



{ewl RoboEx32.dll, WinHelp2000, }

# **SAP Entry**

Preface

Requirements

Prompting the SAP Import

Importing SAP Tables

Building Relationships and TE

Error List Window

Restrictions in SAP Data Models

Restrictions in SAP Data Sources

Appendix: Siron Help Draft

## Preface

Ziel dieses Benutzerhandbuches ist es, dem Zeidon Anwender eine Hilfestellung zu geben, um Zeidon Datenmodelle, inklusive des Technical Environments (TE), für Zugriffe auf SAP effektiv zu erzeugen.

In einigen Textpassagen werden Sie Aussagen zur zeidon-internen Realisierung finden. Nutzen Sie diese bitte, um möglichst performant zu arbeiten.

Weitere Themen dieses Benutzerhandbuches sind u.a. die Besonderheiten von SAP Datenmodellen und SAP Data Sources. Diese Ausführungen sollen zeigen, wo und warum es Unterschiede zu SAP Datenmodellen bzw. SAP Data Sources gibt.

Haben Sie Anregungen und Verbesserungsvorschläge für dieses Benutzerhandbuch, so sind für eine kurze Information sehr dankbar.

### See Also

[Symbols](#)

[Usage of Terms](#)

# Symbols

Um den Umgang mit dem Benutzerhandbuch zu erleichtern, wurden im Text verschiedene Stellen mit Symbolen gekennzeichnet.



Wichtige Informationen werden durch dieses Symbol markiert.



Verweis auf die Online-Hilfe von Siron.

## See Also

[Preface](#)

[Usage of Terms](#)

## Usage of Terms

Kontextmenüs: Aufruf per rechter Maustaste.

Selektieren: Linker Mausklick zur Auswahl eines Eintrages.

STRG-Taste + linker Mausklick für selektive Mehrfach-Auswahl.

Linker Mausklick auf den ersten Eintrag + SHIFT-Taste auf den letzten Eintrag zur  
Selektion aller Einträge, die zwischen dem ersten und letzten Mausklick liegen

### See Also

[Preface](#)

[Symbols](#)

# Requirements

Für den SAP Import müssen folgende Voraussetzungen gegeben sein:

## See Also

[SAP System Requirements](#)

[Siron Requirements](#)

[Zeidon Requirements](#)

## SAP System Requirements

- ☞ Alle SAP Zugriffe basieren auf der Siron-SAP R/3-Schnittstelle. Für diese Schnittstelle wird die SAP R/3-Komponente RFC-SDK benötigt.
- ☞ Für die Kommunikation mittels RFC-Programmen ist im SAP System eine Destination einzurichten, die aus Rechnername, Systemname und Systemnummer zusammengesetzt ist (i.a. Transaktion SM59).
- ☞ Für das Lesen von SAP Tabellen wird die von Siron bereitgestellte ABAP/4 Funktion vorausgesetzt. Implementieren Sie diese bitte in Ihrem SAP R/3 System (vgl. Anhang A).

### See Also

[Siron Requirements](#)

[Zeidon Requirements](#)



## Siron Requirements

1. Da SAP Zugriffe auf der Siron-SAP R/3 Schnittstelle basieren, ist die Existenz der Siron Environment Datei "sirenv.txt" notwendige Voraussetzung. Diese Datei ist durch Setzen der System-Variable "SIRENV" in ihrer Systemumgebung bekannt zu geben:

**Variable:** SIRENV

**Value:** *Laufwerk:\Pfad\sirenv.txt*

Beispiel: SIRENV C:\SironCS\Env\sirenv.txt

2. Innerhalb der sirenv.txt müssen in der <COMM> Sektion alle Server eingetragen werden, mit denen Sie zugreifen möchten.

**<COMM>**

**logischer Servername:** SAPSYS = *Systemname* ,

HOST = *Rechnername*,

SYSNR = *Systemnummer*;

Beispiel: <COMM>

SAP40B: SAPSYS = A01,

HOST = scio\_r3,

SYSNR = 01;

SAP40A: SAPSYS = A01,

HOST = scio\_r3a,

SYSNR = 01;



Beachten Sie: Systemname, Rechnername und Systemnummer müssen mit den Werten im SAP System eingetragenen Destination identisch sein (vgl. SAP System).

3. Des Weiteren sind in der sirenv.txt die ABAP Lese-Funktion (vgl. SAP System) sowie die Version des SAP System selbst zu erfassen. Erweitern Sie dazu die <PARM> Sektion wie folgt.

#### <PARM>

SAPREADFUNK = *Name der im SAP eingetragenen Lesefunktion;*

SAPVERSION = *Version des SAP Systems;*

Beispiel: <PARM>

SAPREADFUNK = YTBZ\_READ\_TABLE;

SAPVERSION = 45B;

#### See Also

[SAP System Requirements](#)

[Zeidon Requirement](#)

## Zeidon Requirements

Sind alle o.g. Voraussetzungen erfüllt, so kann der SAP Import immer dann erfolgen, wenn in Zeidon ein Projekt angelegt wurde, das mindestens über die Zeidon Standard Domains TZAPDMAA, TZAPDMAB und TZAPDMAC verfügt.

### See Also

[SAP System Requirements](#)

[Siron Requirements](#)

# Prompting the SAP Import

Beim SAP Import werden SAP Tabellen in ein Entity Relationship Diagramm überführt. Der Aufruf erfolgt deshalb über das Zeidon Tool "Data Model".

Der Import umfasst das:

## 1. Importieren von SAP-Tabellen

Aufruf: Menü <Utilities>,  
Menüoption <SAP Import>  
Submenüoption <Tables>

Der Import von SAP Tabellen kann generell ohne existierendes Datenmodell erfolgen. Importiert der Benutzer SAP Tabellen und ist noch kein Datenmodell vorhanden, so wird das Datenmodell von Zeidon automatisch erstellt.

Der Import von SAP Tabellen unterstützt zusätzlich auch das Bilden der Relationships und des Technical Environments (TE). Vergleichen Sie hierzu Kapitel 4 "Bilden der Relationships und des TEs".

## 2. Building Relationships and Technical Environment (TE)

Aufruf: Menü <Utilities>,  
Menüoption <SAP Import>  
Submenüoption <Build Relationships and TE>

Das Bilden der Relationships und des Technical Environments (TE) sind als nachgelagerte Stufe zu betrachten. Diese Option sollte immer erst dann aufgerufen werden, wenn wirklich alle benötigten SAP Tabellen in das Datenmodell importiert wurden.



Beachten Sie: Diese Menüoption ist immer dann gesperrt, wenn Zeidon im aktuell geöffneten Datenmodell keine Entität findet, die über den SAP Import erzeugt wurde.

## See Also

[Building Relationships and TE](#)

## Importing SAP Tables

Sind die notwendigen Voraussetzungen für den SAP Import erfüllt, können Sie über das folgende Fenster:

1. Anmeldeparameter für den Zugriff auf SAP definieren.
2. Die Sprache sowie den Importumfang festlegen.
3. Das Verhalten bei Importfehlern definieren.
4. Die zu importierenden Tabellen suchen.
5. Für die gefundenen SAP Tabellen die abhängigen Tabellen anzeigen lassen.
6. SAP Tabellen in ein Zeidon Datenmodell importieren.
7. Relationships importieren und das Technical Environment (TE) generieren lassen.
8. Das Error List Fenster abfragen.

The screenshot shows the 'SAP Import' dialog box with the 'Search' tab selected. The 'Server' field is set to 'SAP40B' and the 'Tables' field is set to 'T00'. A 'Search' button is visible. Below the search fields, a message states '68 Table(s) found'. A table lists the found tables with their names and descriptions. The status bar at the bottom indicates 'Search Tables completed successfully'.

SAP Table Name	Description
T000	Mandanten
T000CM	Mandantenabhängige FI-AR-CR Einstellungen
T000F	Mandantenübergreifende FI-Einstellungen
T000G	Mandantenübergreifende FI-SL-Buchungen
T000GL	Flexibles Hauptbuch: Customizingprüfung und Aktivierung
T000K	Konzern
T000MD	Disposition auf Dispositionsebene

**See Also**

[Logon Parameters for the SAP Access](#)

Definition of Import Range and Language

Behaviour at Import Errors

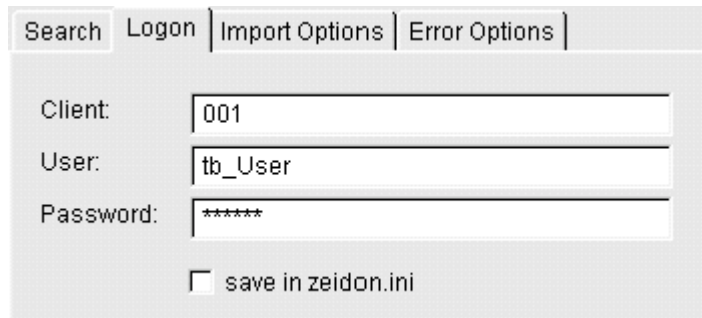
Searching SAP Tables

Searching dependant SAP Tables

Importing SAP Tables into a Zeidon Data Model

## Logon Parameters for the SAP Access

Vor dem ersten SAP Zugriff müssen über das Tab <Logon> Mandant (Client), Passwort und User erfasst werden.



The screenshot shows a dialog box with four tabs: 'Search', 'Logon', 'Import Options', and 'Error Options'. The 'Logon' tab is active. It contains three input fields: 'Client:' with the value '001', 'User:' with the value 'tb\_User', and 'Password:' with the value '\*\*\*\*\*'. Below these fields is a checkbox labeled 'save in zeidon.ini' which is currently unchecked.

Sollen diese Eingaben künftig entfallen, ist die Checkbox <save in zeidon.ini> zu aktivieren.

Immer wenn diese Checkbox aktiviert ist, aktualisiert bzw. erzeugt (falls noch nicht vorhanden) Zeidon nach dem Schließen des Fensters folgende Einträge unter der Gruppe [SAP] :

- DefaultClient
- DefaultUser
- DefaultPassword (wird verschlüsselt gespeichert).

### See Also

[Definition of Import Range and Language](#)

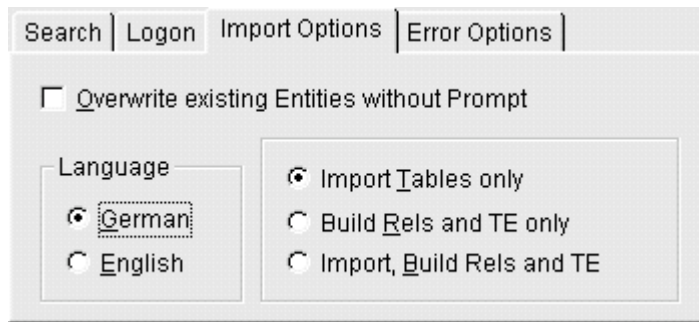
[Behaviour at Import Errors](#)

[Searching SAP Tables](#)

[Searching dependant SAP Tables](#)

[Importing SAP Tables into a Zeidon Data Model](#)

## Definition of Import Range and Language



Search | Logon | **Import Options** | Error Options |

☐ Overwrite existing Entities without Prompt

Language

☒ German  
☐ English

☒ Import Tables only  
☐ Build Rels and TE only  
☐ Import, Build Rels and TE

### See Also

[Logon Parameters for the SAP Access](#)

[Option <Overwrite existing Entities without Prompt>](#)

[Language Definition](#)

[Option <Import Tables only>](#)

[Option <Build Rels and TE Only>](#)

[Option <Import, Build Rels and TE>](#)

[Behaviour at Import Errors](#)

[Searching SAP Tables](#)

[Searching dependant SAP Tables](#)

[Importing SAP Tables into a Zeidon Data Model](#)



## Option <Overwrite existing Entities without Prompt>

Beim Import von SAP Tabellen prüft Zeidon, ob die zu importierenden Tabellen im Datenmodell bereits existieren (vgl. Punkt "3.6.1 Erzeugung der Entitäten"). Ist dies der Fall, muss der Benutzer entscheiden, ob die jeweilige Entität ersetzt werden soll.

Anders verhält sich Zeidon, wenn die Checkbox <Overwrite existing Entities without Prompt> aktiv ist. Hier löscht Zeidon die vorhandenen Entitäten selbständig und erzeugt sie anschließend neu. Der Benutzer kann somit nicht mehr in den Ersetzungsprozess eingreifen.

Implementiert wurde diese Funktion für Benutzer, die eine große Anzahl von Tabellen importieren möchten, ohne dass Benutzereingaben den Import unterbrechen.

### See Also

[Language Definition](#)

[Option <Import Tables only>](#)

[Option <Build Rels and TE Only>](#)

[Option <Import, Build Rels and TE>](#)

## Language Definition

Die Spracheinstellungen beziehen sich nur auf das SAP System. Der Benutzer steuert hierüber, in welcher Sprache Tabellen- und Spaltenbeschreibungen importiert werden.

### See Also

Option <Overwrite existing Entities without Prompt>

Option <Import Tables only>

Option <Build Rels and TE Only>

Option <Import, Build Rels and TE>

## Option <Import Tables only>

Beim Import von SAP Tabellen erzeugt Zeidon nur Entitäten mit Attributen und Identifiers, Relationships werden nicht generiert.

Diese Option sollte immer dann verwendet werden, wenn Tabellen über mehrere Importprozesse ins Datenmodel geladen werden sollen.

### See Also

[Option <Overwrite existing Entities without Prompt>](#)

[Language Definition](#)

[Option <Build Rels and TE Only>](#)

[Option <Import, Build Rels and TE>](#)

## Option <Build Rels and TE Only>

Alle gewünschten SAP Tabellen sollten bereits in das Datenmodell importiert sein. Befindet sich im Datenmodell keine Entität, die über den SAP Import erzeugt wurde, so ist diese Option gesperrt.

Beim Bilden der Relationships werden alle Relationships, die zwischen SAP Entitäten bestehen, gelöscht und anschließend neu erzeugt. Ist dieser Prozess erfolgreich beendet, generiert Zeidon das Technical Environment (TE).

### See Also

[Option <Overwrite existing Entities without Prompt>](#)

[Language Definition](#)

[Option <Import Tables only>](#)

[Option <Import, Build Rels and TE>](#)

## Option <Import, Build Rels and TE>

Die Option <Import, Build Rels and TE> importiert die gewählten SAP Tabellen. Konnte dieser Vorgang fehlerfrei beendet werden, löscht Zeidon alle Relationships, die zwischen SAP Entitäten bestehen und erzeugt diese neu. Erst wenn auch dieser Prozess erfolgreich beendet wurde, generiert Zeidon das Technical Environment (TE).

Da beim Generieren der Relationships immer erst alle bestehenden Relationships gelöscht werden, sollte man aus Performancegründen hiervon nur dann Gebrauch machen, wenn wirklich alle gewünschten Tabellen in einem Schritt importiert werden können. Sind mehrere Importprozesse notwendig, ist die Option <Import Tables only> wesentlich performanter.

### See Also

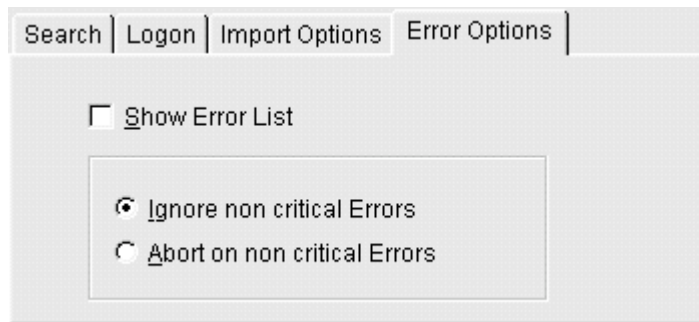
[Option <Overwrite existing Entities without Prompt>](#)

[Language Definition](#)

[Option <Import Tables only>](#)

[Option <Build Rels and TE Only>](#)

## Behaviour at Import Errors



The screenshot shows a software window with four tabs: 'Search', 'Logon', 'Import Options', and 'Error Options'. The 'Error Options' tab is active. Inside this tab, there is a checkbox labeled 'Show Error List' which is currently unchecked. Below this checkbox is a group box containing two radio button options: 'Ignore non critical Errors' (which is selected) and 'Abort on non critical Errors'.

### See Also

[Option <Show Error List>](#)

[Behaviour at SAP Import Errors](#)

[Logon Parameters for the SAP Access](#)

[Definition of Import Range and Language](#)

[Searching SAP Tables](#)

[Searching dependant SAP Tables](#)

[Importing SAP Tables into a Zeidon Data Model](#)

## Option <Show Error List>

Ist die Option <Show Error List> aktiviert, so lädt Zeidon zu Beginn des Imports ein "Error List" Fenster.

Treten während des Imports unkritische Fehler auf, so werden diese sofort im Error Fenster angezeigt.

Ist diese Option deaktiviert und sind unkritische Fehler beim Import aufgetreten, lädt Zeidon die Error List erst nach Abschluss des Imports.

Ist das Error List Fenster geöffnet und ist die Option <Show Error List> deaktiviert und sind keine unkritischen Fehler aufgetreten, schließt Zeidon die Error List automatisch.

### See Also

[Behaviour at SAP Import Errors](#)

## **Behaviour at SAP Import Errors**

I ist, haben unkritische Fehl0 1keinen Einfluss



## Searching SAP Tables

Search | Logon | Import Options | Error Options |

Server: SAP40B

Tables: T00

Table Name delimited by space, semicolon or comma ( Example: T0, SFL or T0; SFL or T0 SFL )

Search

### See Also

[Server Settings](#)

[SAP Logon](#)

[Stating the Search Parameters](#)

[Locating SAP Tables](#)

[Logon Parameters for the SAP Access](#)

[Definition of Import Range and Language](#)

[Behaviour at Import Errors](#)

[Searching dependant SAP Tables](#)

[Importing SAP Tables into a Zeidon Data Model](#)

## Server Settings

In der Combobox <Server> zeigt Zeidon alle SAP Server, die in der sirenv.txt definiert wurden (vgl. Kapitel 1 "Voraussetzungen"). Wählen Sie hier bitte den Server, über den Sie auf SAP zugreifen möchten. Ist der gewünschte Server nicht in der Combobox enthalten, erfassen Sie diesen in der Datei sirenv.txt.

### See Also

[SAP Logon](#)

[Stating the Search Parameters](#)

[Locating SAP Tables](#)

# SAP Logon

Eine SAP Anmeldung erfolgt bei Betätigung des <Search> Buttons und ist aktiv bis entweder:

- ein anderes Projekt geladen (Switch Project),
- das Tool "Data Model" geschlossen oder
- vor einer erneuten Suche Client, User, Password oder der Server geändert wird.

Beim Ändern der SAP Anmeldeparameter fährt Zeidon automatisch ein SAP Logoff und anschließend das gewünschte SAP Logon.

Wird beim Logon festgestellt, dass diese Anmeldeparameter bereits schon einmal erfasst wurden, lädt Zeidon alle unter dieser SAP Anmeldung gesuchten Tabellen und zeigt das Suchergebnis **ohne erneuten SAP Zugriff** an.

- Beispiel:
1. Sie suchen auf dem User ADMIN die Tabelle T000.
    - ⇒ SAP Logon
    - ⇒ Tabelle wird von SAP gelesen
  2. Nun wechseln Sie den User auf TEST und suchen die Tabelle SPFLI und importieren diese.
    - ⇒ SAP Logoff
    - ⇒ neues SAP Logon
    - ⇒ Tabelle wird von SAP gelesen
  3. Da Sie vergessen haben, die Tabelle T000 zu importieren, kehren Sie zum User ADMIN zurück und suchen wieder die Tabelle T000
    - ⇒ SAP Logoff
    - ⇒ SAP Logon
    - ⇒ Suchstring wird für diese SAP Anmeldung gefunden, Tabelle wird aus zeidon-internen Work-Objekt gelesen, es erfolgt kein weiterer SAP Zugriff

## See Also

[Server Settings](#)

[Stating the Search Parameters](#)

[Locating SAP Tables](#)

## Stating the Search Parameters

Die Combobox <Tables> dient zur Eingabe der zu suchenden SAP Tabellen. Sie zeigt alle gesuchten Tabellen für die gerade aktive SAP Anmeldung.

Für die Eingabe der zu suchenden Tabellen gelten folgende Regelung:

- Die zu suchenden SAP Tabellen müssen durch mindestens 1 Leerzeichen, Semikolon oder Komma getrennt werden. Dabei ist es uninteressant, welches Trennzeichen Sie verwenden.
- Die Eingabe der Tabellennamen kann in Klein- und/oder Großschreibung erfolgen.
- Wildcards (z.B. \* oder %) sind nicht statthaft, da diese stets als Tabellennamen interpretiert werden.
- Sollen z.B. alle Tabellen gesucht werden, die mit T00 beginnen, so erfassen Sie einfach T00. Zeidon sucht stets nach den Tabellen, die mit den eingegebenen Tabellennamen beginnen. Dieses Verhalten können Sie nicht beeinflussen.
- Tabellennamen können in einer Suche doppelt vorkommen. Im Suchergebnis werden Sie diese Tabellen nur einmal finden.
- Maximale Eingabelänge für die zu suchenden Tabellen beträgt 254 Zeichen.

Beispiele für gültige Such-Eingaben:

- t00, spfli,  
t001
- T00;  
SPFLI;  
T001
- t00 Spfli  
T001
- t00,; spfli  
t001
- ,; t00 spfli  
, t001 ,;

### See Also

[Server Settings](#)

[SAP Logon](#)

[Locating SAP Tables](#)

## Locating SAP Tables

Bei Aktivierung des <Search> Buttons prüft Zeidon, ob die SAP Anmeldeparameter (Client, User, Passwort, Server) sowie die zu suchenden Tabellen erfasst sind.

