation	33551
Datei > Verwalten > Datensynchronisation > Datenbankreplikation	33401
Datei > Verwalten > Sichern und Wiederherstellen	33402
Datei > Verwalten > Datensätze neu geokodieren	33518
Datei > Verwalten > Benutzerverwaltung	32805
Datei > Verwalten > Microsoft Exchange-Anmeldeverwaltung	33552
Datei > Verwalten > Tools > address pick-up	33358
Datei > Verwalten > Tools > E-Mail-Autopilot	33359
Datei > Verwalten > Tools > LDAP-Server	33360
Datei > Verwalten > Tools > Workflow-Server	33361
Datei > Verwalten > Tools > Lizenz verwalten	33363
Datei > Verwalten > Tools > Migration von address manager 15 oder älter	33365
Datei > Verwalten > Tools > Debug-Tool	33362
Datei > Papierkorb	33526
Datei > Hilfe	57666
Datei > Hilfe > Support > Supportanfrage stellen	32991
Datei > Hilfe > Support > Knowledgebase	32942
Datei > Hilfe > Support > Foren	32941
Datei > Hilfe > Support > Support-Bereich	32940
Datei > Hilfe > Support > An Onlinesitzung teilnehmen	32229
Datei > Hilfe > Kontakt	32236

Datei > Hilfe > Aktualitätsprüfung im Internet	32939
Datei > Hilfe > Onlineregistrierung	32938
Datei > Hilfe > Ideenwerkstatt	33400
Datei > Hilfe > Info über combit CRM	57664
Datei > Konto > Allgemein	32915
Datei > Konto > Terminverwaltung	33091
Datei > Konto > Akustische Signale	32914
Datei > Konto > Kennwort ändern	32937
Datei > Konto > DMS-Anmeldung	33042
Datei > Konto > E-Mail-Tool-Anmeldung	33440
Datei > Konto > Datenbankanmeldung	32899
Datei > Beenden	57665
Fenster > Ansicht > Aktualisieren	32784
Fenster > Wechseln zu > Ansicht	33186
Fenster > Wechseln zu > Info-Zentrale	33131
Fenster > Wechseln zu > Termine & Aufgaben	33091
Fenster > Wechseln zu > Anrufe	33127
Fenster > Wechseln zu > Papierkorb	33526
Fenster > Anzeigen > Navigation	59512
Fenster > Anzeigen > Dubletten	59513
Fenster > Anzeigen > Suchergebnisse	59525
Fenster > Anzeigen > Statusleiste	59393
Fenster > Anzeigen > Favoriten	59530
Fenster > Anzeigen > Verlauf	59531
Fenster > Anzeigen > Sofortberichte	59532
Fenster > Anzeigen > Filter	59533
Fenster > Anzeigen > Überwachungsergebnisse	59534
Fenster > Anzeigen > Aktive Benutzer	59535
Fenster > Anzeigen > Aktionen	59536
Fenster > Anzeigen > Web-Panel	59537
Fenster > Fenster > Alle schließen	33148
Fenster > Fenster > Fenster wechseln	32933
Fenster > Fenster > Fensterlayout zurücksetzen	33576

13.1.2 Ansichten-Menü

Befehl in Menüband	**ID**
Start > Anzeigen > Ansicht > Eingabemaske	32783
Start > Anzeigen > Ansicht > Übersichtsliste	32782
Start > Anzeigen > Ansicht > Bericht	32815
Start > Anzeigen > Ansicht > Web	32989
Start > Anzeigen > Ansicht > Vertikal geteilt	33404
Start > Anzeigen > Ansicht > Horizontal geteilt	33403
Start > Anzeigen > Ansicht > Landkarte	33432
Start > Datensatz > Neu	32781
Start > Datensatz > Neu > Duplizieren	32809
Start > Datensatz > Bearbeiten	32807
Start > Datensatz > Bearbeiten > In neuem Fenster bearbeiten	33202
Start > Datensatz > Löschen	32774
Start > Datensatz > Löschen > Alle löschen	32896
Start > Datensatz > Verweis > Teilen	32864
Start > Datensatz > Verweis > Speichern als	32865
Start > Datensatz > Verweis > In Zwischenablage	32866
Start > Datensatz > Verweis > Zu Favoriten hinzufügen	33108

Start > Datensatz > Erweitert > Datensätze zusammenführen	33138
Start > Datensatz > Erweitert > Datensatz überwachen	33252
Start > Suchen > Schnellsuche	32949
Start > Suchen > Suchen	32810
Start > Suchen > Spezialsuche	33177
Start > Suchen > In Verzeichnissen > Datensatz in Onlineverzeichnissen und Social Media suchen	33254
Start > Suchen > In Verzeichnissen > Datensatz in Telefonverzeichnis suchen	32816
Start > Suchen > In Verzeichnissen > Manuelle Übernahme aus einem Telefonverzeichnis	32817
Start > Suchen > Ersetzen	32821
Start > Filtern > Filter erstellen	32829
Start > Filtern > Filter erstellen > Filter zufällig aufteilen	33193
Start > Filtern > Letzten Filter aktivieren	32823
Start > Aktionen > Anrufen	33182
Start > Aktionen > E-Mail senden	33183
Start > Aktionen > Scripte & Workflows > Script ausführen	32934
Start > Aktionen > Scripte & Workflows > Workflow ausführen	33174
Start > Aktionen > Scripte & Workflows > 01. Programm / Script / Workflow	32951
Start > Aktionen > Scripte & Workflows > 02. Programm / Script / Workflow	32952
Start > Aktionen > Scripte & Workflows > 03. Programm / Script / Workflow	32953
Start > Aktionen > Scripte & Workflows > 04. Programm / Script / Workflow	32954
Start > Aktionen > Scripte & Workflows > 05. Programm / Script / Workflow	32955
Start > Aktionen > Scripte & Workflows > 06. Programm / Script / Workflow	32956
Start > Aktionen > Scripte & Workflows > 07. Programm / Script / Workflow	32957
Start > Aktionen > Scripte & Workflows > 08. Programm / Script / Workflow	32958
Start > Aktionen > Scripte & Workflows > 09. Programm / Script / Workflow	32959
Start > Aktionen > Scripte & Workflows > 10. Programm / Script / Workflow	32960
Start > Aktionen > Scripte & Workflows > 11. Programm / Script / Workflow	32961
Start > Aktionen > Scripte & Workflows > 12. Programm / Script / Workflow	32962
Start > Aktionen > Scripte & Workflows > 13. Programm / Script / Workflow	32963
Start > Aktionen > Scripte & Workflows > 14. Programm / Script / Workflow	32964
Start > Aktionen > Scripte & Workflows > 15. Programm / Script / Workflow	32965
Start > Aktionen > Scripte & Workflows > 16. Programm / Script / Workflow	32966
Start > Aktionen > Scripte & Workflows > 17. Programm / Script / Workflow	32967
Start > Aktionen > Scripte & Workflows > 18. Programm / Script / Workflow	32968
Start > Aktionen > Scripte & Workflows > 19. Programm / Script / Workflow	32969
Start > Aktionen > Scripte & Workflows > 20. Programm / Script / Workflow	32970
Start > Aktionen > Scripte & Workflows > 21. Programm / Script / Workflow	32971
Start > Aktionen > Scripte & Workflows > 22. Programm / Script / Workflow	32972
Start > Aktionen > Scripte & Workflows > 23. Programm / Script / Workflow	32973
Start > Aktionen > Scripte & Workflows > 24. Programm / Script / Workflow	32974
Start > Aktionen > Scripte & Workflows > 25. Programm / Script / Workflow	32975
Start > Aktionen > Scripte & Workflows > 26. Programm / Script / Workflow	32976
Start > Aktionen > Scripte & Workflows > 27. Programm / Script / Workflow	32977
Start > Aktionen > Scripte & Workflows > 28. Programm / Script / Workflow	32978
Start > Aktionen > Scripte & Workflows > 29. Programm / Script / Workflow	32979
Start > Aktionen > Scripte & Workflows > 30. Programm / Script / Workflow	32980
Start > Aktionen > Routenplanung > Aktuellen Datensatz als Startpunkt setzen	33056
Start > Aktionen > Routenplanung > Aktuellen Datensatz als Zwischenstopp setzen	33057
Start > Aktionen > Routenplanung > Aktuellen Datensatz als Endpunkt setzen	33058
Start > Aktionen > Routenplanung > Alle Datensätze im Filter als Wegpunkte setzen	33069
Start > Aktionen > Routenplanung > Alle Wegpunkte löschen	33060
Start > Aktionen > Routenplanung > Routenplanung	33059
Start > Aktionen > Routenplanung > Routenplanung zum aktuellen Datensatz	33061