Ist das der Fall, wird der String für die zu suchenden Tabellen wie folgt aufbereitet:

- Führende und nachfolgende Trennzeichen werden entfernt.
- Alle Tabellennamen werden in Großbuchstaben umgewandelt und durch Semikolon getrennt.

Ergibt die Aufbereitung einen gültigen Suchstring, erfolgt die SAP Anmeldung und das Suchen/Laden der SAP Tabellen, falls notwendig (vgl. Punkt "3.4.2 SAP Anmeldung").

Als Suchergebnis zeigt Zeidon die Anzahl der gefundenen Tabellen sowie die Tabellen selbst und deren Beschreibungen.

Die Tabellennamen liest Zeidon stets aus der SAP Tabelle DD02L. Die Beschreibungen werden aus DD02T geladen.



Beachten Sie: Das Suchergebnis zeigt stets nur Tabellen vom Typ TRANSP, POOL

und VIEW. Alle anderen Tabellentypen sind für Zeidon uninteressant und werden ignoriert.

### See Also

[Server Settings](#)

[SAP Logon](#)

[Stating the Search Parameters](#)

## Searching Related SAP Tables

Für den Import von SAP Tabellen ist es interessant zu wissen, ob die zu importierende Tabelle zu anderen Tabellen in Beziehung steht.

Aus diesem Grund wurde in der Liste der gefundenen SAP Tabellen das folgende Kontextmenü integriert:



### See Also

[Option <Show Related Tables> and <Show Related for all Tables>](#)

[Option <Remove Related Tables> and <Remove Related for all Tables>](#)

[Option <Remove Tables from List>](#)

[Logon Parameters for the SAP Access](#)

[Definition of Import Range and Language](#)

[Behaviour at Import Errors](#)

[Searching SAP Tables](#)

[Importing SAP Tables into a Zeidon Data Model](#)

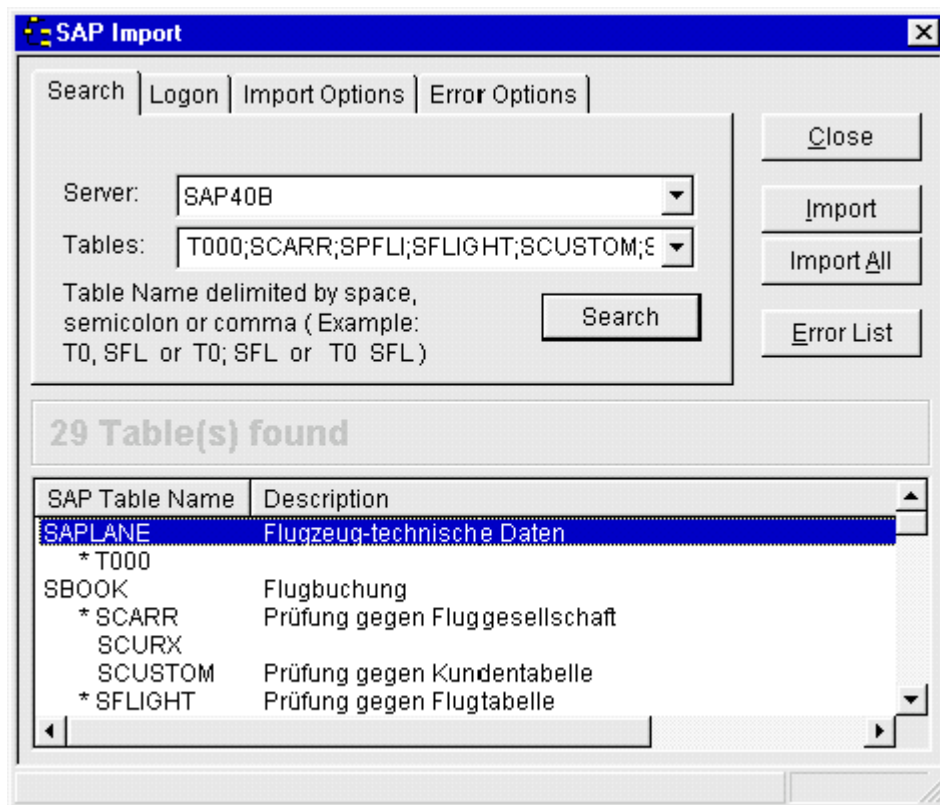
## Option <Show Related Tables> and <Show Related for all Tables>

Die Kontext Menüoption <Show Related Tables> bezieht sich nur auf die selektierten SAP Tabellen, die Option <Show Related for all Tables> hingegen sucht für alle angezeigten Tabellen diejenigen, zu denen eine Beziehung besteht.

Bei der Suche nach abhängigen Tabellen werden Foreign Key Informationen von SAP Tabelle DD08V gelesen.

Aus diesen Foreign Keys werden später Relationships zu den abhängigen Tabellen generiert.

Ist der Foreign Key Bestandteil des Keys, so sprechen wir von einer identifizierenden Relationship. Die dazu gehörende Tabelle wird im Import Fenster durch einen Stern gekennzeichnet (vgl. Abbildung unten).



Die Existenz der gesternten Tabellen und somit der identifizierenden Relationships sind notwendige Voraussetzung für die erfolgreiche Generierung des Technical Environments (TE).

Konkret bedeutet dies, auch wenn eine gesternzte Tabelle nicht importiert wurde, so wird diese beim Bilden der Relationships automatisch in das Datenmodell geladen (vgl. Punkt "4.3 Erzeugung der Relationships").



Beachten Sie:

Beim Lesen der abhängigen Tabellen werden die Anmeldeparameter nicht noch einmal geprüft. Wechseln Sie z.B. nach erfolgreicher Suche von SAP

Tabellen auf einen anderen User und starten anschließend die Suche nach abhängigen Tabellen, so sucht Zeidon stets mit dem User, für den das angezeigte Suchergebnis ursprünglich gefunden wurde.

Die Suche von abhängigen Tabellen bezieht sich jeweils nur auf Tabellen vom Typ "TRANSP", "POOL" und "VIEW". Allen anderen Tabellentypen sind für Zeidon uninteressant und werden ignoriert.

Sind abhängige Tabellen im Import Fenster sichtbar, so werden diese bei Aktivierung des Buttons <Import All> mit importiert. Ähnlich verhält es sich, wenn der Button <Import> betätigt wird und abhängige Tabellen selektiert sind (vgl. Punkt "3.6 SAP Tabellen in ein Zeidon Datenmodell importieren").

### Zugriffsoptimierung:

Um die Anzahl der SAP Zugriffe auf ein Minimum zu reduzieren, speichert Zeidon alle gelesenen abhängigen Tabellen in einem Work-Objekt. D.h., bei erneuten Aufruf dieser Menüoptionen werden die abhängigen Tabellen aus dem zeidon-internen Work-Objekt und nicht von SAP gelesen.

Beispiel: Das Suchergebnis beinhaltet die SAP Tabellen SAPLANE und SBOOK:

1. Für die Tabelle SAPLANE rufen Sie die Menüoption <Show Related Tables> auf.
  - ⇒ SAP Tabelle SAPLANE wird nicht im Work-Object gefunden, Suche der abhängigen Tabellen im SAP
2. Sie rufen nun die Menüoption <Show Related for all Tables> auf.
  - ⇒ SAP Tabelle SAPLANE wird im Work-Object gefunden und die abhängigen Tabellen werden vom Work-Object geladen, es erfolgt kein SAP Zugriff für diese Tabelle
  - ⇒ SAP Tabelle SBOOK wird nicht im Work-Object gefunden, Suche der abhängigen Tabellen im SAP

### **See Also**

Option <Remove Related Tables> and <Remove Related for all Tables>

Option <Remove Tables from List>



## Option <Remove Related Tables> and <Remove Related for all Tables>

Bei der Kontext Menüoption <Remove Related for all Tables> werden bei allen SAP Tabellen der Stufe 1 die abhängigen Tabellen ausgeblendet.

Das Verhalten der Kontext Menüoption <Remove Related Tables> ist abhängig von der Stufe der selektierten Tabelle. Gehört sie zur der Stufe 1, so blendet Zeidon die zugeordneten abhängigen Tabellen (Stufe 2) aus. Sind nur abhängige Tabellen selektiert und ist keine übergeordnete Tabelle der Stufe 1 selektiert, so werden nur die selektierten abhängigen Tabellen (Stufe 2) ausgeblendet.

Beispiel: Das Suchergebnis sieht wie folgt aus:

SAPLANE	Stufe 1
* T000	Stufe 2
T00	Stufe 2
SBOOK	Stufe 1
* SCARR	Stufe 2
SCURX	Stufe 2
SCUSTOM	Stufe 2
* SFLIGHT	Stufe 2

1. Selektion der abhängigen Tabelle T001 (Stufe 2) und der Tabelle SBOOK (Stufe 1).  
⇒ abhängige Tabelle T001 und alle abhängigen Tabellen von SBOOK werden ausgeblendet, die abhängige Tabellen T000 bleibt sichtbar
2. Selektion der abhängigen Tabellen T001 (Stufe 2) und SCARR (Stufe 2).  
⇒ ausgeblendet werden nur die abhängigen Tabellen T001 und SCARR, die restlichen abhängigen Tabellen von SAPLANE und SBOOK bleiben sichtbar



Beachten Sie: Unter dem Ausblenden von abhängigen Tabellen versteht Zeidon nur das Löschen aus der Anzeige. Es erfolgt kein Löschen der Daten im zeidon-internen Work-Objekt. Es ist somit sichergestellt, dass ausgeblendete abhängige Tabellen ohne erneuten SAP Zugriff wieder zur Anzeige gebracht werden können.

Sind abhängige Tabellen ausgeblendet, so werden diese auch nicht mit importiert. Es gilt die Regelung, importiert wird stets nur das, was im Import Fenster sichtbar ist (vgl. Punkt "3.6 SAP Tabellen in ein Zeidon Datenmodell importieren").

#### **See Also**

Option <Show Related Tables> and <Show Related for all Tables>

Option <Remove Tables from List>

## Option <Remove Tables from List>

Die Kontext Menüoption <Remove Tables from List> blendet je nach Stufe der selektierten Tabellen entweder Tabellen der Stufe 1 und/oder abhängige Tabellen der Stufe 2 aus der Anzeige des Import Fensters aus.

Ist eine Tabelle der Stufe 1 selektiert, so wird diese gemeinsam mit ihren abhängigen Tabellen (Stufe 2) gelöscht.

Sind jedoch nur abhängige Tabellen der Stufe 2 selektiert, so bleibt die Tabelle der Stufe 1 selbst erhalten, entfernt werden nur die selektierten abhängigen Tabellen der Stufe 2.

Beispiel: Das Suchergebnis sieht wie folgt aus:

SAPLANE	Stufe 1
* T000	Stufe 2
T00	Stufe 2
SBOOK	Stufe 1
* SCARR	Stufe 2
SCURX	Stufe 2
SCUSTOM	Stufe 2
* SFLIGHT	Stufe 2

1. Selektion der abhängigen Tabelle T001 (Stufe 2) und der Tabelle SBOOK (Stufe 1).  
⇒ abhängige Tabelle T001, die Tabelle SBOOK sowie deren abhängigen Tabellen werden ausgeblendet
2. Selektion der abhängigen Tabellen T001 (Stufe 2) und SCARR (Stufe 2).  
⇒ ausgeblendet werden nur die abhängigen Tabellen T001 und SCARR, die restlichen abhängigen Tabellen von SAPLANE und SBOOK sowie die Tabellen selbst (Stufe 1) bleiben sichtbar

Eine abhängige Tabelle (Stufe 2) kann über die Kontext Menüoption <Show Related Tables> oder <Show Related for all Tables> wieder zur Anzeige gebracht werden.

Anders verhält es sich mit Tabellen der Stufe 1. Diese können nur durch Aktivierung des Buttons <Search> erneut geladen werden, wobei danach die abhängigen Tabellen nicht mehr sichtbar sind.



Beachten Sie: Unter dem Ausblenden von SAP Tabellen bzw. von abhängigen Tabellen versteht Zeidon nur das Löschen aus der Anzeige. Es erfolgt kein Löschen der Daten im zeidon-internen Work-Objekt. Dies bedeutet, solange die Anmeldeparameter beibehalten werden, lädt Zeidon nach Aktivierung des Buttons <Search> die ausgeblendete Tabellen ohne erneuten SAP Zugriff aus dem Work-Objekt.

Haben sich die Anmeldeparameter geändert und ist für den selektierten Suchstring noch keine SAP Anfrage gestartet worden, erfolgt ein neuer SAP Zugriff (vgl. Punkt "3.4.2 SAP Anmeldung").

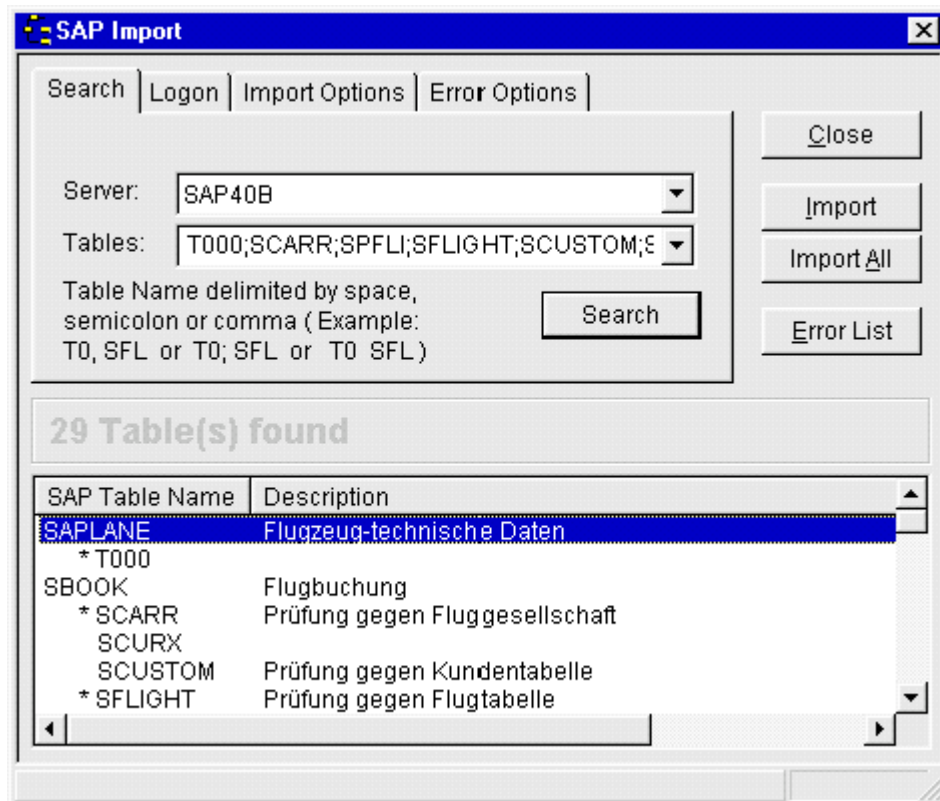
Sind SAP Tabellen bzw. abhängige Tabellen ausgeblendet, so werden diese auch nicht mit importiert. Es gilt die Regelung, importiert wird stets nur das, was im Import Fenster sichtbar ist (vgl. Punkt "3.6 SAP Tabellen in ein Zeidon Datenmodell importieren").

#### **See Also**

Option <Show Related Tables> and <Show Related for all Tables>

Option <Remove Related Tables> and <Remove Related for all Tables>

## Importing SAP Tables into a Zeidon Data Model



Der Import von SAP Tabellen erfolgt über die Buttons <Import> bzw. <Import All>. Sind diese nicht verfügbar, dann prüfen Sie bitte die Einstellungen auf dem Tab <Import Options>.

Per Standard Einstellung importiert Zeidon stets nur Tabellen mit ihren Attributen und Identifiers, Relationships und das Technical Environment (TE) werden nicht gebildet.

Aus Performancegründen sollten Sie diese Standard Einstellung nur dann ändern, wenn Sie in einem Schritt alle gewünschten Tabellen importieren können (vgl. Punkt "3.2 Definition des Importumfangs sowie der Sprache").

Beim Button <Import> werden nur die selektierten Tabellen in das Datenmodell geladen. Dies gilt auch für abhängige Tabellen, wenn diese eingeblendet und selektiert sind.

Der Button <Import All> importiert alle im Import Fenster angezeigten Tabellen mit ihren abhängigen Tabellen, falls sichtbar.

Sollen keine abhängigen Tabellen importiert werden, so müssen diese vor dem Import über das Kontextmenü der Listbox aus der Anzeige entfernt werden (vgl. Punkt "3.5.2 Menüoption "Remove Related Tables" und "Remove Related for all Tables").



Beachten Sie:

Beim Import von Tabellen werden die Anmeldeparameter nicht noch einmal geprüft. Wechseln Sie z.B. nach erfolgreicher Suche von SAP Tabellen auf einen anderen User und starten anschließend den Import, so lädt Zeidon die Daten stets für den User, für den das angezeigte Suchergebnis ursprünglich gefunden wurde.

Alle Fehler und Warnungen, die während des Imports auftreten, schreibt Zeidon in das sogenannte "Error List" Fenster (vgl. Kapitel 5 "Error List Fenster" ). Wird für den aktuellen Import ein Fehler gefunden, lädt Zeidon dieses Fenster in den Vordergrund.

## See Also

[Creating Entities](#)

[Creating Attributes](#)

[Creating Identifiers](#)

[Logon Parameters for the SAP Access](#)

[Definition of Import Range and Language](#)

[Behaviour at Import Errors](#)

[Searching SAP Tables](#)

[Searching dependant SAP Tables](#)

I

## Creating Entities

Für jede zu importierende Tabelle wird eine Entität mit Attributen und Identifiers, aber vorerst ohne Relationships generiert.

Es gelten folgende Regelungen:

### SAP Zugriffe:

Zum Anlegen der Entität selbst wird kein SAP Zugriff notwendig. Den Entitätsnamen sowie die Beschreibungstexte wurden bereits beim Suchen der Tabellen gelesen und im zeidon-internen Work-Objekt gespeichert.

### Entitätsnamen:

Zeidon erzeugt beim Import von SAP Tabellen stets Entitäten, deren Namen den Tabellennamen entsprechen. Neben dem Tabellennamen wird auch der "TE Table Name" gesetzt. Dieser physikalische Name darf im Gegensatz zum logischen Entitätsname nicht geändert werden. Deshalb sperrt Zeidon das Feld "TE Table Name" immer dann, wenn eine Entität über die SAP Import Schnittstelle erzeugt wurde.

Auf der Basis des Feldes "TE Table Name" prüft Zeidon, ob die zu importierende Tabelle im Zeidon Datenmodell bereits existiert. Ist dies der Fall, muss der Benutzer entscheiden, ob diese Entität ersetzt werden soll. Beachten Sie, dass beim Ersetzen die Entität komplett mit allen Attributen und Relationships gelöscht und anschließend neu erzeugt wird.

### Entitätstypen:

Anhand der SAP Tabellen kann nicht eindeutig entschieden werden, ob hier fundamentale, attributive oder assoziative Entitäten erzeugt werden sollen. Es gilt deshalb folgende Regelung:

1. Besitzt eine SAP Tabelle keine Foreign Keys oder solche, die nicht Bestandteil des Keys sind, so wird diese als fundamentale Entität abgebildet.
2. Alle anderen Tabellen werden als attributive Entitäten angelegt.

### Beschreibungstexte:

Die Beschreibungstexte sind je nach eingestellter Import Option entweder in deutsch oder englisch. Besitzt eine Entität keinen Beschreibungstext, so wurde für diese Tabelle kein Eintrag in der SAP Tabelle DD02T gefunden.

### Positionierung der Entitäten:

Bei jedem Import ermittelt Zeidon die letzte vergebene Y-Position und fügt die neuen Entitäten darunter ein. Wurde allerdings eine bereits existierende Entität ersetzt, so wird

diese Position für die neue Entität genutzt.

#### **See Also**

[Creating Attributes](#)

[Creating Identifiers](#)



## Creating Attributes

Zeidon Datenmodelle beinhalten stets nur Attribute, die die Entität selbst beschreiben. Die Attribute der referenzierenden Entität sind nicht sichtbar und werden über Relationships abgebildet.

Beim Import von Tabellen wird vorerst von diesem Grundsatz abgewichen. Zeidon erzeugt für jede Spalte stets ein Attribut. Dabei ist es zu diesem Zeitpunkt uninteressant, ob es sich um die Attribute einer identifizierenden Entität (Foreign Keys) handelt. Diese Attribute (Foreign Keys) werden erst beim Bilden der Relationship entfernt.

Das bedeutet, direkt nach im Import werden Sie in einer Entität Attribute sehen, die evtl. später nach dem Bilden der Relationship (vgl. Punkt "4.3 Erzeugung der Relationships") nicht mehr vorhanden sind. Durch diese Verfahrensweise wird sichergestellt, dass alle Informationen einer SAP Tabelle innerhalb einer Zeidon Anwendung zur Anzeige gebracht werden können, auch wenn die jeweils referenzierende Entität nicht im Datenmodell existiert.

### SAP Zugriffe:

Für jede Entität werden jeweils 2 SAP Zugriffe notwendig. Beim ersten Zugriff werden aus der SAP Tabelle DD03L alle Attribute und deren Eigenschaften gelesen. Anschließend liest Zeidon aus DDFTX die jeweiligen Attributbeschreibungstexte.