Filtern > Zusammenstellen > Filter-Assistent	33375
Filtern > Zusammenstellen > Filter Allgemein	32829
Filtern > Zusammenstellen > Filter über Landkarte	33432
Filtern > Zusammenstellen > Letzter Filter	32823
Filtern > Zusammenstellen > Filter rückgängig	32824
Filtern > Zusammenstellen > Formularabfrage	32840
Filtern > Zusammenstellen > Freie SQL-Abfrage	32988
Filtern > Zusammenstellen > Filter invertieren	33140
Filtern > Zusammenstellen > Filterausdruck ausführen	32887
Filtern > Zusammenstellen > Volltextrecherche	32832
Filtern > Zusammenstellen > Dublettenprüfung	32831
Filtern > Manuell > Zusammenstellen	33089
Filtern > Manuell > Anwenden	33114
Filtern > Manuell > Datensatz einbeziehen	32892
Filtern > Manuell > Datensatz ausschließen	32893
Filtern > Manuell > Auswahl aufheben	33098
Filtern > Manuell > Alle einbeziehen	32894
Filtern > Manuell > Alle ausschließen	32895
Filtern > Manuell > Datensatzauswahl > Manuellen Filter laden	33141
Filtern > Manuell > Datensatzauswahl > Manuellen Filter speichern	33143
Filtern > Manuell > Datensatzauswahl > Aktuellen Filter als manuellen Filter speichern	33178
Ausgeben > Aktueller Datensatz > E-Mail senden	32811
Ausgeben > Aktueller Datensatz > Brief drucken	32925
Ausgeben > Aktueller Datensatz > Übernahme in Textverarbeitung	32945
Ausgeben > Aktueller Datensatz > Drucken	32230
Ausgeben > Aktueller Datensatz > Drucken > Einzelnes Etikett drucken	32792
Ausgeben > Aktueller Datensatz > Drucken > Einzelne Karteikarte drucken	32793
Ausgeben > Aktueller Datensatz > Drucken > Liste (Bericht) zum Datensatz drucken	32920
Ausgeben > Alle Datensätze > Serien-E-Mail senden	32812
Ausgeben > Alle Datensätze > Serienbriefe drucken	32923
Ausgeben > Alle Datensätze > Serienbrief per Textverarbeitung	32930
Ausgeben > Alle Datensätze > Seriendruck	32231
Ausgeben > Alle Datensätze > Seriendruck > Etiketten drucken	32794
Ausgeben > Alle Datensätze > Seriendruck > Karteikarten drucken	32795
Ausgeben > Alle Datensätze > Seriendruck > Listen/Berichte drucken	32796
Ausgeben > Auswertungen > Sofortbericht	33209
Ausgeben > Auswertungen > Statistik	32906
Ausgeben > Auswertungen > Als Landkarte	33432
Termine/Aufgaben > Neu > Termin zum Datensatz	32833
Termine/Aufgaben > Neu > Aufgabe zum Datensatz	32834
Termine/Aufgaben > Anzeigen > Termine zum Datensatz	32931
Termine/Aufgaben > Anzeigen > Aufgaben zum Datensatz	32932
Termine/Aufgaben > Anzeigen > Terminübersicht	32835
Termine/Aufgaben > Anzeigen > Aufgabenübersicht	32836
Termine/Aufgaben > Wechseln zu > Termine und Aufgaben	32848
Daten > Externe Daten > Importieren	32820
Daten > Externe Daten > Importieren mit Abgleich	32943
Daten > Externe Daten > Exportieren	32830
Daten > Bearbeiten > Relational ergänzen > Allgemein	32900
Daten > Bearbeiten > Relational ergänzen > Vorlage ausführen	33134
Daten > Bearbeiten > Wechseln	32814
Daten > Bearbeiten > Datenanreicherung	32886
Daten > Übernehmen > Filter in den phone manager übernehmen	32898