### Attributnamen:

Attributnamen werden aus dem jeweiligen Spaltennamen gebildet. Da der Attributname nur ein logischer Name ist und somit auch jederzeit geändert werden kann, setzt Zeidon zusätzlich den für den SAP Zugriff benötigten physischen Namen ( "Column Name for Attribute"). Dieser muss stets identisch zum Spaltennamen sein und darf nicht verändert werden.

### Beschreibungstexte:

Die Beschreibungstexte sind je nach eingestellter Import Option entweder in Deutsch oder Englisch. Besitzt ein Attribut keinen Beschreibungstext, so wurde für diese Spalte kein Eintrag in der SAP Tabelle DDFTX gefunden.

### Not Null Flag:

Die von SAP übergebenen Not Null Informationen sind nicht immer korrekt. So gibt es Attribute die Bestandteil des Keys sind, ohne dass das Not Null Flag gesetzt ist.

Findet Zeidon beim Import ein solches Attribut, wird automatisch das Not Null Flag eingeschaltet, unabhängig davon, ob es im SAP gesetzt ist.

### Domainzuordnung:

Die folgende Tabelle zeigt, welche Domain bei welchen Datentypen zugeordnet wird:

ABAP Datentyp	Zeidon Domain
C = char	Text
D = Date JJJJMMDD FU8	Date
F = Floatingpoint	Decimal oder Decimal0, ... Decimal9, falls Dez.stellen bekannt
I = Integer 4	Integer
N = numeric FU	Decimal oder Decimal0, ... Decimal9, falls Dez.stellen bekannt
P = packed	Decimal oder Decimal0, ... Decimal9, falls Dez.stellen bekannt
T = Time hhmmss FU6	Time
X = AlphaNumeric	AlphaNumeric
b = binaer 1 bype	Integer
s = short	Integer

### **See Also**

[Creating Entities](#)

[Creating Identifiers](#)

# Creating Identifiers

Identifier dienen zur eindeutigen Identifizierung einer Entität. Er kann aus Attributen und/oder Relationships bestehen.



Beachten Sie:

Beim Import erzeugt Zeidon maximal einen Identifier pro Entität. Dieser besteht zu diesem Zeitpunkt nur aus Attributen.

Eine Entität kann somit evtl. erst dann den korrekten Identifier erhalten, wenn durch das Bilden der Relationships (vgl. Punkt "4.3 Erzeugung der Relationships") die identifizierenden Relationships (falls vorhanden) dem Identifier hinzugefügt wurde.

## SAP Zugriffe:

Identifier werden beim Anlegen der Attribute auf der Basis der vorliegenden Attribut-Informationen gebildet. Es ist somit kein zusätzlicher SAP Zugriff notwendig.

## Identifiernamen:

Da für jede Entität nur ein Identifier generiert wird, bildet Zeidon den Namen aus dem Entitätsnamen und dem Suffix "\_ID".

## See Also

[Creating Entities](#)

[Creating Attributes](#)

## Building Relationships and TE

Das Bilden der Relationships und des Technical Environments (TE) sollte aus Performancegründen immer erst dann erfolgen, wenn alle SAP Tabellen importiert wurden.

Der Aufruf zum Bilden der Relationships und des Technical Environments (TE) kann erfolgen über:

- das Fenster "SAP Import" oder
- Menü <Utilities>, Menüoption <SAP Import>, Submenü < Build Relationships and TE>.

### See Also

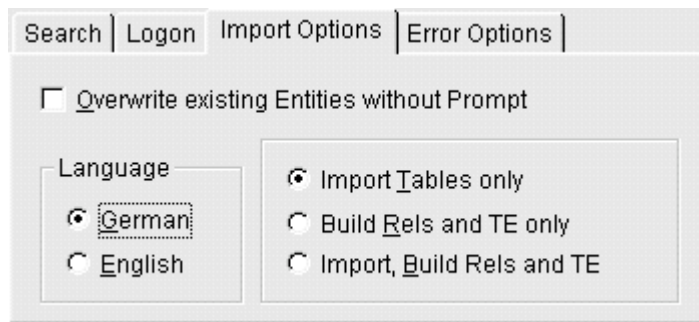
[Call via SAP-Import-Window](#)

[Call via the Option <Build Relationships and TE>](#)

[Creating Relationships](#)

[Creating the Technical Environment \(TE\)](#)

## Call via SAP-Import-Window



Das Bilden der Relationships und des Technical Environments (TE) wird im Import Fenster über die Import Options <Build Rels and TE only> und <Import, Build Rels and TE> unterstützt (vgl. Punkt "3.2 Definition des Importumfangs sowie der Sprache").

Ist die Option <Build Rels and TE only> gesperrt, so existiert keine SAP Entität im Datenmodell. Importieren Sie diese vorher oder wählen Sie die Option <Import, Build Rels and TE>.



Beachten Sie:

Die Option <Import, Build Rels and TE> sollte grundsätzlich nur dann verwendet werden, wenn alle gewünschten SAP Tabellen in einem Schritt importiert werden können. Sind mehrere Imports notwendig, dann ist die Option <Import Tables only> mit anschließendem <Build Rels and TE only> wesentlich performanter.

Beim Generieren werden die Anmeldeparameter geprüft. Wechseln Sie z.B. nach der Suche von SAP Tabellen den User, so fährt Zeidon für den alten User ein SAP Logoff und anschließend ein SAP Logon für den neuen User. Die Relationship Informationen werden für den neuen User gelesen. Sind die Anmeldeparameter identisch, erfolgt keine neue SAP Anmeldung.

### See Also

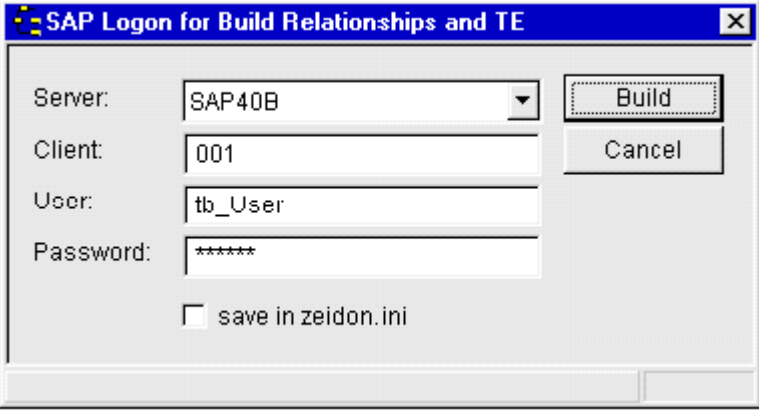
[Call via the Option <Build Relationships and TE>](#)

[Creating Relationships](#)

[Creating the Technical Environment \(TE\)](#)

## Call via the Option <Build Relationships and TE>

Vor dem Bilden der Relationships und des Technical Environments (TE) müssen die Parameter für die SAP Anmeldung abgefragt werden. Ist das Fenster "SAP Import" geöffnet, gelten die dort eingestellten Werte. Ansonsten lädt Zeidon das folgende Logon Fenster.



Wählen Sie hierfür über die Combobox <Server> den gewünschten Server. Ist der Server, auf den zugegriffen werden soll, nicht aufgelistet, dann erfassen Sie diesen in der Datei sirenv.txt (vgl. Kapitel 1 "Voraussetzungen").

Findet Zeidon in der Datei zeidon.ini die SAP Default Einstellungen für das Logon, werden diese beim Öffnen des Fensters geladen. Sind diese Werte nicht gespeichert, müssen Sie Mandant (Client), Passwort und User erfassen.

Sollen diese Eingaben künftig entfallen, ist die Checkbox <save in zeidon.ini> zu aktivieren.

Immer wenn diese Checkbox aktiviert ist, aktualisiert bzw. erzeugt (falls noch nicht vorhanden) Zeidon nach dem Schließen des Fenster folgende Einträge unter der Gruppe [SAP] :

- DefaultClient
- DefaultUser
- DefaultPassword (wird verschlüsselt gespeichert).



Beachten Sie: Der Aufruf dieser Menüoption ist immer dann gesperrt, wenn im Datenmodell keine Entität gefunden wird, die durch den SAP Import erzeugt wurde.

**See Also**

Call via SAP-Import-Window

Creating Relationships

Creating the Technical Environment (TE)

## Creating Relationships

Vor Prozeßbeginn werden stets alle Relationships gelöscht, die zwischen SAP Entitäten bestehen.

Anschließend prüft Zeidon für jede SAP Entität, ob Foreign Key Informationen in der SAP Tabelle DD08V vorliegen. Ist das der Fall, wird in einem zweiten Schritt geprüft, ob der Foreign Key Bestandteil des Keys ist.

Für jeden gefundenen Foreign Key sucht Zeidon die referenzierenden Entität im Datenmodell. Existiert diese Entität, wird die Relationship erzeugt.



Ist die referenzierende Entität im Datenmodell nicht vorhanden und ist der Foreign Key Bestandteil des Keys, so handelt es sich hier um eine Key Relationship (identifizierende Relationship), deren Existenz für den SAP Zugriff zwingend erforderlich ist.

Die abhängige Entität wird deshalb automatisch angelegt, die Relationship erzeugt und dem Identifier hinzugefügt. Einen Hinweis über das automatische Anlegen dieser Entität finden Sie im "Error List" Fenster.

Wie unter Punkt "3.6.2 Erzeugung der Attribute" bereits beschrieben, wurde beim Import der SAP Tabellen für jede Spalte stets ein Attribut angelegt. Dabei war es zu diesem Zeitpunkt uninteressant, ob es sich um die Attribute einer identifizierende Entität (Foreign Keys) handelt.

Da in Zeidon Datenmodellen stets nur die Attribute vorhanden sein dürfen, die die Entität selbst beschreiben, müssen die Attribute der identifizierende Entität entfernt werden. D.h., immer wenn eine Relationship erzeugt wurde, sucht Zeidon nach dem Attribut der identifizierende Entität und löscht dieses.

#### SAP Zugriffe:

Für jede SAP Entität werden jeweils 2 SAP Zugriffe notwendig. Beim ersten Zugriff werden aus der SAP Tabelle DD08V alle Relationships und deren Eigenschaften gelesen.

Anschließend prüft Zeidon über die SAP Tabelle DD03L, ob die Relationship Bestandteil des Keys ist.

#### Relationshipnamen:

Relationshipnamen basieren auf dem Namen des Attributes der über identifizierende Relationship vererbt wurde (vgl. oben). Es gelten folgende Regelungen:

- Attributname + Leerzeichen + Schlüsselwort "for" bzw.
- Schlüsselwort "has" + Leerzeichen + Attributname

#### Kardinalitäten:

Die folgende Tabelle zeigt, wie Zeidon die SAP Kardinalitäten auflöst:

SAP Kardinalität	Zeidon Kardinalität
1	Min: 1, Max: 1
C	Min: 0, Max: 1
N	Min: 1, Max: m
CN	Min: 0, Max: m

## **See Also**

[Call via SAP-Import-Window](#)

[Call via the Option <Build Relationships and TE>](#)

[Creating the Technical Environment \(TE\)](#)

## Creating the Technical Environment (TE)

Über das Technical Environment (TE) wird durch Anlegen einer Data Source vom Typ "SAP R3" der physikalische Zugriff auf das SAP System definiert.

Existiert zum Zeitpunkt der Erzeugung der Data Source noch kein TE, so wird dieses von Zeidon automatisch erstellt. Ist das TE bereits vorhanden, werden alle Data Sources vom Typ "SAP R3" gelöscht.

Beim Erzeugen der Data Source werden die SAP Entitäten und deren Relationships in die SAP Tabellenstruktur überführt. Es ist somit kein SAP Zugriff notwendig.

### See Also

[Call via SAP-Import-Window](#)

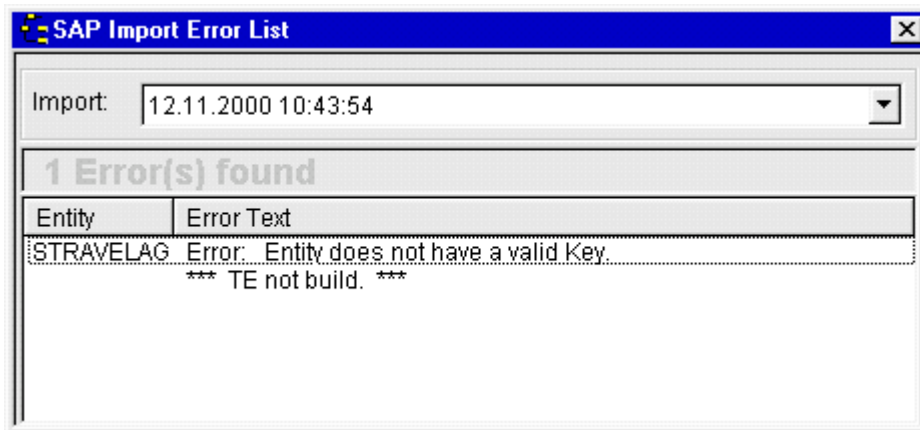
[Call via the Option <Build Relationships and TE>](#)

[Creating Relationships](#)

## Error List Window

Das "Error List" Fenster dient zum Abfragen von Fehlern oder Warnungen, die während des Imports von Tabellen und/oder beim Bilden der Relationships bzw. des Technical Environments (TE) aufgetreten sind.

Wurde der jeweilige Prozess ohne Fehler/Warnung beendet, so ist die Liste leer.



### See Also

[Creating the Error-List-Window](#)

[Contents of the Error List](#)

[Context Menu of the Error List](#)

## Creating the Error-List-Window

Der Aufruf des "Error List" Fensters erfolgt über den Button <Error List> des Fensters "SAP Import".



Beachten Sie:

Das "Error List" Fenster wird von Zeidon automatisch geladen, sobald ein Prozess fehlerhaft beendet wurde.

Ist das Fenster "SAP Import" geöffnet und wurde hier die Error Option <Show Error List> aktiviert, so lädt Zeidon die Error List bereits zum Prozessbeginn und aktualisiert diese, sobald ein Fehler oder eine Warnung auftritt.

### See Also

[Contents of the Error List](#)

[Context Menu of the Error List](#)

## Contents of the Error List

Traten Fehler oder Warnungen beim Importieren von SAP Tabellen bzw. beim Bilden der Relationships und des TEs auf, dann gelten folgende Regelungen:

- Für jeden Prozess speichert Zeidon Datum und Uhrzeit sowie die dabei aufgetretenen Fehler bzw. Warnungen.
- Kann dem aufgetretenen Fehler/Warnung eine Entität zugeordnet werden, so wird diese angezeigt.
- Führt der Fehler zum Abbruch des jeweiligen Prozesses, erscheint eine entsprechende Meldung.
- Werden aufgrund der angezeigten Fehler Korrekturen im Datenmodell notwendig, so können diese bei geöffneter Error List durchgeführt werden. Nutzen Sie je nach Bedarf das in der Error List integrierte Kontextmenü.
- Alle Fehler bzw. Warnungen vorangegangener Prozesse können über die Combobox <Import> solange abgerufen werden, bis entweder ein anderes Projekt geladen oder das Zeidon Tool "Data Model" geschlossen wird.

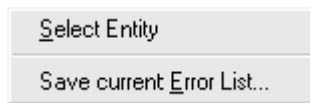
### See Also

[Creating the Error-List-Window](#)

[Context Menu of the Error List](#)

## Context Menu of the Error List

Beinhaltet die Error List Fehler bzw. Warnungen, so kann über die Listbox das folgende Kontextmenü abgerufen werden:



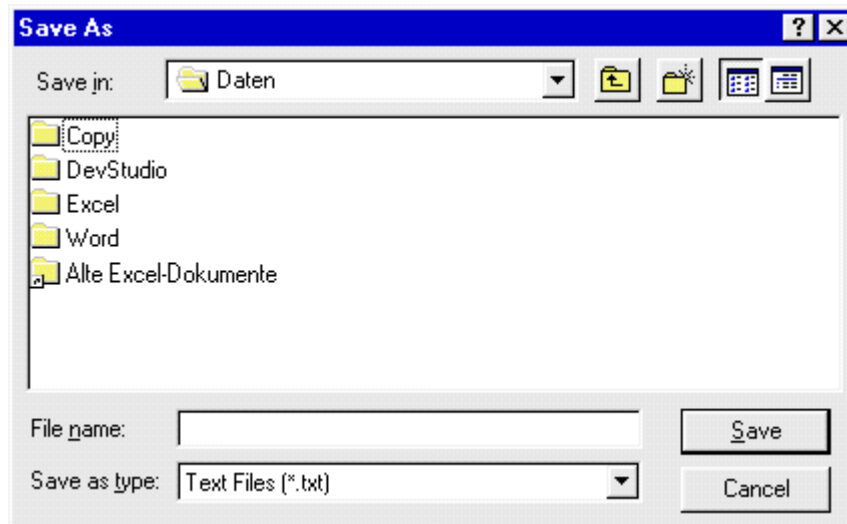
### Menüoption "Select Entity":

Ist bei einer Fehlermeldung bzw. Warnung ein Entitätsname aufgeführt, so wird diese Entität im Datenmodell selektiert. Das ist vor allem dann sehr hilfreich, wenn mit sehr großen Datenmodellen gearbeitet wird.

### Menüoption "Save current Error List":

Möchten Sie die Korrekturen am Datenmodell später realisieren, dann empfiehlt sich das Speichern der aktuellen Error List.

Gespeichert wird über das folgende Fenster in eine Text-Datei Ihrer Wahl:



### See Also

[Creating the Error-List-Window](#)

[Contents of the Error List](#)

# Restrictions in SAP Data Models

SAP Datenmodelle zeigen u.a. einen Ausschnitt der SAP Datenbankstruktur in Form eines Entity Relationship Diagramms. Dieses Datenmodell bildet die Basis für die Erstellung von Zeidon Anwendungen.

Eine Zeidon Anwendung kann nur dann SAP Daten fehlerfrei lesen, wenn die definierte Zugriffsschicht (logische Objekte) mit der SAP Struktur identisch ist.

Es ist für den Benutzer nicht immer eindeutig entscheidbar, ob und welche Änderungen am Datenmodell zu Lesefehlern führen könnten. Deshalb unterliegen SAP Datenmodelle den folgenden Restriktionen:

## Entitäten:

Bei Entitäten die durch einen SAP Import erzeugt wurden, sind die physikalischen Namen "TE Table Name" und "TE Table Owner" gesperrt. Der "TE Table Name" wird für die Bildung des Technical Environments (TE) benötigt und muss identisch zur SAP Tabelle sein.

## Attribute:

In Analogie zu den Entitäten prüft Zeidon auch bei den Attributen, ob es sich um ein SAP Attribut (erzeugt durch SAP Import) handelt. Ist das der Fall, werden die Felder "Column Name for Attribute" und "Column Name for Foreign Key" für die Eingabe gesperrt.

Möchten Sie zu einer SAP Entität ein neues Attribut erfassen, so ist dies nur als Work-Attribut möglich. Konkret bedeutet dies, dass Zeidon die Checkbox <Work> aktiviert und sofort sperrt.

## Identifizier:

Bei den Identifizier einer SAP Entität darf maximal der Identifiziername geändert werden. Alle anderen Funktionen sind gesperrt. Dies schließt auch das Anlegen und Löschen von Identifizier mit ein.

## Relationships:

Relationships, die zwischen zwei SAP Entitäten bestehen, können nicht bearbeitet oder gelöscht werden. Des weiteren dürfen keine Relationships von oder zu einer SAP Entität erzeugt werden.





## Restrictions in SAP Data Sources

Das Zeidon Tool "Physical Data Model" dient zur Definition des Technical Environments (TE).

Es ist die zentrale Stelle, an der durch das Anlegen von Data Sources entschieden wird, auf welche Datenbasis der physikalische Zugriff erfolgen soll.

Möchten Sie SAP Zugriffe realisieren, so wird eine Data Source vom Typ "SAP R3" benötigt. Diese Data Source sollten Sie niemals händisch anlegen, sondern stets von Zeidon generieren lassen (vgl. Kapitel 4 "Bilden der Relationships und des TEs").

Beachten Sie, dass die Tabellen und Spalten einer SAP Data Source für den Benutzer unveränderbar sind. Außer an den Beschreibungstexten wird keine weitere Manipulation unterstützt.

Ändern Sie Beschreibungstexte, so bedenken Sie bitte, dass diese nur so solange Bestand haben, bis eine erneute TE Generierung über das Zeidon Tool "Data Model" angestoßen wird.

## **Appendix: Siron Help Draft**

### **See Also**

[Preparing\\_the\\_SAP\\_R\\_3\\_Interface](#)

## Preparing the SAP R/3-Interface

Mit der Siron-SAP R/3-Schnittstelle werden Lesezugriffe auf Tabellen und Views des R/3-Systems durchgeführt. Die Siron-Schnittstelle arbeitet auf Basis der SAP R/3-Komponente RFC-SDK. Die Kommunikation erfolgt zwischen einer RFC-fähigen ABAP/4-Funktion und der in Windows bzw. OS/2 integrierten Siron-Programmbibliothek. Daher muß die SAP R/3- Komponente RFC-SDK auf dem PC (Client) installiert sein. Die SAP R/3 RFC-Schnittstelle verlangt eine SAP R/3-Version ab SAP R/3 Release 3.0c.

Aufgrund der aufwärtskompatiblen Frontend-Versionen der SAP R/3-Releases (SAP-Version 4.x ist derzeit noch nicht getestet) ist für alle SAP R/3-Kernels ab SAP R/3-Version 3.0c für die Siron-Version 98.1 die Frontend-Installation der SAP R/3 RFC-SDK-Schnittstelle unter SAP R/3-Version 3.1h durchzuführen. Falls diese Installation mit firmeneigenen RFC-Anwendungen unter speziellen R/3-Frontend-Releases kollidieren

Name Domäne	Format		Bedeutung
YTBBC H64	CHAR	Länge 64	String 64 Stellen
YTBBC H128	CHAR	Länge 128	String 128 Stellen
YTBBC H256	CHAR	Länge 256	String 256 Stellen
YTBBC H512	CHAR	Länge 512	String 512 Stellen
YTBBC H1024	CHAR	Länge 1024	String 1024 Stellen
YTBBC H4096	CHAR	Länge 4096	String 4096 Stellen
YTBBC H8000	CHAR	Länge 8000	String 8000 Stellen
Bei der Definition der Domänen über 255 Stellen werden Sie eine Warnung erhalten, daß das Element lediglich als Struktur verwendet werden darf; dies ist jedoch in dieser Anwendung immer gewährleistet.			
YTBBD COUNT	INT4	Länge 10	Zählfeld
YTBBD FLAG	CHAR	Länge 1	Flag
YTBBD OPCO	CHAR	Länge 4	Operationscode Ton Beller
YTBBNUMC6	NUMC	Länge 6	numerische Längfelder Ton Beller

4. Anlegen der folgenden Datenelemente in der SAP R/3 ABAP/4-Workbench; die Namen liegen ebenfalls im kundenspezifischen Namensraum:

Name Datenelement	Domäne	Bedeutung
YTBDECIM	YTBBNUMC6	Dezimalstellen eines Datenelementes
YTBFCOUNT	YTBBNUMC6	Anzahl Datenelemente in Tabelle
YTBFLAG	YTBFLAG	Flag Tabellenverarbeitung Ton Beller
YTBFLN	YTBBNUMC6	Länge eines Datenelementes
YTBKEYLENG	YTBBNUMC6	Keylänge einer Tabelle
YTBKEYPOS	YTBBNUMC6	Keyposition in einer Tabelle
YTB OFFS	YTBBNUMC6	Offset eines Datenelementes in Tabelle
YTBOPCO	YTBBDOPCO	Operations code Ton Beller
YTBPOS	YTBBNUMC6	Position des Datenelementes in Tabelle
YTBRECLN	YTBBNUMC6	Länge eines kompletten Datensatzes
YTB SRCLEN	YTBBNUMC6	Länge der Tabellen-Source-Ausgabe
YTBROWC	YTBBD COUNT	Anzahl zu lesender Tabellensätze
YTBROWS	YTBBD COUNT	Anzahl zu überlesender Tabellensätze
YTB TABTYP	YTBBD FLAG	Tabellentyp-Kennzeichen
YTBW A72	YTBBC H72	String zur Übergabe der WHERE-Clausel
YTBW A64	YTBBC H64	Übergabestring bis 64 Stellen
YTBW A128	YTBBC H128	Übergabestring bis 128 Stellen
YTBW A256	YTBBC H256	Übergabestring bis 256 Stellen
YTBW A512	YTBBC H512	Übergabestring bis 512 Stellen
YTBW A1024	YTBBC H1024	Übergabestring bis 1024 Stellen
YTBW A4096	YTBBC H4096	Übergabestring bis 4096 Stellen
YTBW A8000	YTBBC H8000	Übergabestring bis 8000 Stellen

5. Anlegen der folgenden Strukturen in der SAP R/3-ABAP/4-Workbench; die Namen liegen ebenfalls im kundenspezifischen Namensraum:

**a) YTBBFIELDS**

Struktur für die Dictionary-Beschreibung einer R/3-Tabelle, bestehend aus:

	Feldname	Datenelement	Erläuterung
1	TABNAME	TABNAME	DD02L-TABNAME
2	FIELDNAME	FIELDNAME	DD03L-FIELDNAME
3	FIELDPOS	YTBBPOS	
4	OFFSET	YTBBUFFS	
5	LENGTH	YTBBFLEN	
6	DECIMALS	YTBBDECIM	
7	TYPE	YTBBTABTYP	
8	FIELDTEXT	AS4TEXT	60 Bytes Langtext aus DD03T

**b) YTBORDER**

Struktur für die Übergabe der ORDER-Clause

(Diese Struktur wird für SAP R/3 Version 3.0 noch nicht benötigt, aber ggf. in SAP R/3 Folgeversionen), bestehend aus:

Feldname: ORDER

Datenelement: YTBWA72

**c) YTBBREAD**

Kommunikationsstruktur ABAP/4-Anwendung mit Siron, bestehend aus:

	Feldname	Datenelement
1	OPCODE	YTBBOPCO
2	FLAG	YTBBFLAG
3	ROWSKIP	YTBBROWS
4	ROWCOUNT	YTBBROWC
5	FIELD COUNT	YTBBFCOUNT
6	RECLEN	YTBBRECLEN
7	KEYLEN	YTBBKEYLEN
8	KEYPOS	YTBBKEYPOS
9	TABTYP	YTBBTABTYP
10	SRCLEN	YTBBSRCLEN

#### d) YTBBWHERE

Struktur für die Übergabe der WHERE-Clause, bestehend aus:

Feldname: WHERE

Datenelement: YTBBWA72

**e) YTBBST64**

Struktur für die Übergabe der Tabellensätze bis 64 Stellen, bestehend aus:

Feldname: DATATB64

Datenelement: YTBBWA64

**f) YTBBST128**

Struktur für die Übergabe der Tabellensätze bis 128 Stellen, bestehend aus:

Feldname: DATATB128

Datenelement: YTBBWA128

**g) YTBBST256**

Struktur für die Übergabe der Tabellensätze bis 256 Stellen, bestehend aus:

Feldname: DATATB256

Datenelement: YTBBWA256

**h) YTBBST512**

Struktur für die Übergabe der Tabellensätze bis 512 Stellen, bestehend aus:

Feldname: DATATB512

Datenelement: YTBBWA512

**i) YTBBST1024**

Struktur für die Übergabe der Tabellensätze bis 1024 Stellen, bestehend aus:

Feldname: DATATB1024

Datenelement: YTBBWA1024



#### **j) YTBBST4096**

Struktur für die Übergabe der Tabellensätze bis 4096 Stellen, bestehend aus:

Feldname: DATATB4096

Datenelement: YTBBWA4096

#### **k) YTBBST8000**

Struktur für die Übergabe der Tabellensätze bis 8000 Stellen, bestehend aus:

Feldname: DATATB8000

Datenelement: YTBBWA8000

Generell können alle Namen frei in dem kundeneigenen Namensraum gewählt werden, wenn die entsprechenden Beziehungen (Domäne - Datenelement - Feldname) eingehalten werden.

Die Reihenfolge der Felder in der Struktur YTBBFIELDS ist unbedingt einzuhalten. Alle Dictionary-Objekte müssen nach der Definition gesichert und aktiviert werden.

Es ist nun ein neuer Funktionsbaustein im kundeneigenen Namensraum anzulegen. Standardmäßig verwenden wir den Namen YTBB\_READ\_TABLE. Sie können auch hierfür einen anderen Namen wählen, müssen diesen dann jedoch als Parameter in der Siron Environment Datei mitführen (vgl. untenstehendes Beispiel).

Der Quelltext dieses Funktionsbausteins wird per -UPLOAD- vom PC in das SAP R/3 System übernommen, ebenso wie die Funktionsbausteindokumentation (ytbnsrc3.txt und ytbndoc3.txt). Ggf. sind in dem Quelltext im vorstehenden Kommentar von Ihnen abgeänderte Namen anzupassen.

Einzugeben sind Import-, Export- und Tabellen-Parameter sowie die Ausnahmebedingungen. Im einzelnen sind dies wie folgt:

#### **1. Import-Parameter:**

	Import-Parameter	Bezugsfeld	Vorschlag	Opt.	Bedeutung
1	QUERY_TABLE	DD02L-TABNAME	space		Name der Tabelle / der View
2	OPERATION	YTBBREAD-OPCODE	space	x	Operationscode TB
3	USERNAME	USR04-BNAME	space	x	SAP R/3-User
4	TABVALUE	XU180-VALUE	space	x	Tabellenname für AUTHORITY_CHECK
5	DELIMITER	YTBBREAD-FLAG	space	x	Begrenzungszeichen
6	ROW SKIPS	YTBBREAD-ROWSKIP	0	x	Anzahl zu überlesender Einträge
7	ROW COUNT	YTBBREAD-ROWCOUNT	0	x	Anzahl zu lesender Einträge

## 2. Export-Parameter:

	Export-Parameter	Bezugsfeld	Bedeutung
1	FIELD COUNT	YTBBREAD-FIELD COUNT	Anzahl Tabellenfelder
2	RECL EN	YTBBREAD-RECL EN	Datensatzlänge
3	KEYLEN	YTBBREAD-KEYLEN	Länge Primärkey
4	KEYPOS	YTBBREAD-KEYPOS	Position Primärkey in Tabelle
5	TABTYP	YTBBREAD-TABTYP	Typ der Tabelle
6	SRCL EN	YTBBREAD-SRCL EN	Länge der Übergabe-Source

## 3. Tabellen-Parameter:

	Tabellen-Parameter	Bezugsstruktur	Bedeutung
1	OPTIONS	YTBBWHERE	Struktur für WHERE-Claus e (in)
2	FIELDS	YTBBFIELDS	Struktur für ausgewählte Tabellenfelder (in/out)
3	DATATB64	YTBBST64	Struktur für zurückgelieferte Daten der Tabelle (out)
4	DATATB128	YTBBST128	Struktur für zurückgelieferte Daten der Tabelle (out)
5	DATATB256	YTBBST256	Struktur für zurückgelieferte Daten der Tabelle (out)
6	DATATB512	YTBBST512	Struktur für zurückgelieferte Daten der Tabelle (out)
7	DATATB1024	YTBBST1024	Struktur für zurückgelieferte Daten der Tabelle (out)
8	DATATB4096	YTBBST4096	Struktur für zurückgelieferte Daten der Tabelle (out)
9	DATATB8000	YTBBST8000	Struktur für zurückgelieferte Daten der Tabelle (out)



## Ausnahmen:

Ausnahmen	Bedeutung
TABLE_NOT_AVAILABLE	Tabelle nicht verfügbar
TABLE_WITHOUT_DATA	Tabelle nicht gefüllt
OPTION_NOT_VALID	WHERE-Clausel fehlerhaft
FIELD_NOT_VALID	Falscher Feldname ausgewählt
NOT_AUTHORIZED	User nicht zum Lesen dieser Tabelle berechtigt
DATA_BUFFER_EXCEEDED	Datenpuffer reicht nicht aus zur Aufnahme der Tabellensätze
USER_DOES_NOT_EXIST	User existiert nicht in diesem R/3-System

Die Bezeichnung und die Reihenfolge der Parameternamen und Ausnahmebedingungen muß unbedingt den oben aufgeführten Namen und der Reihenfolge entsprechen; die Namen der Bezugfelder und der Bezugsstrukturen sind ggf. den individuell gewählten Namen des Data Dictionary anzupassen.

Nach dem Upload des Quelltextes und der Funktionsbaustein Dokumentation sind die folgenden Aktivitäten für den Funktionsbaustein durchzuführen:

1. Prüfen des Funktionsbausteins  
Es sollte die Meldung "Es wurden keine Syntaxfehler gefunden" erscheinen.
2. Sichern des Funktionsbausteins
3. Aktivieren des Funktionsbausteins als RFC-fähigen Funktionsbaustein  
(Verwaltungsdaten anpassen)

Da vor dem Zugriff auf die Daten einer Tabelle eine Berechtigungsprüfung durchgeführt wird, muß nun eine Objektklasse YTBB mit Berechtigungsobjekt YTBB\_READ angelegt werden; dabei ist als einiger Eintrag unter Berechtigungsobjekte Felder der Eintrag "table", d.i. Tabellename, vorzunehmen. Anstatt YTBB / YTBB\_READ können Sie andere Namen aus Ihrem kundeneigenen Namensraum wählen, müssen dann aber den Quelltext des Funktionsbausteins bei Aufruf der Funktion AUTHORITY\_CHECK in Zeile 39 "object = 'YTBB\_READ' " entsprechend anpassen und die obigen 3 Aktivitäten nochmals durchführen.

Zu Testzwecken sollte eine Berechtigung YTBB\_S\_READ angelegt werden, die mit Eintrag "\*" Zugriff auf alle Tabellen ermöglicht. Weiterhin muß ein zugehöriges Berechtigungsprofil, z.B. YTBB\_P\_READ, erstellt

werden, das wiederum die oben definierte Berechtigung beinhaltet. Dieses Profil ist nun dem User, der mit Siron R/3 Tabellen lesen will, zuzuordnen.

Für den produktiven Betrieb können dann weitere Berechtigungen und Profile definiert werden, die dann pro User die Tabellen und Views, auf die zugegriffen werden darf, eingrenzen.



# User's Guide

Version 2001 - Release 010407



Introduction



Workstation Administration



Data Model



Zeidon Objects



Domains



Global Operations



Dialogs

Accessing SAP R/3 with Zeidon

# Introduction

Overview

The Windows Environment

User Input

Window Components

Window Actions

Interacting with the Dialogs

## Introduction - Overview

The Zeidon Users Guide is a detailed reference to the Zeidon application development system. It is designed to provide a set of detailed instructions for the user working with the Zeidon tool Set. It is assumed the user of this manual has a basic understanding of Zeidon, its concepts and components. That information can be obtained from the Zeidon Basics manual.

This chapter provides a summary of the major features of *Windows*, intended for users new to that interface. An overview of some of the standard user interfaces of the Zeidon tools follows.



## The Windows Environment

For a detailed source of information about the *Windows* interface, refer to your *Windows* documentation. The following is a brief summary.

*Windows* displays windows and icons on the screen. An application may have several related windows open at the same time and further, several applications may be open concurrently. Although many windows can be open, only one window at a time receives input from the user through the mouse or keyboard. This window is said to be *active*. It is also said that the active window has the *focus* of the system. All other windows are said to be *inactive* or *out of focus*. The user can tell which window is active by noting that the active window has its title bar highlighted.

Windows in an application can be categorized roughly into two groups: those that allow you to change focus to another window within the application and those that don't. Windows that do not allow you to change the focus within the application are often encountered in situations where the user must respond to the application before processing can continue. Note that you can almost always change focus to windows in other applications.

Any logical grouping of windows that performs some task in an application is often referred to as a windowing *dialog*. An individual window is made of several components such as title bars, scroll bars and menus among others.

## User Input

The keyboard and the mouse are the primary input devices for a user. Instead of a mouse, sometimes an alternative pointing device such as a track ball is used. We will refer to any pointing device as a mouse for simplicity. Keyboard and mouse input is directed only to the current active window.

### See also:

[Using the Mouse](#)

[Using the Keyboard](#)

## Using the Mouse

You can use the mouse to make different kinds of selections in a window, to move objects and to initiate various actions.

The *Windows* operating system assumes the use of a mouse with two buttons; a primary button and a secondary one. For right handed mouse users, the left mouse button is set as the primary button and the right button as the secondary. For left handed mouse users, this is usually reversed. In our literature, we will refer to the primary button as the *left button* and the secondary as the *right button* regardless of the orientation of the mouse. Unless otherwise specified, all mouse actions refer to the left button. The most common types of actions you'll make with the mouse include:

**Clicking and Double-clicking** - Clicking the left button once on an item highlights it and establishes focus on it. Clicking twice selects the item, usually causing an action to take place or opening a detail window that lists the item's properties. Single-clicking the right mouse while positioned on certain controls will present a pop-up menu.

**Highlighting** - You highlight items differently depending on what type of control you interacting with. Most often you place the mouse over an item, then click the left mouse button once to establish focus on the item.

**Selecting** - Place the mouse pointer over an item, then double-click the left button to select the item. Selecting an item is usually equivalent to highlighting it and then initiating some action on it.

**Dragging** - Place the mouse pointer on an item, hold the left button down, then drag the mouse to the item's new location before releasing it. Not all items can be moved.

## Using the Keyboard

Many actions can be performed with either the mouse or the keyboard. To highlight an item in a window without using the mouse you can often use the tab key to activate the desired item. Repeatedly pressing tab will cycle you through all items you can get focus on. (Use Alt+tab to cycle through open applications.) To select an item using the keyboard, tab to the item and press the **Enter** key.

## Window Components

A typical dialog window is shown below

Domain Maintenance

File Domain DomainGroup Help

Domain Group: TZAPDMAA

Name: AlphaNumeric Type: Expression

Desc: Characters 'a-z', 'A-Z', '0-9'

Prev

Next

Data Type: String - Any length

Maximum String Lth: 254 Decimal Precision: 1234

Context Name: ALPHANUMERIC Default: ☐ Restricted

Operation: zdmText

Standard Zeidon Description

Some of the various window components are:

**Title Bar** - The title (or caption) for the window appears in the bar at the top of the window frame. The title bar is divided into two sections. The left section contains the window's caption while the right section contains the window's title.

**System Menu** - A system menu icon is located in the top left corner of many windows. It is used to obtain a pull-down with window-related actions such as moving, sizing, or closing.

**Sizing Icons** - Many windows also contain window sizing icons in the far right corner of the window. These icons make the window larger or smaller depending on which one is pressed. Maximizing a window makes it fill the entire screen while minimizing it makes the window into an icon. Double-clicking on a minimized window restores the window to its original size.

**Scroll Bar** - Scroll bars let you know that the window contains more information than what you see on the screen. The vertical scroll bar moves the contents up or down, and the horizontal scroll bar moves the contents left or right. Scroll bars appear only if the window contains something that can be scrolled. If hidden information can be seen only by scrolling vertically, the vertical scroll bar appears, but not the horizontal one. Using scroll bars is described in more detail below.

**Menu Bar** - The menu bar is directly underneath the caption bar, and shows you the types of functions you can do in the application. Each menu bar menu opens a pull-down items list from which you can select options.

**Dynamic Information Line** - The Dynamic Information Line (DIL) at the very bottom of the window is used to display information about the currently highlighted field or button.



## Window Actions

Some user actions affect properties of the window being displayed as opposed to interacting with the application itself. In the following related sections, we give just a sample of these windows actions.

### See also:

[Opening a Window](#)

[Activating a Window](#)

[Scrolling](#)

[Moving a Window](#)

[Closing a Window](#)

## **Opening a Window**

Whenever you access a dialog from the Zeidon program group, a window is opened. Once inside a dialog, other windows will be opened depending upon which options, buttons, and list box items are selected.



## **Activating a Window**

To activate a window, simply move the mouse anywhere on the window and click on it. You should be able to tell which window is active by the color of its title bar or the position of the window on the screen. Keep in mind that just because a window is open doesn't mean it's active. You can have numerous windows open on your screen, but only one is active.

## Scrolling

Scrolling allows you to view portions of a window that are outside the display area. If the window you're working in has scroll bars and arrows around its perimeter, you can scroll large increments by dragging the scroll box along the scroll bar either left or right (with the horizontal scroll bar), or up and down (with the vertical scroll bar). To scroll in small amounts, move the pointer to a scroll arrow and click the mouse button once. The contents of the window move one increment in the direction that the arrow points. If you hold down the mouse button on the scroll arrow, the scrolling continues one increment at a time.

## Moving a Window

You can use the mouse to move a window to any area of the screen:

1. Place the mouse pointer on the active windows title bar.
2. Press the mouse button, and hold it down while moving the pointer. A faint outline of the window follows the pointer movement.
3. When the window outline is where you want it, release the mouse button.

You can also move a window by clicking the system menu icon in the upper left corner and selecting **Move**. Use the keyboard's arrow keys (left, right, left, and right) to position the window anywhere you want and then hit **Enter**.

## Closing a Window

Depending upon the type of window, you can use one of the following methods for closing windows:

- From main level windows select **Exit** from the **File** pull-down.
- From dialog boxes, click a push button, such as **Cancel** or **OK** to return you to the prior window.

Keep in mind that if you think you will be returning to the current main level window, you can often minimize it instead of exiting and reopening it later.

## Interacting with the Dialogs

Whereas some user actions only affect the look of a window, others interact directly with the application that it running. As noted above, a window or group of windows that can be logically grouped together for the accomplishment of some task is sometimes called a dialog. Interaction with dialogs occurs through a series of menus (pull-downs and pop-up) and dialog boxes.

- Menus list what actions you can perform in the dialog.
- Dialog boxes ask questions, prompt for input, or advise you of problems when you request an action. Whenever possible, default selections are set to the choice you're most likely to make.

### See also:

[Menus](#)

[Dialog boxes](#)

## Menus

Pull-down menus occur when you place the mouse pointer over the menu name on the window's menu bar and click. Pop-up menus occur when you place the mouse pointer over certain select controls, such as list boxes, and then right-mouse click. Some items on the menu are followed by an ellipse (...). This means that if you make that choice, you'll encounter a dialog box requesting further instructions. Some items on the menu may be disabled (or "grayed"), meaning that they cannot be performed due to certain conditions. To close the menu without choosing an option, just move the mouse pointer away from the menu and click.

## Dialog Boxes

Whenever a dialog requires additional information to do something or finds a potential problem, it displays a dialog box. A dialog box will usually contain lists of alternatives to choose from and/or places to enter text. Use the series of push buttons to respond when you are finished.

Dialog boxes have different kinds of fields depending on the type of information they require. Sometimes you can choose between many options, other times you need to make an either/or choice. The most common controls in dialog boxes are described here:

**Check Boxes** - used for options that don't have to be exclusive. You can make one or several choices. To select a check box move the mouse over it and click. An "X" appears in the box when it is selected. To de-select it, move the pointer over the "X" and click.