Daten > Übernehmen > Aktuell gefilterte Datensätze teilen	33179
Konfigurieren > Ansicht > Eigenschaften	33156
Konfigurieren > Ansicht > Eingabemaske	32785
Konfigurieren > Ansicht > Eingabemaske > Eingabemaske WebAccess	33208
Konfigurieren > Verwalten > Sortierungen	32825
Konfigurieren > Verwalten > Schnellsuchfelder	33255
Konfigurieren > Verwalten > Filterausdrücke	32888
Konfigurieren > Verwalten > Sofortberichte	33207
Konfigurieren > Verwalten > Scripte > Ausführen	32934
Konfigurieren > Verwalten > Scripte > Bearbeiten	32935
Konfigurieren > Verwalten > Scripte > Neu	32936
Konfigurieren > Verwalten > Scripte > Verschlüsseln	33125
Konfigurieren > Verwalten > Scripte > Komprimieren	33425
Konfigurieren > Verwalten > Workflows > Ausführen	33174
Konfigurieren > Verwalten > Workflows > Bearbeiten	33175
Konfigurieren > Verwalten > Workflows > Neu	33176
Konfigurieren > Verwalten > Verzeichnisse+Routenplaner	32818
Konfigurieren > Vorlagen > E-Mails	32813
Konfigurieren > Vorlagen > Etiketten	32917
Konfigurieren > Vorlagen > Karteikarten/Briefe	32918
Konfigurieren > Vorlagen > Listen/Berichte	32919
Konfigurieren > Vorlagen > Übernahmemasken > Auswählen	32924
Konfigurieren > Vorlagen > Übernahmemasken > Bearbeiten	32863
Konfigurieren > Vorlagen > Relationales Ergänzen	33136
Gehe zu > Erster Datensatz	32787
Gehe zu > Vorheriger Datensatz	32788
Gehe zu > Nächster Datensatz	32789
Gehe zu > Letzter Datensatz	32790
Statusleiste > Erster Datensatz	33354
Statusleiste > Vorheriger Datensatz	33355
Statusleiste > Nächster Datensatz	33356
Statusleiste > Letzter Datensatz	33357

Relationen-Container in der Eingabemaske	**ID**
Doppelklick oder Eingabetaste auf einen Container-Datensatz	33577

Kontextabhängige Registerkarten	**ID**
Bearbeiten > Änderungen > Speichern	32806
Bearbeiten > Änderungen > Speichern & Schließen	33163
Bearbeiten > Änderungen > Abbrechen	33150
Bearbeiten > Zwischenablage > Einfügen	57637
Bearbeiten > Zwischenablage > Ausschneiden	57635
Bearbeiten > Zwischenablage > Kopieren	57634
Bearbeiten > Zwischenablage > Rückgängig	57643

Bericht > Navigieren > Vorherige Seite	33218
Bericht > Navigieren > Nächste Seite	33220
Bericht > Navigieren > Einen Bericht vor	33216
Bericht > Navigieren > Einen Bericht zurück	33215
Bericht > Zoom > Ganze Seite anzeigen	33226
Bericht > Ausgeben > Drucken	33227
Bericht > Ausgeben > Als PDF exportieren	33228
Bericht > Ausgeben > Nach Excel exportieren	33229

Bericht > Ausgeben > Nach Word exportieren	33368
Bericht > Ausgeben > Als Datei speichern	33230
Bericht > Ausgeben > Per E-Mail versenden	33231
Bericht > Schließen	33244
Manuell Filtern > Datensatzauswahl > Zusammenstellen	33089
Manuell Filtern > Datensatzauswahl > Anwenden	33114
Manuell Filtern > Datensatzauswahl > Datensatz einbeziehen	32892
Manuell Filtern > Datensatzauswahl > Datensatz ausschließen	32893
Manuell Filtern > Datensatzauswahl > Alle einbeziehen	32894
Manuell Filtern > Datensatzauswahl > Alle ausschließen	32895
Manuell Filtern > Datensatzauswahl > Auswahl aufheben	33098
Manuell Filtern > Datensatzauswahl > Datensatzauswahl > Manuellen Filter laden	33141
Manuell Filtern > Datensatzauswahl > Datensatzauswahl > Manuellen Filter speichern	33143
Manuell Filtern > Datensatzauswahl > Datensatzauswahl > Aktuellen Filter als manuellen Filter speichern	33178

Landkarte > Ansicht > Pins	33392
Landkarte > Ansicht > Heatmap	33393
Landkarte > Screenshot > Zwischenablage	33511
Landkarte > Screenshot > Datei	33512
Landkarte > Schließen	32238

13.1.3 Termine & Aufgaben Menü

Befehl in Menüband	**ID**
Start > Neu > Termin	33001
Start > Neu > Aufgabe	33002
Start > Gehe zu > Datensatz	33014
Start > Gehe zu > Heute	33011
Start > Filtern > Benutzer	33027
Start > Filtern > Termine (Erweitert)	33043
Start > Filtern > Aufgaben (Erweitert)	33044
Start > Ansicht > Tag	33006
Start > Ansicht > Arbeitswoche	33005
Start > Ansicht > Woche	33085
Start > Ansicht > Monat	33004
Start > Ansicht > Terminliste	33009
Start > Ansicht > Aufgaben	33003
Start > Daten > Importieren	33191
Start > Daten > Exportieren	33192
Start > Daten > Löschen > Alle Termine löschen	33046
Start > Daten > Löschen > Alle Aufgaben löschen	33047
Start > Daten > Löschen > Termin / Aufgabe löschen	33008
Ausgeben > Ausgeben > Listen > Termine	32997
Ausgeben > Ausgeben > Listen > Aufgaben	32998
Ausgeben > Ausgeben > Karteikarten > Termine	32999
Ausgeben > Ausgeben > Karteikarten > Aufgaben	33000
Ausgeben > Ausgeben > Kalender	33195
Konfigurieren > Einstellungen > Optionen	33023
Konfigurieren > Einstellungen > Ressourcen	33018
Konfigurieren > Einstellungen > Kategorien	33250
Konfigurieren > Einstellungen > Sortierungen > Termine	33029
Konfigurieren > Einstellungen > Sortierungen > Aufgaben	33030
Konfigurieren > Druckvorlagen > Listen > Termine	33019
Konfigurieren > Druckvorlagen > Listen > Aufgaben	33020

Konfigurieren > Druckvorlagen > Karteikarten > Termine	33021
Konfigurieren > Druckvorlagen > Karteikarten > Aufgaben	33022
Konfigurieren > Druckvorlagen > Kalender	33199
Termin > Bearbeiten > Bearbeiten	33010
Termin > Bearbeiten > Delegieren	33012
Termin > Bearbeiten > Löschen	33008
Termin > Aktion > Gehe zu Datensatz	33014
Termin > Aktion > Ausgeben	33015

13.1.4 phone manager Menü

Befehl in Menüband	**ID**
Datei > Hilfe	32784
Datei > Optionen > Allgemein > Wahlhilfe	32773
Datei > Optionen > Allgemein > Telefonanlage	32791
Datei > Optionen > Allgemein > Wahlhilfeeinstellungen	32789
Datei > Optionen > Allgemein > Anruferkennung	32790
Datei > Optionen > Allgemein > Ausgabe Anrufliste	101
Datei > Optionen > Allgemein > Anrufsimulation	32787
Datei > Optionen > Akustische Signale	32795
Datei > Beenden	32799
Start > Wählen > Anrufen	32774
Start > Wählen > Anhalten	32775
Start > Wählen > Nächster	32776
Start > Wählen > Manuell	32779
Start > Wählen > Powerdialing	32794
Start > Bearbeiten > Eintrag verschieben	32778
Start > Bearbeiten > Eintrag löschen	32777
Start > Bearbeiten > Eintrag löschen > Alle Einträge löschen	32782
Start > Bearbeiten > Zur Anwendung wechseln	32781
Start > Bearbeiten > Protokoll im Datensatz hinzufügen	32800
Start > Ausgeben > Anrufliste	106