**Radio Buttons** - Radio buttons are used when one and only one selection out of several can be made. Choosing one button turns off any other. To select a radio button move the mouse over it and click. The button will appear filled in when selected.

**List Boxes** - List boxes present a list of things to choose from, and are distinguished by scroll bars. You can normally make a selection by double-clicking on the item you want. Some list boxes allow for multiple selection, meaning you can select more than one item at a time. The technique of selecting multiple items in a list box varies.

Where it is likely that the user will need to select a sequence of items, click on an item to mark it as an anchor. Then use Shift+click to select another item. This selects all items between (and including) the second selected item and the anchored item. To make a non-sequential selection of more than one items, use Ctrl+click to toggle the select state of each item individually. As usual, the selected items are highlighted.

**Edit Boxes** - Edit boxes are places where you can input text. A single line entry field accepts only one line of text. A multiple line edit box allows carriage returns and word wrapping.

**Combo Boxes** - Combo boxes are a combination of single selection list boxes and edit boxes and are used when either a selection or an input entry is needed. There are various styles of combo boxes. A drop-down list contains a static control and list box (presented when the drop-down arrow is pressed). Selections are made by clicking from the list only. Entered character keys position and map the list box entry to the static control on matches. A drop-down contains an edit control and list box which is presented when the drop-down arrow is pressed. Selections are made from the list or keyed in the edit box. Entered character keys also position the list box if matches occur. A simple combo box contains an edit control and associated list box. Selections are made from the list or keyed in the edit box. Entered character keys also position the list box if matches occur.

**Push Buttons** - Push buttons are controls which usually allow you to specify how you wish to proceed with the dialog. To select a push button, move the mouse pointer over the push button and click. Alternatively, you can usually tab to the push button and then press the space bar. The default push button selection will generally be shaded slightly darker around the edges. To use the default button on any dialog box, press the **Enter** key.

# Workstation Administration

Overview

Starting the Workstation Administration Tool

Creating an LPLR

Opening an LPLR for Update

Updating an LPLR

Workstation User Administration



## **Workstation Administration - Overview**

Configuration Management is the process of controlling the way things are created and maintained in the Zeidon repository. The repository is the database, on a server, which contains all the Zeidon application components. Components are organized on the repository into Central Project Library Releases (CPLR). Development work on components is not done on the repository but on a user's workstation.

Workstation Administration is the process of establishing the Zeidon workstations, specifying workstation users, and identifying and maintaining Local Project Library Releases (LPLRs) including the compilation of LPLR source files. LPLRs are where changes are made to the application components of a corresponding CPLR. New components may be created on an LPLR but, before an existing component may be changed, it must first be checked out from the CPLR. After changes and/or additions are completed and tested they are checked in to the CPLR.

## Starting the Workstation Administration Tool

To start the Workstation Administration tool, double-click on the Workstation Mgr icon in the Zeidon program group.

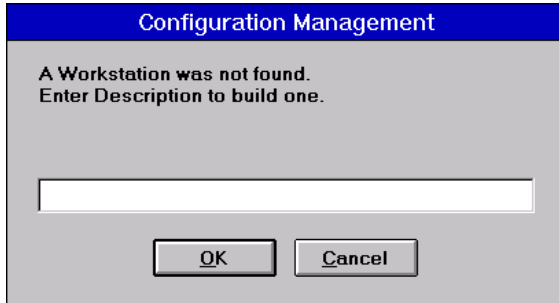
### **See also:**

[First Time Entering the Tool](#)

[Subsequent Entries into the Tool](#)

## First Time Entering the Tool

If this is the first time using Zeidon on your workstation, you will receive the following prompt:



Enter a description of your workstation in edit box provided and press the **OK** button.

**NOTE:** After your workstation has been successfully set up you should never again see the dialog box indicating that the Repository Client was not found on this workstation. If you do, contact your Zeidon System Administrator to identify the problem.

---

Selecting **OK** presents a Set Up Workstation dialog box. If not, a dialog box will inform you of the possible reasons why, in which case you may need to check with the Zeidon System Administrator to fix the problem.

To set up your workstation:

1. Select your User Name from the drop-down combo box.
2. Enter your password in the text box provided.  
NOTE: you will probably need to obtain your password from the Zeidon System Administrator.
3. Select **OK**.

If your password is valid, the New LPLR dialog box will be presented.

## Subsequent Entries into the Tool

Once your workstation has been initialized, subsequent entries into the Workstation Administration tool will present a Welcome to Zeidon - Sign On dialog box.

A screenshot of a Windows-style dialog box titled "Welcome to Zeidon - Sign On". The dialog has a blue title bar. Inside, there are two input fields: "Workstation User:" which is a drop-down menu with a blue background and a small downward arrow, and "Password:" which is a standard text box. To the right of these fields are two buttons: "OK" and "Cancel".

1. Select your user name from the workstation drop-down combo box.  
NOTE: more than one user may register themselves to a workstation.
2. Enter your password in the text box provided.
3. Select **OK**.

The Open Local Project Library Release window is presented if any LPLR's are defined on your workstation, otherwise a New LPLR dialog box is presented.

# Creating an LPLR

To create a new LPLR:

1. Either push the **New** button in the Open Local Project Library Release window or from the Local Project Library Release window, select the **New** item from the **File** pull-down menu. In either case, the New Local Project Library Release dialog box will be presented.

2. Enter a name and an optional description in the text boxes provided.  
NOTE: the name may be up to eight characters and must follow the rules for proper filename naming.
3. In the Executable Directory` text box, enter the directory where the application executables (i.e., XODs, XWDx and DLLs ) are to be stored. For example, D:\LPLRNAME\EXEC.
4. In the Component Source Directory text box, enter the directory where the Zeidon application Components (i.e., LODs, PWDs, PMD) are to be stored. For example, D:\LPLRNAME\META.
5. In the VML /C Source Directory text box, enter the directory where the application source files (i.e. VML modules, C modules, header files, etc.) are to be stored. For example, D:\LPLRNAME\SRC.  
NOTE: the three directories above are usually the same since it is not necessary to separate the file types.
6. In the Repository Type control box, use the radio buttons to select the type of the LPLR. The choices are:  
  
**Repository LPLR** - A repository LPLR is created relative to an existing CPLR in the repository. The next step after creating a repository LPLR is to refresh from the CPLR to initialize the LPLR with components from the CPLR.  
  
**Non-Repository LPLR** - A non-repository LPLR is not related to any existing CPLR from the repository. Non-repository LPLRs exist only on your local workstation and cannot be checked in to the repository. Usually, the next step after creating a non-repository LPLR is to initialize it by migrating in components from one or more existing LPLRs.  
  
**External LPLR** - An external LPLR is one that already exists on the workstation but is not yet known by the Workstation Manager tool. Usually this an LPLR from an earlier release of the Zeidon system or from a different workstation whose LPLR directory has been copied to this workstation.
7. From the CPLR for Repository LPLR Only drop-down combo box, select the CPLR for which this LPLR is being created to do development for. This list includes any CPLR in the Zeidon Repository for which you have been assigned access authority. If the list is empty, you should contact the Administrator of the CPLR you wish to work with.

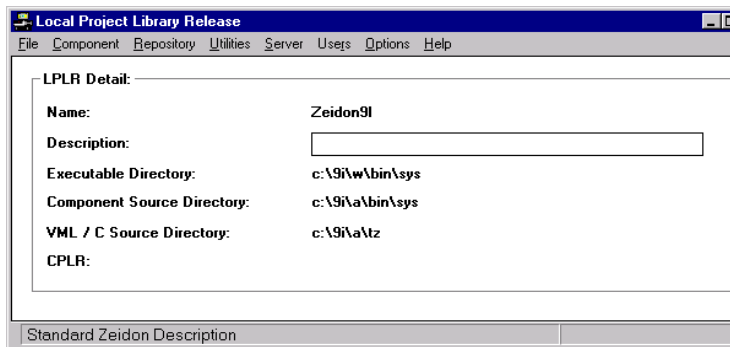
8. When you have done all of the above, select **OK** to create the new LPLR or select **Cancel** to abort.

## Opening an LPLR for Update

Open an LPLR from the Open Local Project Library Release dialog box by either double-clicking on the desired LPLR in the list or by highlighting the LPLR and pressing **OK**. You may get to this dialog box from the Local Project Library Release window by selecting the **Open** item from the **File** drop-down menu. If LPLRs are defined for your workstation, this dialog box is presented upon entering the tool.

## Updating an LPLR

An LPLR is maintained in the Local Project Library Release window. Opening this window is described in [Opening an LPLR for Update](#).



Contents of the LPLR may be modified by the following functions:

**Refreshing from the CPLR** - This process compares all the application components in your LPLR with those in the corresponding CPLR and the reusable CPLRs associated with it, and refreshes (i.e. copies) any application components from the CPLR which are newer or do not exist on the LPLR. Use this process to initialize your LPLR and when you need to get in sync with any changes which may have occurred to the CPLR.

**Checking Out Components** - This process provides your LPLR with update capability for the application component checked out. It also en queues that component on the CPLR so that no other user may work on it until you are finished modifying it and have checked it back in. The only modifiable components on the LPLR are those which you have checked out or those which you have created as new.

**Checking In Components** - This process involves the committing of a new or checked out component back to the repository. This will free the en queue on the CPLR and remove update capability for that component on the LPLR.

**Releasing a Check Out** - This process will free the en queue of that component from the CPLR and remove its update capability on the LPLR. It also forces the LPLR to be refreshed from the CPLR and restores the component to its pre-modified state.

**Removing the Active Status** - This process will remove a component's update capability and return it to its pre-modified state. This function is to be used if the Administrator of the corresponding CPLR removed the en queue for that component via a CPLR administration utility.

Each of these functions are discussed in greater detail in the following related sections.

### See also:

[Refreshing an LPLR](#)

[Checking Out Components](#)

[Checking In Components](#)

[Removing a Check Out](#)

[Listing Components](#)

[Compiler and Target Specifications](#)

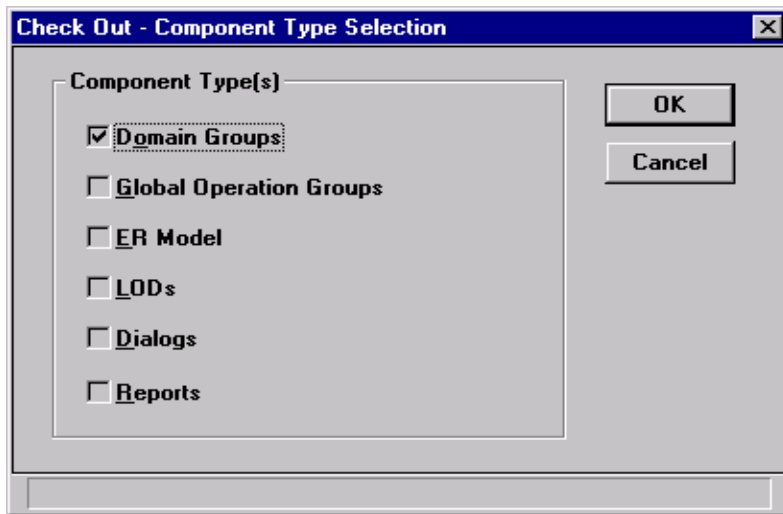


## Refreshing an LPLR

From the **Utilities** drop-down menu of the Local Project Library Release window, select the **Refresh From CPLR** item.

## Checking Out Components

From the **Component** drop-down menu of the Local Project Library Release window, select the **Check Out** item. A Check Out Component Type Selection dialog box is presented.



Click on the Component Type(s) you wish to check out and press **OK**. A Component Check Out dialog box is presented with a list of all components of the type(s) selected.

Single-click on the component desired for check out. If the component is already checked out the **Check Out** button is grayed and the **Details** button is enabled. You may press the **Details** button to view information about who and when the component was checked out. If the component is a candidate for check out, press the **Check Out** button. Key in any comments you may wish to record about your checking out of this component in the Verify Check Out dialog box presented (shown below) and select **OK**.

Verify Check Out

CPLR: Auftrag1

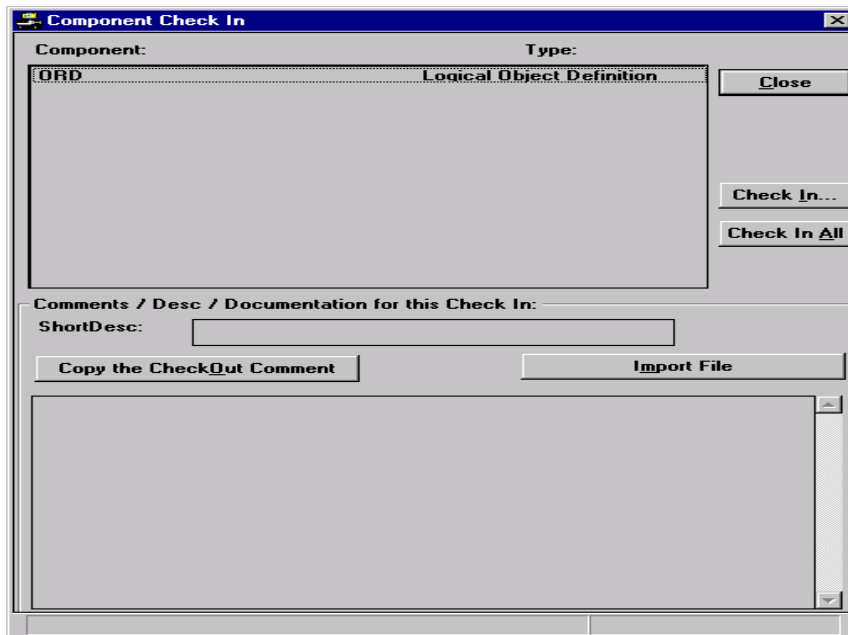
Comments:

OK

Cancel

## Checking In Components

From the **Component** drop-down menu of the Local Project Library Release window, select the **Check In** item. A Component Check In dialog box is presented listing all candidates for check in from your LPLR.



You can select a single or multiple components to be checked in and press the **Check In** button or, if you wish to check in all components listed, press the **Check In All** button. If the list presented to you has more than one component and you did not select all for check in, the tool will determine if the member(s) you've selected may be checked in or, if due to inter-component dependencies, some of the non-selected components would be required to be checked in also. If that is the case, you will receive a message indicating so and you will be shown the Component Check In dialog box showing the minimum list of components which must be checked in based on your previous selection. Your choices now are to **Check In All** or **Cancel**.

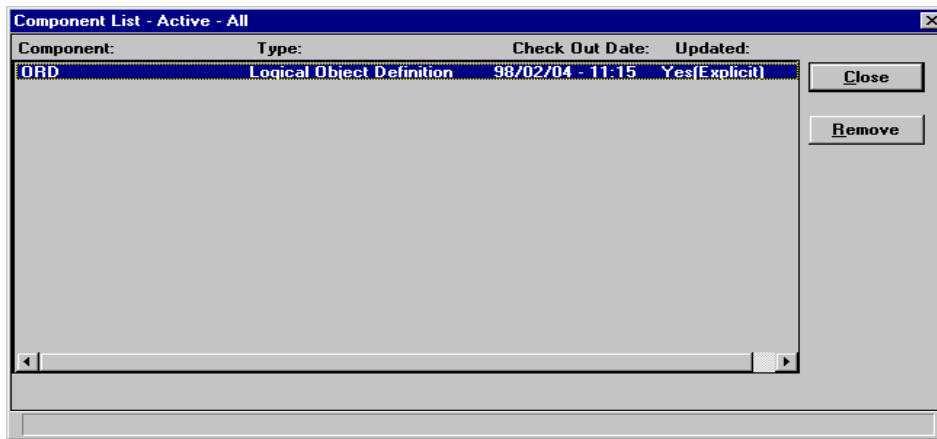
Continuing with the check in, you may now see a Configuration Management message letting you know that the backup is in progress. Please wait until it finishes before selecting **OK**. There will also be a minimized Backup icon. When that icon disappears, press the **OK** button. The tool will refresh the LPLR as a part of the check in process. You may get an idea of which components are being refreshed by viewing the DIL messages at the bottom of the dialog box. If, during that process, your LPLR has no new or updated components copied from the CPLR, check in will proceed and, upon completion, return you to the Local Project Library Release window. Otherwise, you will receive a warning message prompting you as to whether you would like to view the Refresh results, cancel the check in, or continue. Select **Check In** only if you are confident that what you are checking in will be compatible with what has been refreshed to your LPLR, otherwise you could be putting bad data onto the CPLR. If you are not sure, you may wish to review the refresh report and perhaps test your changes, and then run check in again.

## Removing a Check Out

A component is considered active when it has either been checked out from the repository or created new in the LPLR. An active component can be modified and checked into the repository. It is sometimes desirable to remove the active status of a component. This will remove the component from the LPLR and alter the check out status of the component in the repository, allowing another LPLR to check out that component.

If you remove the active status of a component that has been checked out from the repository, you must do a refresh from the repository to get the repository's most recent version of the component.

To remove a components active status, select the **Component List** item from the **Component** pull-down menu of the Local Project Library Release window. This will present the Component List dialog box.



Single-click on the desired component in the list box and press the **Remove** button or press **Cancel**.

## Listing Components

Components of an LPLR may be listed according to whether the user has them as Active (updatable), Refreshable (new or newer on the server), or Checked Out. From the **Options** pull-down menu of the Local Project Library Release window, select the **Component Criteria** item. The Component List Options dialog box is presented. This dialog box is used in conjunction with the **List** item of the **Components** drop-down menu. Select one option from the Component List criteria group, click on the All, Active, or New radio buttons and select the Component Type(s) to filter the list further. Press **OK**. Select **List** from the **Components** drop-down menu to see a Component List dialog box of your specified criteria.

## Compiler and Target Specifications

To compile source files created in the Zeidon tools, two things need to be defined: a Compiler Specification and a Target Specification.

A compiler specification contains all of the necessary compiler and local workstation information to compile Zeidon source. Compiler specification's are defined for every LPLR on each workstation. Each LPLR can have multiple compiler specification's although only one can be used at a time.

The target specification is the list of all executables files that are made up of the Zeidon source files. There is a target specification for every LPLR on each workstation. The target specification lists each of the target executables (e.g. .DLLs in MS-Windows) and all of the Zeidon components (e.g. Dialogs, Object Definitions) that make up the target executable.

Once a compiler specification and a target specification have been created, source files can be compiled from either the target specification window or from the editor.

### **See also:**

[Creating and Updating Compiler Specifications](#)

[Selecting a Current Compiler Specification](#)

[Target Specification](#)

[Make Options](#)

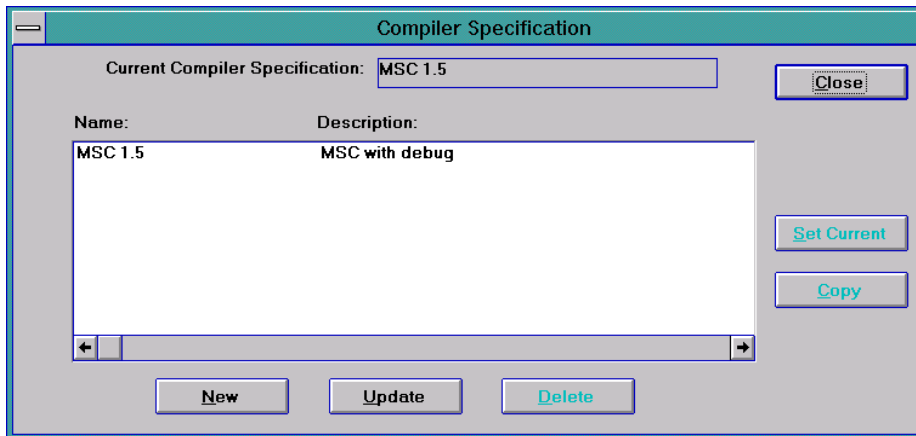
[Making Targets](#)

## Creating and Updating Compiler Specifications

A compiler specification can be created from one of two places:

- From the main window in the Work Station Manager dialog, select the **Options | Compiler Specification** menu option.
- From the editor, select the **Zeidon | Compiler Specification** menu option.

In either case, the following window will be presented:



To update an existing compiler specification, highlight it in the list box and press the **Update** button. To create a new specification, press the **New** button. In either case, the following window is presented:

The following fields may be updated:

**Compiler Specification Name** - Name of this compiler specification. This can contain blanks, etc.

**Description** - Description for this compiler specification.

**Target Executable Directory** - The directory that will contain the final target executables such as the dialog



DLLs.

**OBJ Directory** - The directory that will contain the intermediate .OBJ files.

**Makefile Directory** - The directory that will contain all the makefiles created by Zeidon.

**Resource Directory** - The directory that will contain the executable resource files (e.g. for MS-Windows, this includes .RC and .BMP files).

**Compiler Directory** - The directory contain the compiler executable files.

**Compiler** - Use the radio buttons to select from the list of all compilers currently supported by Zeidon.

**Debug Compile Options** - Determines if debug information is to be created. Options are:

**Debug OFF** - No debug information.

**Debug ON** - Executables compiled with debug information.

**Debug determined by env var** - The debug option is determined by the environment variable DEBUG. If this variable is set (i.e. non-null) then the executable is compiled with debug information. If DEBUG is null, then no debug information is created.

Once this information has been specified, click on the **INC/LIB...** button. This brings up the following window:

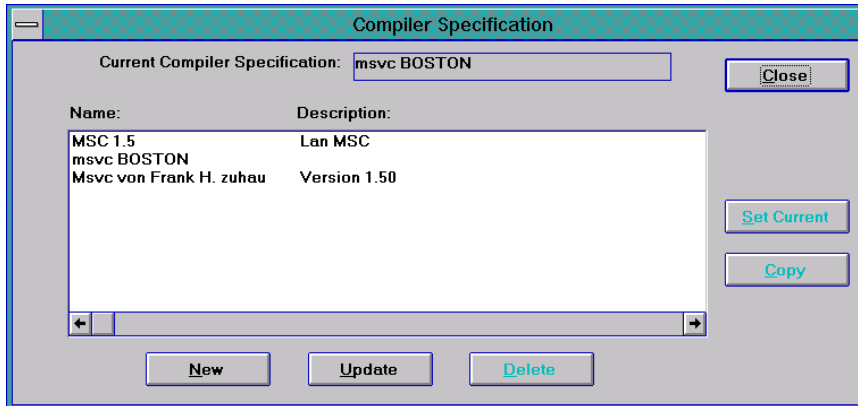
On this window you must specify all the directories that contain C header files and .LIB files needed for linking. The directories that **MUST** be specified here are the include and lib directories for Zeidon and the compiler. You only need to have more if you use third-party tools in your Zeidon source files.

Since the resolving of .H and .LIB files is dependent on the order of the directories, you may use the arrow buttons to re-arrange the order of the directories.

When you are done, press **Close** to return to the Edit Compiler Specification dialog. In the Edit Compiler Specification window, press **OK** to accept the changes to the specification or press **Cancel**.

## Selecting a Current Compiler Specification

Every LPLR must have one compiler specification identified as its current specification. From the Compiler Specification dialog, select the desired specification by single-clicking on it in the list box then click on the **Set Current** button. The selected compiler specification should then appear at the top of the window as in the example below:



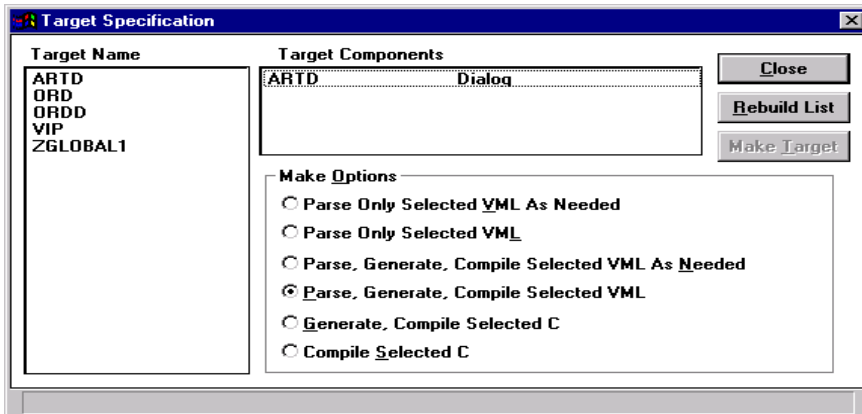
Once you have selected a compiler specification as current, all compiles will use that specification until it is changed.

## Target Specification

A target specification can be created from one of two places:

- From the main window in the Work Station Manager dialog, select the menu option **Options | Target Specification**.
- From the editor, select the menu option **Zeidon | Target Specification**.

In either case, the following window will be presented:



Note that the listboxes Target Name and Target Components will be empty the first time this window is opened. Click on **Rebuild List** to build the list of target executables.

The Target Name listbox contains all target executable names (e.g. the DLL name for MS-Windows). This list is created by checking each Zeidon component for operations. If it has operations, then the target executable for that component is added here. The Target Components listbox is a list of all components that make up the target executable.

Any time you create a new Zeidon component with operations, you must rebuild the list of target executables to be able to compile the source created with the new component.

## Make Options

The Target Specification window allows you specify what kind of processing will be done when a make is initiated. Making a target will invoke the compiler specified in the current compiler specification. One of the following make options must be selected from the corresponding radio buttons:

**Parse Only VML As Needed** - Each VML source file for the target will be parsed if it has been changed since the last parse. However, the corresponding C files will not be generated nor will they be compiled.

**Parse Only Selected VML** - Each VML source file for the target will be parsed regardless of whether or not it has been changed since the last parse. The corresponding C files will not be generated nor will they be compiled.

**Parse, Generate, Compile As Needed** - Each VML source file for the target will be parsed if it has been changed since the last parse. All C files derived from VML, for the selected target(s), will be generated as needed. And all C files will be compiled if they have changed since the last compile.

**Parse, Generate, Compile Selected VML** - Each VML source file for the target will be parsed regardless of whether or not it has been changed since the last parse. All C files derived from VML, for the selected target(s), will be generated. All C files will be compiled.

**Generate, Compile Selected C** - Each VML source file for the target will be parsed if it has been changed since the last parse. All C files derived from VML, for the selected target(s), will be generated. All C files will be compiled.

**Compile Selected C** - Each VML source file for the target will be parsed if it has been changed since the last parse. All C files derived from VML, for the selected target(s), will be generated as needed. All C files will be compiled.

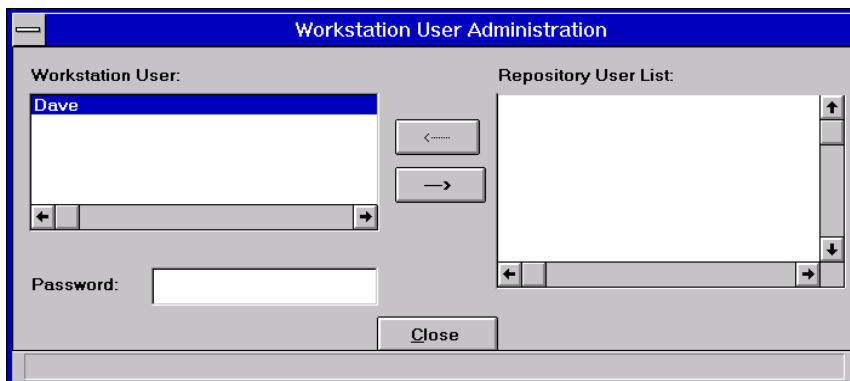
## Making Targets

You can make one or more targets for the LPLR by selecting one or more entries in the list and pressing the **Make Target** button. If a make option has been selected that will call for a compile to be initiated, a make file will then be created for each target. A DOS box will automatically be opened and the make files will be executed (again, automatically).

When making a single target, processing will stop whenever an error is encountered. When making all targets any error will stop the processing for the current target but processing will continue with the next target.

## Workstation User Administration

From the **User** drop-down menu of the Local Project Library Release window, select the **Administer User** item. The Workstation User Administration dialog box is presented.



The dialog box has two lists, one of users currently assigned to the workstation, and one of all users assigned to the Zeidon repository.

### See also:

[Adding Users to the Workstation](#)

[Removing Users from the Workstation](#)

## **Adding Users to the Workstation**

To add a user to the workstation, simply:

1. Select the user to be assigned by clicking on the desired user in the Repository User List list box.
2. In the Password edit control enter the user's password.
3. Press the Arrow button ( <- ) pointing to the Workstation User list box.

## **Removing Users from the Workstation**

To remove a user from the workstation, simply:

1. Select the user to be removed by clicking on the desired user in the Workstation User list box.
2. Press the Arrow button ( -> ) pointing to the Repository User List list box.



# **Data Model**

**Overview**

**Starting the Data Model Tool**

**Viewing a Data Model**

**Creating a Data Model**

**Opening a Data Model for Update**

**Entities**

**Relationships**

**Attributes**

**Identifiers**

**Subject Areas**

**Importing External Data Models**

**Printing the Data Model**

**Genkeys and Dynamic Table Domains**

## **Data Model - Overview**

The Data Model tool is used to create and maintain the logical data model component of an application. The primary element in a data model is an entity. An entity is a discrete class or set of data items about which an organization stores information. Each entity in a data model is made up of one or more attributes which describe the entity. Attributes can be thought of as the specific data fields for the entity. Entities are linked to each other via relationships. A relationship can represent either a physical or conceptual connection between two entities. An identifier is a characteristic of an entity that uniquely identifies each instance of that entity in the physical database. Identifiers are similar to the concept of "keys" in traditional data processing.

The Data Model tool allows you to view and maintain the model in several modes, some of which are graphical. The tool also allows you to import data models from some external data modeling tools such as ADW.

## **Starting the Data Model Tool**

To start the Data Model tool, double-click on the Data Model icon in the Zeidon program group. If a data model exists for the application, it will be presented automatically when the tool is started. Otherwise, you will be presented with a window in which to create a data model.

## Viewing a Data Model

The data model can be viewed in three basic ways:

**Data Model window** - Provided that a data model already exists for the current application LPLR, the data model will automatically be displayed graphically in the Data Model window.

**Entity List window** - This window displays the data model in a list format. It is opened by selecting the **Entity List** item from the **View** pull-down menu in the Data Model window.

**Subject Areas** - For data models with a large number of entities, viewing the entire data model at one time may be unwieldy. Subject areas allow you to view a subset of the data model. Subject areas can be displayed in either a graphical or list format.

**NOTE:** the Data Model window is the primary window for the tool and it is always open. Changes made in the Entity List window or in a Subject Area window will automatically be reflected in the Data Model window.

---

## Creating a Data Model

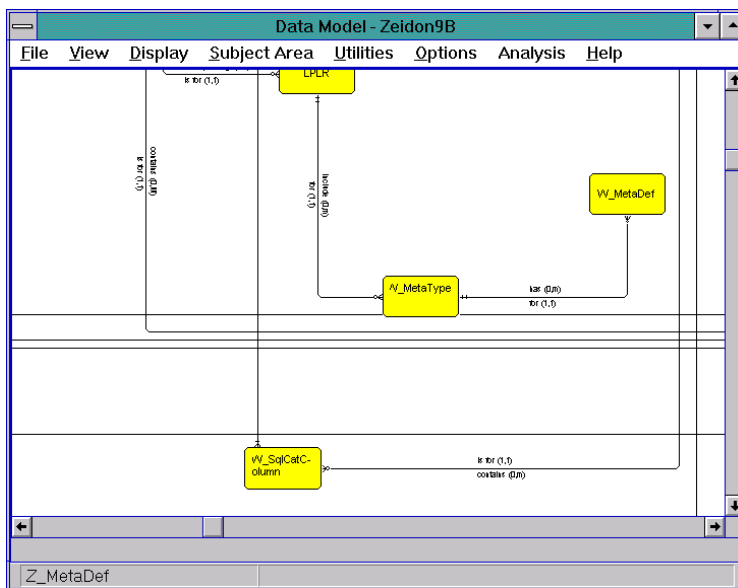
Only one data model may exist for an LPLR. Therefore, a data model may only be created when one doesn't already exist for the LPLR. To create the data model for an LPLR, enter the Data Model tool for an LPLR which does not have a data model. A New Model dialog box will be presented in which you can enter an optional description for the data model. Selecting the **OK** button brings you to the Data Model window of the newly created model. One initialized entity is created for you at this time.

## Opening a Data Model for Update

Upon entering the Data Model tool, the model for the LPLR is automatically opened for update. If no model exists for the LPLR, a window is presented which allows you to create the data model. Note that the model will not be updateable unless it has been checked out to your workstation using the Workstation Administration tool or you have created it on your workstation and have not yet checked it into the repository.

To open a data model for an LPLR other than the current one, select **Open LPLR** from the **File** pull-down menu. This will present the Open Local Project Library Release dialog from which you can select another LPLR by single-clicking on it in the list box and pressing **OK**.

In any event, the following Data Model window will be presented:



From this window, you can update the major components of the data model including its subject areas, entities, relationships (between entities), attributes (of entities) and identifiers (for entities).

# Entities

An entity is a discrete class or set of data items about which an organization stores information.

## See also:

[Viewing Entities](#)

[Creating an Entity](#)

[Updating an Entity](#)

[Deleting an Entity](#)

## Viewing Entities

Entities can be viewed in a number of ways while in the Data Model tool.

- The data model will always be visible in a graphical format in the Data Model window. In this window, entities are displayed as boxes.
- To view the entities in a list format, select the **Entity List** option from the **View** pull-down menu of the Data Model window.

Regardless of how the data model is viewed, one entity at a time is considered to be the *current entity*. If the data model is being viewed graphically, the current entity is highlighted in red. In a list format, the current entity name is highlighted in the Entity list box. In both cases, single-clicking on an entity will highlight it and make it the new current entity.

The current entity has focus with respect to any entity specific actions that you might take. For example, if the Attribute List window is open (discussed in greater detail later), it will display attribute information only about the current entity. Changing the current entity in the Data Model or Entity List window, will automatically change what is displayed in the Attribute List window.



## Creating an Entity

The information required to create a new entity is specified in the Entity Create dialog. This window can be opened in a variety of ways including:

- In the Data Model window, paint an entity into the diagram by holding down the left mouse button and dragging to form a box where you want the entity placed. An Entity Create dialog box is presented.
- Alternatively, in the Data Model window, right mouse click while the cursor is positioned on an entity. Select the **New** option from the pop-up menu. This will open the Entity Create dialog.
- From the Entity List window, right mouse click on the Entity List dialog box and select the **New** option from the **Entity List** pop-up menu. This will open the Entity Create dialog also.

Once the Entity Create dialog has been opened, enter a name for the entity and an optional description in the text boxes provided. From the Purpose drop-down combo list, select one of the following:

**Fundamental** - An entity that exists and is of interest in its own right.

**Attributive** - An entity that exists only to describe another entity.

**Associative** - An entity that exists primarily to interrelate other types.

**Work** - An entity that is non-persistent on the data base.

If you wish, you may also specify other details for the newly created entity. If so, press the **More>>>** button to open the Entity Detail dialog. The fields in this dialog are described in detail in [Updating an Entity](#).

## Updating an Entity

Entities are updated from the Entity Detail dialog box. You get to this dialog box in one of the following ways:

- Double-click on the desired entity in the Data Model window.
- Double-click on the desired entity in the Entity List dialog box.
- With the desired entity highlighted in either the Entity List dialog box or on one the diagrams, right mouse click and select the **Entity Detail** option from the pop-up menu.
- In the Entity Create dialog, press the **More>>>** button.

In any case, the following Entity Detail window is presented.

The screenshot shows the 'Entity Detail' dialog box. The 'Name' field contains 'VIP'. The 'Desc' field contains 'Kundenstammdatei'. The 'Purpose' dropdown is set to 'Fundamental'. The 'TE Table Name' and 'TE Table Owner' fields are empty. The 'Relationship' field contains 'Kunde bestellt' and the 'Target Entity' field contains 'ORD'. The 'Attribute' list contains 'NUM', 'NAME', 'CLUB', 'STR', 'NAT', and 'LOC'. The 'Identifier' list contains 'NUM'. On the right side, there are buttons for 'Close', 'Next', 'Prev', 'New...', and 'Delete'.

In the Entity Detail dialog, the following items may be specified:

**Name** - Enter a name to be used to identify the entity in the corresponding text box.

**Description** - An optional description can be entered in the text box provided.

**Purpose** - Using the drop-down combo box, define the type of entity. Choose one of the following:

**Fundamental** - An entity that exists and is of interest in its own right.

**Attributive** - An entity that exists only to describe another entity.

**Associative** - An entity that exists primarily to interrelate other entity types.

**Work** - An entity that is non-persistent on the data base.

**TE Table Name** - The table name in the TE to be generated from this entity. If no name is entered, a default name will be generated, which will be as close to the Entity Name as possible within the length limitation of

the table name.

**TE Table Owner** - The table owner name in the TE to be generated from this entity. If no name is entered, no owner name will be created in the TE for the Entity.

**Relationship** - Relationships can be added, updated or deleted. With the cursor in the Relationship list box, single-click the right mouse button to bring up a pop-up menu. Maintaining data model relationships is discussed in greater detail in the section [Relationships](#).

**Attribute** - Attributes can be added, updated or deleted. With the cursor in the Attribute list box, single-click the right mouse button to bring up a pop-up menu. Maintaining attributes is discussed in greater detail in the section [Attributes](#).

**Identifier** - As is the case with Relationships and Attributes, identifiers can be maintained by right mouse clicking in the associated list box to produce a pop-up menu. Identifiers are discussed in greater detail in the [Identifiers](#) section below.

Once all the entity details have been modified, press **Close** to accept the changes. If you are modifying several entities at a time, you can cycle through all of the entities for the data model (or subject area) by pressing the **Next** or **Prev** buttons.

## Deleting an Entity

Delete an entity in one of the following ways:

- In the Data Model diagram, highlight the entity to be deleted by single-clicking on it. Then, with the cursor on the entity, click the right mouse button. From the pop-up menu select the **Delete Selected** option.
- In the Entity List dialog box, highlight the entity to be deleted by clicking on it and then click the right mouse button while the cursor is still in the entity list box. From the pop-up menu select the **Delete** option.
- From the Entity Detail dialog box, press the **Delete** button to delete the entity currently displayed.

## Relationships

Entities are linked to other entities via relationships. A relationship can represent either a physical or conceptual connection between two entities. A relationship between two entities in the data model captures information about the possible relationships of the corresponding entity instances in the database. For example, instances of entities may be related to each other in a one to one or a one to many manner.

### See also:

[Viewing Relationships](#)

[Creating a Relationship](#)

[Updating a Relationship](#)

[Deleting a Relationship](#)

## Viewing Relationships

How we view and manipulate the relationships between entities depends on whether or not we are viewing the data model in a diagram or list format. For more information, please refer to the following related sections.

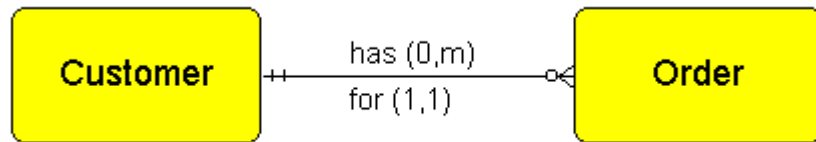
### See also:

[Viewing Relationships in a Diagram](#)

[Viewing Relationships in a List](#)

## Viewing Relationships in a Diagram

When a data model is viewed graphically, as it is in the Data Model window, relationships between entities are generally displayed as labeled lines connecting the entities. For example, consider the following diagram.



Here, the entities Customer and Order are related. There are several aspects of this relationship and its representation in the diagram that we should take note of. First, the relationship is two-way. That is, the line in the Data Model diagram connecting the two entities actually represents two separate relationships: the relationship from Customer to Order and conversely, the relationship from Order to Customer. This dual aspect of the relationship is represented by the two labels on the line. A Customer "has" Orders while an Order exists "for" a Customer.

Secondly, *cardinalities* are the number of instances that may occur between two related entities. The minimum and maximum cardinalities for each uni-directional relationship are shown in the diagram as ordered pairs. In our example above, a Customer can "have" anywhere from zero to many Orders. This is denoted by the ordered pair (0,m). On the other hand, an Order can be "for" exactly one Customer. The minimum and maximum for that relationship are both 1 as is denoted by (1,1) in the diagram. Note, that this implies that an instance of the entity Order can not exist independent from the existence of an instance of a Customer entity.

Finally, we should note that relationships may be displayed without the corresponding labels. If this is the case, some information about the cardinalities can be understood from the form of the line connecting the entities in the diagram. Relationships with upper bounds greater than one are denoted with a crow's foot symbol. For example, the relationship "Customer has Order" terminates in the Order entity with a crow's foot, indicating that the maximum for this relation is greater than one. On the other hand, the relation from Order to Customer does not terminate in the Customer entity with a crow's foot denoting that the maximum in that direction is one.

How relationships are shown is controlled by selecting the **Diagram** option from the **Options** menu. This opens the Diagram Options dialog. In the Draw Relationship group, select either the Line and Text or Line radio button and press **OK**.

## Viewing Relationships in a List

When a data model is viewed in a list format, the relationships for the current (highlighted) entity are listed in the Relationship list box. The label and target entity relative to the current entity are shown. To see other relationship details such as cardinalities and backwards relationships, double-click on the relationship in the list box to open the Relationship Detail dialog. This dialog is discussed in greater detail in [Updating a Relationship](#).



## Creating a Relationship

Relationships between entities may be created in a number of ways:

- From the Data Model window click on the source entity. Then, right button drag from that entity to the entity you want it related to. The entity you drag to becomes the target entity.
- Bring up an Entity Detail dialog box by either double-clicking on the desired source entity in the Data Model window, or on the entity in the Entity List dialog box. Right click in the Relationship list box and select the **New** option from the pop-up menu.
- Select **Relationship List** or **Entity List** item from the **View** pull-down menu in the Data Model window. Identify the source by clicking on the desired entity in the Data Model window, or on the entity in the Entity List dialog box. With the cursor positioned in the Relationship list box of either the Entity List or Relationship List dialog box, right mouse click and select the **New** option from the pop-up menu presented.

In any event, the Relationship Detail dialog is presented. The section, [Updating a Relationship](#) describes in detail how to complete the fields of this dialog.

## Updating a Relationship

Relationships between entities are updated in the Relationship Detail dialog box. You can bring up this dialog box in a number of ways:

- In the Data Model window, double-click on the relationship line you wish to update.
- Bring up an Entity Detail dialog box by either double-clicking on the desired source entity in the Data Model window, or on the entity in the Entity List dialog box. Right click in the Relationship list box and select the **Relationship Detail** option from the pop-up menu.
- From the **View** menu in the Data Model window, select **Relationship List** or **Entity List**. Identify the source by clicking on the desired entity in the Data Model window, or on the entity in the Entity List dialog box. With the cursor positioned in the Relationship list box of either the Entity List or Relationship List dialog box, right mouse click and select the **Relationship Detail** option from the pop-up menu presented.