13.2 Kontextmenüs

Kontextmenü - Container-Datensatz	**ID**
Neu	32845
Neues Dokument > Über Dokumentenverwaltung erzeugen	32903
Neues Dokument > Bestehende Datei	32904
Neues Dokument > Von Scanner einlesen	33028
Bearbeiten	32846
In neuem Fenster bearbeiten	33367
Löschen	32847
Feld in Zwischenablage kopieren	32858
Dokument mit verknüpfter Anwendung öffnen (Eingebettete Datei)	33034
Dokument mit verknüpfter Anwendung bearbeiten (Eingebettete Datei)	33033
Grafik mit verknüpfter Anwendung öffnen	33386
Grafik mit verknüpfter Anwendung bearbeiten	33385
Dokument mit verknüpfter Anwendung öffnen	32861
Dokumentenpfad in Zwischenablage kopieren	33167
Dokumentenverzeichnis öffnen	33169
Dokumentenverzeichnis in Zwischenablage kopieren	33168
Container filtern	33529
Container aktualisieren	33407

Telefonwahl	32849
Aufnehmen in phone manager	32850
Telefontermin planen	32852
Internetbrowser aufrufen	32854
E-Mail senden	32856
E-Mail-Termin planen	32855

Kontextmenü - Telefon-Feld	**ID**
Telefonwahl	32849
Aufnehmen in phone manager	32850
Telefontermin planen	32852

Kontextmenü - Mobiltelefon-Feld	**ID**
Telefonwahl	32849
Aufnehmen in phone manager	32850
Telefontermin planen	32852

Kontextmenü - Internet-Feld	**ID**
Internetbrowser aufrufen	32854

Kontextmenü - Mail-Feld	**ID**
E-Mail senden	32856
E-Mail-Termin planen	32855

Kontextmenü - Adress-Feld	**ID**
Adresseingabeassistent	41037

Kontextmenü - Feld-Clipboard	**ID**
Suchen	32947
Ausschneiden	32857
Kopieren	32858
Einfügen	32859
Löschen	32860
Programmaufruf	33051
Häufigkeitsstatistik über Feld	33115

Kontextmenü - Eingebettete Datei-Feld	**ID**
Dokument einfügen	33032
Dokument mit verknüpfter Anwendung öffnen	33034
Dokument mit verknüpfter Anwendung bearbeiten	33033
Dokument löschen	33035

Kontextmenü - Grafik-Feld	**ID**
Grafik auswählen	32929
Grafik mit verknüpfter Anwendung öffnen	32928
Grafik mit verknüpfter Anwendung bearbeiten	33385
Grafik entfernen	33258

Kontextmenü - Combobox-Feld	**ID**
Eintrag zur Auswahl hinzufügen	32944

Kontextmenü - Code-Feld	**ID**
Häufigkeitsstatistik über Feld	33115

Code-Bezeichnungen bearbeiten	33092

Kontextmenü - Titelzeile von Übersichtsliste/Container	**ID**
Konfigurieren > Layout und Spalten	5012
Konfigurieren > Spalteneigenschaften	5011
Konfigurieren > Profile	33536-33544
Konfigurieren > Profile > Als Profil speichern	33095
Konfigurieren > Profile > Profile verwalten	33094
Konfigurieren > Als Projektvoreinstellung speichern	32984
Konfigurieren > Als Projektvoreinstellung verteilen	33839
Konfigurieren > Auf Projektvoreinstellung zurücksetzen	32985
Sortieren > Aufsteigend	32868
Sortieren > Absteigend	32869
Sortieren > < ohne Sortierung >	32885
Sortieren > 1	32870
Sortieren > 2	32871
Sortieren > 3	32872
Sortieren > 4	32873
Sortieren > 5	32874
Sortieren > 6	32875
Sortieren > 7	32876
Sortieren > 8	32877
Sortieren > 9	32878
Sortieren > 10	32879
Sortieren > 11	32880
Sortieren > 12	32881
Sortieren > 13	32882
Sortieren > 14	32883
Sortieren > 15	32884
Sortieren > Nicht gruppieren	33348
Sortieren > Einfach gruppieren	33349
Sortieren > Mehrfach gruppieren	33350
Filtern	33529
Als Liste/Bericht ausgeben	5020
Daten exportieren	33390
Aktualisieren	33407

Kontextmenü - RTF-Feld	**ID**
Rückgängig	32909
Ausschneiden	32910
Kopieren	32911
Einfügen	32912
Löschen	32913
Alles markieren	32908

Kontextmenü - Titelzeile von Terminliste/Aufgaben in Termine & Aufgaben	**ID**
Konfigurieren > Layout und Spalten	5012
Sortieren > Aufsteigend	32868
Sortieren > Absteigend	32869
Sortieren > < ohne Sortierung >	32885
Sortieren > 1	32870
Sortieren > 2	32871
Sortieren > 3	32872

Sortieren > 4	32873
Sortieren > 5	32874
Sortieren > 6	32875
Sortieren > 7	32876
Sortieren > 8	32877
Sortieren > 9	32878
Sortieren > 10	32879
Sortieren > 11	32880
Sortieren > 12	32881
Sortieren > 13	32882
Sortieren > 14	32883
Sortieren > 15	32884
Sortieren > Nicht gruppieren	33348
Sortieren > Einfach gruppieren	33349
Sortieren > Mehrfach gruppieren	33350
Filtern	33529
Aktualisieren	33407

Kontextmenü - Datei-Feld	**ID**
Dokument auswählen	32862
Dokument mit verknüpfter Anwendung öffnen	32861
Dokumentenpfad in Zwischenablage kopieren	33167
Dokumentenverzeichnis öffnen	33169
Dokumentenverzeichnis in Zwischenablage kopieren	33168

Kontextmenü - Eingebettete Grafik-Feld	**ID**
Grafik einfügen	33032
Grafik mit verknüpfter Anwendung öffnen	33034
Grafik mit verknüpfter Anwendung bearbeiten	33033
Grafik löschen	33035

Kontextmenü - DMS-Feld	**ID**
Dokument in DMS einfügen	33032
DMS-Dokument mit verknüpfter Anwendung öffnen	33034
DMS-Dokument mit verknüpfter Anwendung bearbeiten	33033
Dokument im DMS löschen	33035
Verweis auf DMS-Dokument lösen	33045

Kontextmenü - Zusammenführen	**ID**
Kopieren	32858

Kontextmenü - Überwachungsergebnisse	**ID**
Gehe zu Datensatz	33121
Eintrag löschen	33247
Alle Einträge zu diesem Datensatz löschen	33248
Alle Einträge löschen	33249

14 Änderungen und Neuerungen

14.1 Version 12

14.1.1 Grundlegende Änderungen

Version 12.003

- C# Scripting: Es besteht die Möglichkeit zum Programmstart ein spezielles Script ausführen zu lassen, um große Referenzen (mit `<!--#include ref="[Pfad einer .dll]"-->`) dauerhaft zu laden. Weitere Informationen erhalten Sie im Kapitel Hinweise zur Benutzung von C#-Scripten.

Version 12.000

- Wichtig! Der Anwendungstitel wurde geändert von combit Relationship Manager auf combit CRM. Etwaige Makros, die den Fenstertitel verwenden, müssen angepasst werden.
- Die Methoden RecordSet.DialogSelectRecord, RecordSet.DialogSelectRecordMultiple, RecordSet.SendBulkMail (wenn Mail-Editor angezeigt werden soll) und InputForm.DialogSelectRecordDropDown erfordern ein fully-dynamic RecordSet, welches durch den neuen optionalen Parameter CursorModel für folgende Methoden definiert wird: ViewConfig.CreateRecordSet, View.CurrentRecordSetCopy, Container.CurrentRecordSetCopy, RecordSet.CreateCopy, Record.GetRelationalRecordSet.
- C# Scripting: View.CurrentRecordSetCopy und Container.CurrentRecordSetCopy sind nun Methoden und keine Eigenschaften mehr. Es ist daher eine Scriptanpassung notwendig (von View/Container.CurrentRecordSetCopy auf View/Container.CurrentRecordSetCopy()).