Relationship Detail

Kunde bestellt

Min: 0 To Max: m

☐ AutoSeq

TE FKey Prefix:

VIP — ORD

bestellt von

Min: 1 To Max: 1

☐ AutoSeq

TE FKey Prefix:

Many-to-Many Relationships

Table Name:

OK

Cancel

Next

Prev

New

Delete

Directional name of the relationship

Once the Relationship Detail dialog is open, update the information as follows:

1. The Source entity is displayed on the left hand side of the dialog. Select the Target entity from the Target Entity drop-down combo box on the right.
2. Describe the relationships and specify the Min to Max cardinalities for both Source and Target Entities. Values for Min may be 0 or a number (usually 1). Values for Max may be a number greater than 0, 1, or m (for many).
3. If selected, AutoSeq will control the order of entities through the relationship by their position you either created them in or you reordered them in.
4. The TE F Key Prefix specifies a prefix that will be added to foreign key names generated for this relationship. Note that there is a prefix for each direction in the relationship because foreign keys depend on the relationship direction.
5. The Many-to-Many Relationships Table Name specifies the name of the table generated for their relationship if it is many-to-many. If no name is specified, one will be automatically generated.
6. Once the relationship has been specified, press **OK** to accept the changes.

If you are updating several relationships for the same source entity, you can cycle through all the relationship for the current source entity by pressing the **Next** and **Prev** buttons in the Relationship Detail dialog.

## Deleting a Relationship

Relationships between entities may be deleted via any of the following:

- While in the Relationship Detail dialog box, press the **Delete** button.
- Select the relationship to be deleted from the relationship list box of the Entity List, Relationship List, or Entity Detail dialog box. Right mouse click and select the **Delete** option from the pop-up menu.
- In the Data Model window, single-click on a relationship to highlight it. With the cursor on the selected relationship, right mouse click and select **Delete Relationship** from the pop-up menu.

See [Updating a Relationship](#) for details on opening the above mentioned dialog boxes.

## Attributes

Each entity in a data model is made up of one or more attributes which describe the entity. Attributes can be thought of as the specific data fields for the entity.

### See also:

[Viewing Attributes](#)

[Creating an Attribute](#)

[Updating an Attribute](#)

[Deleting an Attribute](#)

## Viewing Attributes

When a data model is viewed graphically, attribute information for the current (highlighted) entity is not displayed. There are several ways to access this information. One way is to select the **Attribute List** item from the **View** pull-down menu in the Data Model window. In the Entity List window where the data model is displayed in a list format, the attributes for the current (highlighted) entity will be shown in the Attributes list box.

## Creating an Attribute

Attributes are maintained with the Attribute Detail dialog. This dialog is discussed in detail in the section, [Updating an Attribute](#). You can create a new attribute for an entity and open the Attribute Detail dialog in several ways.

- In the Data Model window, select the entity for which you wish to add an attribute by single-clicking on the entity. Select the **Attribute List** item from the **View** pull-down menu to display the Attribute List window. With the cursor in the Attribute List window, right mouse click and select **New** from the pop-up menu.
- The Entity List dialog box, select the entity for which you wish to create a new attribute by single-clicking on the entity name in the Entity list box. (This dialog box is presented by selecting **Entity List** from the **View** menu of the Data Model window.) With the cursor in the Attribute list box, right mouse click and select **New** from the pop-up menu.
- In the Entity Detail dialog box for the target entity, right mouse click with the cursor in the Attribute list box. Select **New** from the pop-up menu.

## Updating an Attribute

Attributes are updated in the Attribute Detail dialog box. You access this dialog box via a Attribute list box by either double-clicking on the desired attribute or by selecting the desired attribute with a single-click and then right mouse clicking to present a pop-up menu. From the pop-up menu select **Attribute Detail**. Attribute list boxes are found in the following dialog boxes:

- The Attribute List dialog box.
- The Entity Detail dialog box.
- The Entity List dialog box.

Refer to [Creating an Attribute](#) for details on presenting the above dialog boxes. In any event, the following Attribute Detail window is presented.

Modifiable fields of attributes include the following:

**Name** - Name is used to identify the attribute. Enter the name in the text box provided.  
NOTE: every attribute must be assigned a name.

**Description** - You can key in an optional description in the text box.

**Domain** - Select a domain from the drop-down combo box.  
NOTE: every attribute must be assigned a domain.

**Length** - For attributes with string data type domains, a length may be specified to override the length specification of the domain.

**Column Name for Attribute** - The column name in the TE to be generated from this Attribute. If no name is entered, a default name will be generated, which will be as close to the Attribute Name as possible within the length limitation of the column name.

**Fkey Suffix for Relationship** - A suffix to be added to the Relationship Prefix for generating foreign keys from this Attribute. If no name is entered, default names will be generated.



**Not Null** - Select this check box if data must be provided for this attribute.

**Case Sensitive** - For attributes with string data type domains, this check box indicates whether or not to enforce upper/lower case sensitivity.

**Prompt** - This Auto Design field is used to enter the text you want to identify this attribute when it is displayed on a window or dialog box created through Auto Design. Do not enclose the text in quotes unless you wish the quotes painted as part of the identifying text. If no text is entered, the default is to use the attribute name for Auto Design.

**Heading** - This Auto Design field is used to enter the text you want to identify this attribute when it is displayed in a list box created through Auto Design. The entered text will be the column heading for that attribute on the list box.

When the fields in the Attribute Detail dialog have been completed, press the **OK** button to accept the changes or press **Cancel** to abort. If you are updating several attributes for the same entity at the same time, you can cycle through all of the attributes for the entity from the Attribute Detail window by pressing the **Next** and **Prev** buttons.

## Deleting an Attribute

Attributes for an entity are deleted in one of two ways:

- In the Attribute Detail dialog box for the attribute to be deleted, press the **Delete** button. For more details on opening the Attribute Detail dialog, refer to [Updating an Attribute](#).
- In any Attribute list box, highlight the desired attribute by single-clicking on it. Then, with the cursor still in the Attribute list box, right click the mouse and select the **Delete** option from the pop-up menu. Attribute list boxes are in the Attribute, Entity Detail and Entity List dialogs.

Refer to [Creating an Attribute](#) for details on opening these dialog boxes.

## Identifiers

An identifier is a characteristic of an entity that uniquely identifies each instance of that entity in a physical database. Identifiers are similar to the concept of "keys" in traditional data processing. In fact, they will actually create "keys" and "foreign keys" when the physical database is a relational database and are likely to create "indexes" in any other physical system. Although some physical implementations of a data base will not require you to create identifiers, typically you must create an identifier for each entity in your data model.

You can specify that an identifier be completely defined and maintained by Zeidon. Or, you can specify that an identifier be defined by one or more attributes or even by a combination of attributes and relationships.

**NOTE:** although an identifier may be defined in terms of several attributes, it is important to note that an entity can have only one identifier.

---

### See also:

[Viewing Identifiers](#)

[Creating an Identifier](#)

[Updating an Identifier](#)

[Deleting an Identifier](#)

## Viewing Identifiers

To view the identifier for the current entity, select the **Identifier List** item from the **View** pull-down menu in the Data Model window. This will open the Identifier List window in which the identifier for the current entity will be displayed.

**NOTE:** do not be misled by the name of this window. An entity can have no more than one identifier.

---

## Creating an Identifier

Identifiers are created in the Identifier Detail dialog box. To create an entity identifier:

1. Select the entity for which you wish to create identifiers for by clicking on the desired entity in the Data Model window, or on the entity in the Entity List dialog box.
2. Right mouse click with the cursor positioned in the Identifier list box and then select the **New** option from the pop-up menu. Identifier list boxes are found in the following dialog boxes:

**Identifier List** - This dialog box is opened by selecting **Identifier List** from the **View** menu of the Data Model window.

**Entity Detail** - Bring up this dialog box by either double-clicking on the desired entity in the Model or Subject Area diagram, or on the entity in the Entity List dialog box.

3. In the Identifier Detail dialog box presented, provide the necessary detail for the identifier and press **OK**.

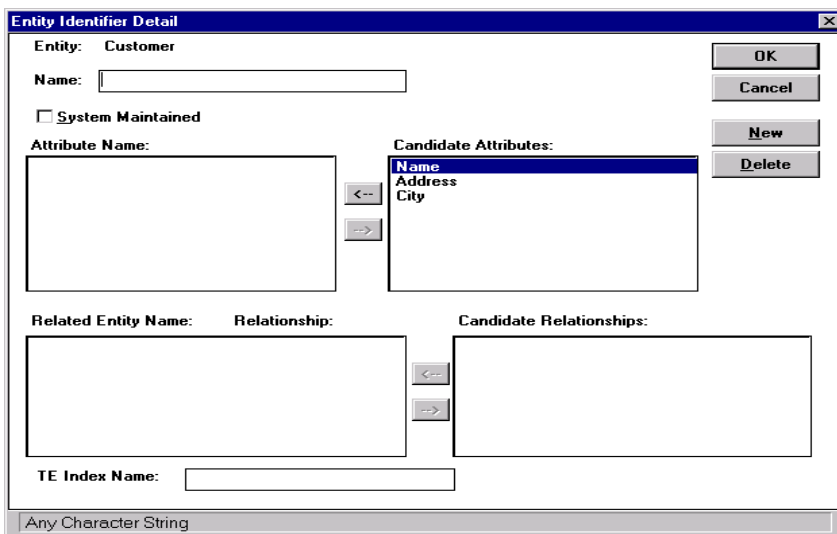
The Identifier Detail dialog is described in further detail in the [Updating an Identifier](#) section.

## Updating an Identifier

Identifiers are updated in the Identifier Detail dialog box. You access this dialog box via an Identifier list box by either double-clicking on the desired identifier or by selecting the desired identifier with a single-click and then right mouse clicking to present a pop-up menu. From the pop-up menu select **Identifier Detail**. Identifier list boxes are in the following dialog boxes:

- The Identifier List dialog box.
- The Entity Detail dialog box.

Refer to [Creating an Identifier](#) for details on opening these dialog boxes. In any case, the following Entity Identifier Detail dialog is presented.



The dialog box is titled "Entity Identifier Detail". It contains the following fields and controls:

- Entity:** A text field with the value "Customer".
- Name:** An empty text field.
- ☐ **System Maintained**
- Attribute Name:** An empty text field.
- Candidate Attributes:** A list box containing "Name", "Address", and "City".
- Related Entity Name:** An empty text field.
- Relationship:** An empty text field.
- Candidate Relationships:** An empty text field.
- TE Index Name:** An empty text field.

Buttons on the right side include "OK", "Cancel", "New", and "Delete". Navigation arrows are present between the Attribute Name and Candidate Attributes fields, and between the Related Entity Name and Candidate Relationships fields. A status bar at the bottom indicates "Any Character String".

There are 5 types of identifiers:

**System Generated/No Attribute Specified** - These are maintained entirely by Zeidon and are not based on any attributes from the entity. You can not control or access the identifier values.

**System Generated/Attribute Specified** - These are maintained by Zeidon but are based on a user specified attribute. The attribute specified must have a domain of type integer and must be defined as being Not Null.

**Automatically Maintained/Relationship Specified** - Some entity instances can be identified through their relationships to other entities. In this case, the identifier is maintained by Zeidon automatically. In relational databases, the relationships will result in foreign keys, the combination of which will uniquely identify an entity instance.

**Application Maintained/Attribute Specified** - These are maintained by the application and are either entered by an operator via a dialog or created by application code. They are defined by one or more attributes, each of which must be defined as being Not Null.

**Attribute and Relationship Specified** - Sometimes a combination of attributes and relationships are required to uniquely identify an entity instance.

System maintained identifiers automatically generate values in the physical database and are maintained by Zeidon for the application. For relational databases, a column within the table for the entity will be created.

The key value for that column will be generated automatically when an instance of the table is created.

The following are modifiable fields describing identifiers:

**Name** - This required field will be the key named used in Zeidon data base queries.

**System Generated** - Select this check box if the key value will be system generated via an algorithm. If an identifier is system maintained and no attribute have been selected, the identifier will be hidden from view by the application LPLR.

**Attribute Name** - Select the attribute or attributes which comprise the key from the Candidate Attributes list box by single-clicking on the desired attribute and then pressing the arrow button pointing to the Attribute Name list box. De-select attributes by clicking on them in the Attribute Name list box and pressing the arrow button pointing back to the list box of Candidate Attributes.

NOTE: if the identifier has been defined as being system generated, you can only select one attribute. Further, that attribute must be defined as an integer domain and be defined as Not-Null.

---

**Related Entity Name** - If one or more relationships are necessary to uniquely identify an entity instance, select the relationships using the Candidates list box. Select a related entity from the Candidates list-box by single-clicking on it and then pressing the arrow button pointing to the Related Entity Name list-box. To de-select a related entity, select it by single-clicking on it in the Related Entity Name list-box and then pressing the arrow button pointing back to the Candidates list-box.

**TE Index Name** - The index name for this Entity in the TE from which the index name in the DDL will be generated. If no name is entered, a default will be generated.

Once the fields of the Entity Identifier Detail dialog have been completed, press the **OK** button to accept the updates or **Cancel** to abort.

## Deleting an Identifier

Identifiers can be deleted two ways:

- In the Entity Identifier Detail dialog, press the **Delete** button. To access this dialog box, follow the instructions in [Updating an Identifier](#).
- In an Identifier list box, select the identifier to be deleted by clicking on it, then right mouse click and select the **Delete** option from the pop-up menu presented. Identifier list boxes are found in the following dialog boxes: the Identifier List dialog box. and the Entity Detail dialog box.

Refer to [Creating an Identifier](#) for details on opening these dialog boxes.



## Subject Areas

A subject area is subset of the data model. For large data models, it is often easier to maintain the data model by only working on a more limited subject area. Like the data model itself, a subject area can be viewed either graphically or in a list format.

It is important to note that a subject area is not a copy of some portion of the data model. It is only a restricted view of the data model. Hence, changes made in a subject area are, in fact, changes to the data model. The creation, modification or deletion of data model components in a subject area modify the data model itself.

### **See also:**

[Viewing Subject Areas](#)

[Creating a Subject Area](#)

[Opening a Subject Area for Update](#)

[Updating a Subject Area](#)

[Deleting a Subject Area](#)

## Viewing Subject Areas

Subject areas are viewed via a Subject Area window. In this window, you can view the subject area either in a graphical format similar to the Data Model window or in a list format similar to Entity List window. In many respects then, the Subject Area window will behave like either the Data Model window or Entity List window, depending on which mode you have chosen. By default, the Subject Area window is opened in a graphical format. To toggle between the two formats, simply press the **View** button in the Subject Area window.

**NOTE:** when the Subject Area window is displayed in a list format, an attribute list box will not be displayed as it is in the Entity List window. To view an attribute list for the current entity in a Subject Area window, you may open the Attribute List window by selecting the **Attribute List** item from the **View** pull-down menu in the Data Model window.

---

## Creating a Subject Area

To create a new Subject Area:

1. Select the **Open Subject Area** option from the **File** menu of the Data Model window to bring up the Open Subject Area dialog box.
2. Press the **New** button which presents the New Subject Area dialog box.
3. Provide a Name and, optionally, a Description and press the **OK** button. This will create the subject area bring up the Subject Area dialog box.

Updating the newly created subject area is discussed in further detail in [Updating a Subject Area](#).

## Opening a Subject Area for Update

To update an existing Subject Area:

1. Select the **Open Subject Area** option from the **File** menu of the Data Model window to bring up the Open Subject Area dialog box.
2. Double-click on the Subject Area to be opened or select it with a click and press the **OK** button. The Subject Area dialog box will be presented.

## Updating a Subject Area

A subject area is entirely defined by the subset of data model entities that are included in it. If an entity is added to a subject area, all of its attributes and its identifier are also visible in the subject area. If two entities that are related in the data model are both added to a subject area, then their relationship will be visible in the subject area as well.

Therefore, we must be careful to distinguish between adding and removing (i.e. including and excluding) entities to and from the subject area as opposed to creating and deleting entities.

Adding an entity to a subject area makes a previously existing data model entity visible in the subject area. Creating a new entity will create a new data model entity but will not always automatically add that entity to a subject area; even when using the Subject Area window to initiate the create!

Removing an entity from subject area will remove it from view within the subject area but will not delete it from the data model itself. However, deleting an entity (or relationship, attribute, etc.) in a subject area will, in fact, delete that entity from the data model. Remember, a subject area is not a copy of the data model. It is only a restricted view into the data model.

### **See also:**

[Adding an Entity to a Subject Area](#)

[Removing an Entity from a Subject Area](#)

[Creating, Updating and Deleting from a Subject Area](#)

## **Adding an Entity to a Subject Area**

In the Subject Area dialog box, paint an entity into the subject area by holding down the left mouse button and dragging where you want the entity placed. A Select Entity dialog box is presented. Double-click on the desired entity from the list to add it to the Subject Area.

## Removing an Entity from a Subject Area

In the Subject Area dialog box select the entity to be removed by clicking on it. Right click and select the **Remove Entity** option from the pop-up menu presented.

## Creating, Updating and Deleting from a Subject Area

With a few notable exceptions, the procedures used in the creation, modification and deletion of data model components from a subject area are identical to those described for the Data Model and Entity List windows. Some of the differences are listed here:

- When in graphic mode, you can create an entity by painting it in the Subject Area window. Doing so will open a Select Entity dialog box in which you can press **New**. This will then open the Entity Create window. Creating an entity in this manner will automatically add the entity to the subject area.
- Alternatively, you can open the Entity Detail window for an entity in the Subject Area window by double-clicking on the entity. In this window, you can press **New** to create a new entity. Creating an entity in this manner will not automatically add it to the subject area.
- You can create new relationships for an entity in the Subject area via the Relationship Detail window as before. However, if you create a relationship to an entity that is not in the subject area, that relationship will not be visible from the Subject Area window.



## Deleting a Subject Area

1. From the Data Model window, select the **Open Subject Area** item from the **File** pull-down window to open the Open Subject Area dialog.
2. Select the subject area to be deleted by single-clicking on the name. This will highlight the subject area.
3. Press the **Delete** button.

**NOTE:** since a subject area is only a view into the data model, deleting a subject area will not delete any entity, relationship or other data model component from the data model itself.

---

## Importing External Data Models

The Zeidon Data Model tool allows you to import data models created using certain external data modeling tools.

### See also:

[Importing ADW Data Models](#)

[Importing Siron Catalogs](#)

## Importing ADW Data Models

From the Data Model window, select the **Import ADW** item from the **Utilities** pull-down menu. This will open the ADW Import window where you specify the following:

**Source Directory** - The fully qualified path of the directory containing the ADW export files which are to be imported into the data model is entered in the text box provided.

**ADW Export Prefix** - The export file prefix for the OI, AI, PI and TI files to be imported is entered in the corresponding text box.

**Info Type** - The **Info Type** button is used to assign Zeidon domains for specific info types from the ADW data model. To use this function, press the **Info Type** button which will take you to the ADW Info Type To Domain Conversion window which will present a list of all info types from the ADW data model and their current domain assignment.

A domain is assigned to an info type by selecting an info type in the Current Settings list box and selecting the domain from the Set Domain Name drop-down combo box.

To save the domain selections for the info types, press **OK** or press **Cancel** to return without saving the selections. In either case you are returned to the ADW Import window.

**NOTE:** if you save the domain selections by pressing **OK** in the ADW Info Type To Domain Conversion window, the domain selections will be saved for any future import of the same ADW export files, regardless of whether or not this import is completed.

---

To import the ADW files, press **OK**.

# Importing Siron Catalogs

(under construction)

## Importing SAP

## Printing the Data Model

To print the data model select the **Print** option from the **File** menu of the Data Model window. This will present the Print dialog box where you can specify the following characteristics for the printing from the Print Profile group box:

**Orientation** - This describes the direction on the page that the data model will be printed. Using the drop-down combo box, select one of the following options:

**Best Fit** - Zeidon will determine if Landscape or Portrait is a better orientation for printing the model.

**Landscape** - Print the model in a lengthwise orientation. Use this if the model appears larger in length than height.

**Portrait** - Print the model with an upright orientation. Use this if the model appears larger in height than length.

**Method** - This option determines how the data model will be sized when it is printed. From the drop-down combo box, select one of the following:

**Fit to page** - Zeidon will fit the model on one page. It determines if 100% of the standard readable image will fit on one page, and if not, reduces that percentage the necessary amount to fit.

**Fill page** - Zeidon will reduce the image if the standard readable image is too large to fit on one page, or increase the image if it is smaller than would fill one page.

**Percent** - Use in conjunction with the Percent parameter. This indicates you will control the image size based on a percentage of the standard readable size. If your data model contains a large amount of entities you may want to choose Percent in order to have the Model printed on more than one page.

**Percent** - Specify the percentage of the standard readable image size you wish the model to be printed at. This parameter is applicable only when Percent is selected as the Method.

After making your Print Profile selections press the **OK** button to print or **Cancel** to abort.

## Genkeys and Dynamic Table Domains

Sometimes it is desirable to use a generated integer as a key for a table, rather than an existing entity or relationship from the data model. Zeidon will automatically generate such keys if you use a system generated identifier, as described in [Identifiers](#). These generated keys require that you define a genkey entity in your data model, however, along with a LOD that references it. You can have Zeidon add the entity and the corresponding LOD automatically from the Data Model tool.

To generate a genkey entity and the corresponding LOD, select the **Create Genkey Entity + LOD** option from the **Utilities** menu. This will generate the entity, save the data model and generate and save the necessary LOD. You must still build the TE yourself, however, and resave the LOD with the new TE.