14.1.2 Eigenschaften und Methoden

14.1.2.1 Eigenschaften

Version 12.000

- Event im WScript Objekt ist nun global verfügbar

14.1.2.2 Methoden

Version 12.000

- CalcAggregationValues im RecordSet Objekt
- DialogSelectRecordDropDown im InputForm Objekt
- GetHwndByName im InputForm Objekt
- FldIndex im ViewConfig Objekt
- FldCaseSensitive im ViewConfig Objekt
- CreatePKCEVerifierAndChallenge im Application/cRM Objekt
- DialogAddressAssistant im Application/cRM Objekt
- InvokeDataContextMenu im Container Objekt
- InvokeTitleContextMenu im Container Objekt

- Neuer Optionaler Parameter CheckForModifiedFields in der Methode Record.Lock. Der neue Standardwert dieses optionalen Parameters (False) entspricht einer Verhaltensänderung.
- Verhaltensänderung für die Methode SetContentsByNameFromFile im Record-Objekt (Löschen und Entfernen der einzubettenden Datei kann erst nach dem Speichern (Record.Save) durchgeführt werden.
- Neuer optionaler Parameter CursorModel für folgende Methoden: ViewConfig.CreateRecordSet, View.CurrentRecordSetCopy, Container.CurrentRecordSetCopy, RecordSet.CreateCopy, Record.GetRelationalRecordSet

14.1.3 Ereignisse

Version 12.000

- CRM_WebViewClose für Microsoft Chromium Edge Runtime basierte Scripte.
- Neues Ereignis: Datensatzbearbeitung wird begonnen
- Neues Ereignis: Datensatzbearbeitung wurde begonnen.

14.2 Version 11

14.2.1 Eigenschaften und Methoden

14.2.1.1 Eigenschaften

Version 11.003

- 'Event' im WScript Objekt ist nun global verfügbar

Version 11.002

- 'DefaultISOCountry' im AddressInfo Objekt

Version 11.000

- 'CascadeOnDelete' im Relation Objekt
- 'SupportsRecycleBin' im ViewConfig Objekt

14.2.1.2 Methoden

Version 11.006

- 'CreateCopy' im RecordSet Objekt
- 'PerformanceCounterCreate' im Application Objekt
- 'PerformanceCounterHit' im Application Objekt
- Optionaler Parameter 'ShowDialog' für 'SendBulkMail' im RecordSet Objekt

Version 11.005

- 'SetOption' im EmailTool Objekt

- Optionaler Parameter 'Files' für 'SendBulkMail' im RecordSet Objekt

Version 11.003

- 'GetRecipientStatus' im EmailToolRecipientList Objekt
- Optionaler Parameter 'MailsSentToTagFilePath' für 'SendBulkMail' im RecordSet Objekt

Version 11.002

- 'ChangeEmailAddress' im EmailToolRecipientList Objekt
- 'CreateGUID' im Application Objekt
- 'GetISOCountryFromUserCountryCode' im AddressInfo Objekt
- 'GetUserCountryCodesFromISOCountry' im AddressInfo Objekt
- 'GetMailingRecipients' im EmailToolMailingResults Objekt

Version 11.001

- 'SetProperty' im EmailToolMailing Objekt erweitert (Brevo)
- 'SetProperty' im EmailToolRecipientList Objekt erweitert (Brevo)

Version 11.000

- 'Update' im Container Objekt
- Erweiterung von 'SetFilterByFieldName' im RecordSet Objekt
- Wichtige Änderung für 'FldTypePhys' im ViewConfig Objekt
- 'HTTPGet' im Application Objekt
- 'HTTPPost' im Application Objekt
- 'HTTPPut' im Application Objekt
- 'HTTPPatch' im Application Objekt
- 'HTTPDelete' im Application Objekt
- 'OAuthRedirectDialog' im Application Objekt

14.2.2 Ereignisse

Version 11.000

- 'Datensatz wird aus Papierkorb wiederhergestellt'
- 'Datensatz wird in Papierkorb endgültig gelöscht'

14.2.3 Informationen zum C# Scripting

14.2.3.1 .NET Framework

Version 11.000

- Es wird nun das .NET Framework 4.8 unterstützt. Weitere Informationen erhalten Sie im Kapitel Hinweise zur Benutzung von C#-Scripten.

14.2.4 Änderung bei Web-Ansicht:

14.2.4.1 Microsoft Chromium Edge Runtimes (WebView2)

Version 11.000

Für die Web-Ansicht können nun die Microsoft Chromium Edge Runtimes verwendet werden. Dies ermöglicht den Zugriff auf das cRM-SDK per JavaScript. Weitere Informationen erhalten Sie im Kapitel Info-Zentrale, Web-Ansichten, Web-Elemente, Web-Panel