For table domains, you usually define the values for those tables inside the domain itself, making the list of valid values a part of the domain definition and thus a part of the application definition. Sometimes it is desirable, however, to allow the user of the application to define table values as a part of the application, yet treat the attributes that use the table values the same as with normal table domains. This is explained under [Table Domains](#). Zeidon generates a genkey entity and the corresponding LOD automatically, yet

# **Zeidon Objects**

**Overview**

**Starting the Zeidon Object Tool**

**Creating a LOD**

**Opening a LOD for Update**

**LOD Details**

**Entities**

**Attributes**

**LOD Operations**

**Specifying the Technical Environment of a Zeidon Object**

**Zeidon Data Locking**

**Overriding Generated SQL**

**Committing Multiple Object Instances**



## Zeidon Objects - Overview

Generally, Logical Object Definitions (LODs) represent views into the data model. They are comprised of an initial entity called the root entity and any number of related entities. These views form a convenient way of structuring the data that is optimal for application development.

The entities which make up a Zeidon object can be imported one-to-one from the entities defined in the data model. Unlike entities in the data model, the entities in the LOD are arranged in a hierarchical relationship, with the root entity as the topmost.

Roughly speaking, we classify attributes and entities into two categories: those that are actually stored in the database and those that are not. Attributes and entities not stored in a data base are called *work attributes* and *work entities*. Work entities can either be defined in the data model, where they could be used in multiple LODs, or they can be defined specifically for a single LOD.

In a similar manner, we can classify LOD's themselves into two categories: database LODs and work LODs. An object instance for a *database LOD* contains data that is stored in the database. On the other hand, an object instance for a *work LOD* either exists only during application execution or it is stored in a file rather than a database.

## **Starting the Zeidon Object Tool**

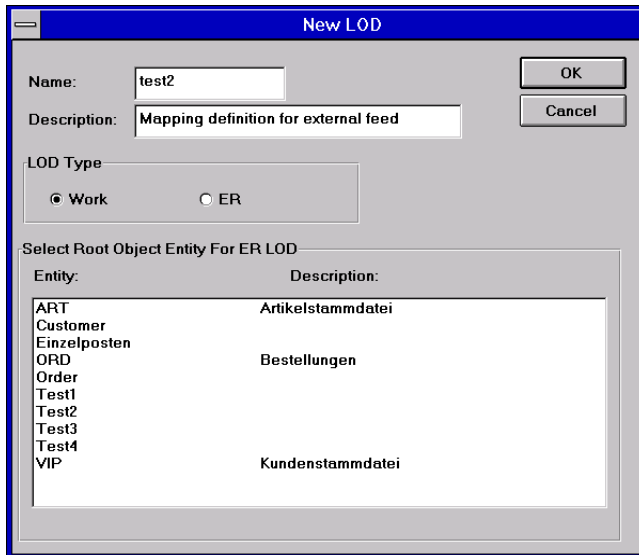
To start the Zeidon Object Definition tool, double-click on the Object icon in the Zeidon program group. If there are no LODs in your LPLR a New LOD dialog box is presented where you may start by creating a LOD. Otherwise, the Open Logical Object Definition dialog box is presented. From this window you can create a new LOD, update an existing LOD, delete a LOD, or exit the tool.

## Creating a LOD

LOD's are created via the New Lod window. This window can be opened in one of the two following ways.

- Select the **New** option from the **File** menu of the Logical Object Definition window.
- From the Open Logical Object Definition dialog box press the **New** button.

In either case, the following New LOD dialog is presented.



Entity:	Description:
ART	Artikelstammdatei
Customer	
Einzelposten	
ORD	Bestellungen
Order	
Test1	
Test2	
Test3	
Test4	
VIP	Kundenstammdatei

The root entity of the LOD can be an entity from the data model or it may be a work (non-persistent) entity. If you want to define a database LOD, that LOD must be defined with a root entity from the data model.

### See also:

[Creating a Database LOD](#)

[Creating a Work LOD](#)

[Creating a LOD by Copying Another LOD](#)

## Creating a Database LOD

To create a database LOD:

1. Open the New LOD window as described above.
2. Provide a Name and, optionally, a description for the LOD in the corresponding text boxes.
3. In the LOD Type box, select ER.
4. From the Select Root Object Entity For ER LOD list box, select an entity from the data model to be the root of the new LOD by single-clicking on it.
5. Press the **OK** button to do the create or press **Cancel** to abort.

## Creating a Work LOD

To create a work LOD:

1. Open the New Lod window as described above.
2. Provide a Name and, optionally, a description for the LOD.
3. In the LOD Type box, select Work.
4. Press the **OK** button to do the create or press **Cancel** to abort.

## **Creating a LOD By Copying Another LOD**

To create a LOD by copying another LOD:

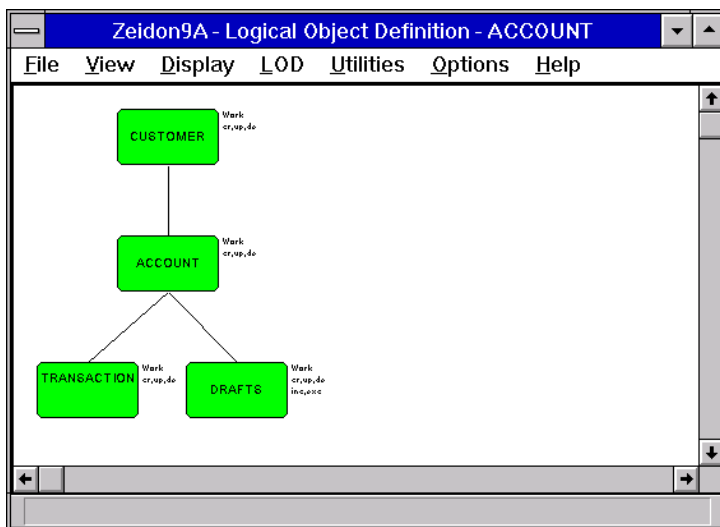
1. 1. Open the LOD list.
2. 2. Position the cursor on the LOD that will be the source of the copy.
3. 3. Press the Copy button.
4. 4. A subwindow titled "New LOD Name" will be presented. Key in the new LOD name and press the OK button or press Cancel to abort.

## Opening a LOD for Update

To Open a LOD for update:

1. Select the **Open** option from the **File** pull-menu of the Logical Object Definition Window
2. From the Open Logical Object Definition dialog select the LOD to be opened by clicking on it in the list box and then press the **OK** button. Alternatively, you can simply double-click on the LOD in the list box.

This will display the selected LOD in the Logical Object Definition window (shown below) from which all aspects of the LOD can be modified including the LOD detail, entities, attributes and operations.



## LOD Details

Certain basic information that describes a LOD are referred to as LOD details. Detail information for the LOD may be updated by selecting the **Object Detail** option from the **LOD** menu of the Logical Object Definition window. The Logical Object Definition Detail dialog box is presented.

Logical Object Definition Detail

Name: PerLogon

Desc:

DLL Name: PerLogon

Technical Environment:

Current TE: SISY

Multiple Root Limit:

Number of Entity Instances to Cache: 5

Data Locking:

Constraints:

Teste\_Abt

☒ Activate

☐ Activate Empty

☐ Commit Instance

☐ Drop Instance

Edit

Standard Zeidon Description

OK

Cancel

The following can be modified:

**Description** - This field is optional. Enter the description in the text box provided.

**DLL Name** - Enter the name of the DLL into which all operations for this LOD will be linked.

**Multiple Root Limit** - Specify the maximum number of root entities to be processed by the data base handler for this LOD if the activation of this object specifies Multiple Roots. If that limit is reached, the activate terminates at that point and gives a return code of 1. The program can then treat the activate as either completing normally or abnormally.

**Number of Entities to Cache** - You can sometimes improve performance by requesting Zeidon to cache entities within an object instance. This caching will not improve performance unless the entity instance occurs multiple times within an object instance; in fact, if this is not the case, this caching will actually have a negative effect on performance. Having an entity instance that occurs multiple times most often when a leaf node of a LOD is an included entity and an instance of that entity is duplicated down many paths from the root. In this case, you can reduce the number of SQL reads by identifying the number of cache buffers that will be searched to see if an entity instance has already been accessed. A typical number is in the range of 3 - 5. NOTE: if you set the caching number too high, you will generate extra overhead with little benefit in terms of performance.

**Constraints** - The Constraint drop-down combo box displays Object Constraint Operations defined for the LOD. Select a constraint, if appropriate, and then click on the check boxes to indicate the processing points where you want the operation invoked. The points at which the code is executed is as follows: Activate constraint code is executed after the object instance is activated. Activate empty constraint code is executed after the empty object instance is created. Commit instance constraint code is executed before the object instance is committed to the data base. Drop instance constraint code is executed before the object instance is dropped.



Once the LOD details have been updated, press **OK** to accept the changes or press **Cancel** to drop them.

## Entities

As in the data model, an entity is a discrete class of data items about which an organization stores information. The entities which make up a Zeidon object can be imported one-to-one from the entities defined in the data model. Unlike entities in the data model, the entities in the Zeidon object are arranged in a hierarchical relationship, with the root entity as the topmost. As such, any other entities added to the object will have a parent entity. Entities subordinate to another entity are child entities. Aside from the root entity, an entity may have any number of child entities related to it, but only one parent. Entities with the same parent in a Zeidon object are siblings.

### See also:

[Viewing Entities](#)

[Adding an Entity to a LOD](#)

[Updating an Entity](#)

## Viewing Entities

Entities are viewed graphically in the Logical Object Definition window. They can also be viewed in a list format. To view the entities in a list format, select the **Object Entity List** option from the **View** menu of the Logical Object Definition window.

In the Logical Object Definition window, where LODs are displayed graphically, viewing options are available which vary the way they entities are displayed. These options are available in the pop-up menu presented via a right mouse click when the cursor is positioned on an entity. They are:

**Hide** - This option reduces the presentation size of the entity positioned on, while still showing the hierarchical relationship line.

**Collapse** - This option will remove from view the child entities of the entity positioned on and their relationship lines. The parent will have a diamond symbol in its upper left corner indicating it has child entities collapsed from view under it.

**Expand** - This option returns the hidden or collapsed entities back to normal presentation.

**Center** - This option centers the selected entity in the view.

Additional viewing options are available from the pop-up menu of a right mouse click with the cursor positioned on empty space of the window background. These options include:

**Expand All** - This option returns all hidden and collapsed entities back to normal presentation.

**Switch Orientation** - This option toggles the view from a Portrait orientation to a Landscape orientation and vice-versa.

## Adding an Entity to a LOD

As described in [Creating a LOD](#) when a LOD is initialized, a root entity is created as part of the initialization. Subsequent additions of entities are made from either the Logical Object Definition window or from the Object Entity List (see [Viewing Entities](#)).

Entities are classified in three categories: related entities, derived entities and work entities. Portions of the addition process for these three types of entity are the same and are discussed in [Addition Basics](#). Specific instructions for each type of entity is discussed in further detail in the corresponding related sections.

### See also:

[Addition Basics](#)

[Adding a Related Entity](#)

[Adding a Derived Entity](#)

[Adding a Work Entity](#)

## Addition Basics

To initiate the addition of an entity, regardless of its type:

1. Select the entity to which you wish to add a child entity by clicking on the entity in the diagram or in the entity list box. This will be the parent entity.
2. Right mouse click on the selected entity to present a pop-up menu. There are three options in the menu which can be selected to add an entity. The choices are:

**Related** - The entity is added to the LOD from the data model and the relationship to its parent will be used to instance the entity type when activated.

**Derived** - The entity is added to the LOD from the data model without any of the relationship information to the parent entity. At activate time, no instance information for the entity and its children will be activated. Instancing of the entity is the responsibility of the application.

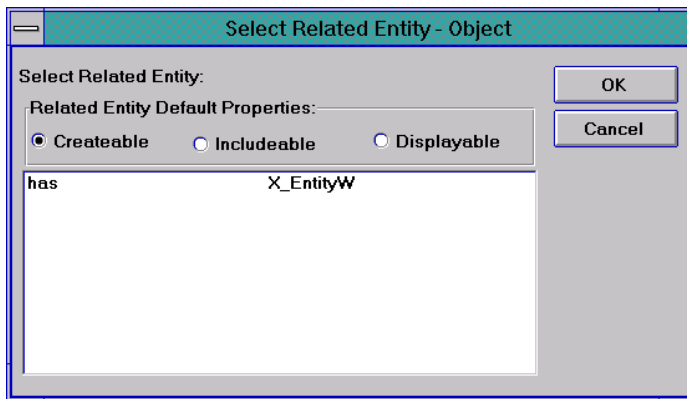
**Work** - The entity has no corresponding entity in the data model. At activate time, no instance information for the entity and its children will be activated. Instancing of the entity is the responsibility of the application.

3. Different dialog boxes will appear based upon the New Entity Type selected. Follow the specific instructions for completing the addition procedure for each entity type as outline in the following sections.

## Adding a Related Entity

To add a related entity:

1. Begin by following the instructions outlined in [Addition Basics](#). When **New Related Entity** is selected, a Select Related Entity dialog box is presented.



2. Select a radio button for one of the Related Entity Default Properties. This initializes the object entity update rules, or runtime permissions. The choices are:  
  
**Createable** - The runtime permissions will be Create, Delete, Update, and Include Source. Parent Delete behavior will be Delete.  
  
**Includeable** - The runtime permissions will be Include, Exclude, and Include Source. Parent Delete behavior will be Exclude.  
  
**Displayable** - The runtime permissions will only be Include Source. Parent Delete behavior will be Restrict..  
  
3. Listed in this dialog box are all relationships defined in the data model for the selected entity. Add a child entity by double-clicking on the entity/relationship in the Select Related Entity list. This will automatically close the Select Related Entity window. If you wish to abort the create, you can close the window by double-clicking on the close box in the upper left hand corner of the window.

## Adding a Derived Entity

To add a derived entity:

1. Begin by following the instructions outlined in [Addition Basics](#). When **New Derived Entity** is selected a Select Derived Entity dialog box is presented which lists all entities in the data model.
2. Select the entity desired to be the child by double-clicking on it. This will automatically close the Select Related Entity window. If you wish to abort the create, you can close the window by double-clicking on the close box in the upper left hand corner of the window.

## Adding a Work Entity

To add a work entity:

1. Begin by following the instructions outlined in [Addition Basics](#). When **New Work Entity** is selected an Object Entity Detail dialog box is presented.
2. Enter a Name for the work entity and other pertinent entity information as described in [Updating an Entity](#).
3. Press **OK** in the Object Entity Detail to add the work entity or press **Cancel** to abort.



## Updating an Entity

Entities are updated from the Object Entity Detail dialog box. There are a number of ways to bring up this dialog box. They are:

- Select the entity you wish to update by clicking on the entity in the entity list box. Right mouse click and select Entity detail from the pop-up menu.
- Positioning the cursor on the entity you wish to update and right mouse click. Select Entity detail from the pop-up menu.
- Double-clicking on the desired entity in the diagram or in the entity list box.

The screenshot shows the 'Object Entity Detail' dialog box. The 'Name' field is 'Mitarbeiter' and the 'Desc' field is 'Mitarbeiter'. The 'Runtime Permissions' section has checkboxes for Create, Delete, Update, Include, Exclude, and Include Source (checked). The 'Constraints via Operation' section has a dropdown menu and checkboxes for Accept, Create, Include, Cancel, Delete, and Exclude. The 'Data Locking' section has a dropdown menu. The 'Specification Options' section has checkboxes for Derived, Recursive, Dup Entity Instance, Dup Relationship Instance, and Work. The 'Parent Delete Behavior' section has a dropdown menu set to 'none'. There are buttons for OK, Cancel, Next, Prev, Delete, and Edit. At the bottom is a field for 'QuinSoft Name'.

The parameters which may be modified on a Zeidon Object include:

**Name** - Enter a name to identify the LOD in the text box.

**Description** - Enter an optional description in the text box.

**Runtime Permissions** - This field identifies how updates to the entity will be constrained. Select from the check box(es) the permissible update types allowed for the entity. The choices include: Create, Delete, Update, Include, Exclude, and Include Source (Source of Include).

**Constraints via Operation** - From the drop-down combo box, select an entity constraint operation for this entity, if appropriate, and click on the check boxes which identify the process points for the operation to be invoked.

**Parent Delete Behavior** - This field identifies what occurs to this entity if a parent entity in the hierarchical chain is deleted. The possibilities are:

**Delete** - This entity will be deleted from the data base.

**Exclude** - Deletes the relationship only.

**Restrict** - Deleting the parent is restricted when an instance of this entity exists.

**Specification Options** - Select the appropriate check boxes from this field. The options include:

**Derived** - The entity is defined in the data model, but the relationship to its parent entity in the LOD is not from the data model. Instanciation of the entity is the responsibility of the application.

**Work** - The entity is not defined on a database or in the data model. Instanciation of the entity is the responsibility of the application.

**Recursive** - The ER Entity type for the Object Entity is the same as the ER Entity for its immediate or ancestral parent and activation of an instance of the entity will continue downward following the definition of the ancestral parent. This can only be specified for a leaf Object Entity.

**Dup Entity Instance** - Dup is short for duplicate. In this case, the ER Entity type occurs more than one time in the LOD and at activation time, instances of the Object Entity can also occur more than once in the activated Object Instance. Specifying Dup Entity Instance will resolve duplicate instances after activation so duplicate instances share one copy of Entity Attribute data.

**Dup Relationship Instance** - Dup is short for duplicate. The ER Relationship type to the parent entity occurs more than one time in the LOD and activation of this relationship can be resolved from other instance data in the object. After completion of database activation, duplicate relationships will be resolved from the other instance information activated instead of resolving the relationship from the database. This is a performance option only.

Once the entity details have been updated, press **OK** to accept the changes or **Cancel** to abort.

## Attributes

Each Entity in a LOD is made up of attributes which describe the entity. LOD Entities which are based on an ER Entity inherit the attributes from the ER Entity. In addition, LOD Entities may add any Work and Derived Attributes which may be desired to simplify application processing requirements.

### See also:

[Viewing Attributes](#)

[Updating an Attribute](#)

[Creating an Attribute](#)

[Deleting an Attribute](#)

[Re-including Entity Attributes](#)

## Viewing Attributes

In the Logical Object Definition window, list the attributes for an entity using one of the following methods:

- Select the **Object Attribute List** option from the **View** menu of the Logical Object Definition window.
- Right mouse click on empty space in the window background of the Logical Object Definition window. Select the **Object Attribute List** option from the pop-up menu presented.

The dialog box presented is the Object Attribute List dialog box. Select and/or change the entity being listed via a single-click on the entity in the diagram of the Logical Object Definition window.

In the Object Entity List window, single-click on the entity in the entity list box. The attributes for the highlighted entity are displayed in the Attributes list box.

## Updating an Attribute

Attributes are updated in the Object Attribute Detail dialog box. This dialog box is accessed through the Object Attribute List dialog box (see [Viewing Attributes](#)). On the attribute list either:

- Double-click the attribute to be updated.
- Select the attribute to be updated with a single-click, then right mouse click inside the list box and select Attribute Detail from the pop-up men.

The Object Attribute Detail dialog box is presented.

**Object Attribute Detail - CUSTOMER**

Name:	<input type="text" value="LastName"/>	Length:	<input type="text" value="32"/>	<input type="button" value="OK"/> <input type="button" value="Cancel"/> <input type="button" value="Next"/> <input type="button" value="Prev"/> <input type="button" value="New"/> <input type="button" value="Delete"/>
Desc:	<div style="border: 1px solid black; height: 100px; width: 100%;"></div>	Length:	<input type="text" value="254"/>	
Domain:	<input type="text" value="Text"/>	Length:	<input type="text"/>	
Data Type:	String - Any length	Length:	<input type="text"/>	
Properties:	<input checked="" type="checkbox"/> NotNull <input type="checkbox"/> Case Sensitive			
Prompt:	<input type="text"/>			
List Title:	<input type="text"/>			
<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <input checked="" type="checkbox"/> Updateable            Initial Value: <input type="text"/>            Derivation Operation: <input type="text"/> </div> <div style="width: 45%;">           Sequence:            Priority: <input type="text"/>            Order:  <input checked="" type="radio"/> Ascending  <input type="radio"/> Descending         </div> </div>				

QuinSoft Name - left justified, no spaces

Whether specific detail information of an attribute may be modified depends on whether the attribute entity is a model, work, or derived attribute. Model attributes are attributes which exist in the corresponding data model entity associated with the Object Entity. Since their attribute information is defined in the model, most of the detail for these attributes is not modifiable. A work attribute is an attribute which is created for use with its associated Object Entity only. It has no tie back to the associated data model entity in the model and therefore, all its detail information must be specified here. A derived attribute is a special kind of work attribute which has a derivation algorithm associated with it. The derivation algorithm is invoked each time the attribute is set or retrieved. Since it too has no tie back to the data model Entity, its detail is defined in this dialog box.

The following fields are modifiable for work and derived attributes only:

**Name** - Name is used to identify the attribute.

**Description** - A short description is optional.

**Domain** - Every attribute must be assigned a domain. Select a domain from the Domain drop-down combo box.

**Length** - For attributes with string data type domains, a length may be specified to override the length specification of the domain.

**Not Null** - Select this check box if data must be provided for this attribute.

**Case Sensitive** - For attributes with string data type domains, this check box indicates whether or not to enforce upper/lower case sensitivity.

**Prompt** - This Auto Design field is used to enter the text you want to identify this attribute when it is displayed on a window or dialog box created through Auto Design. Do not enclose the text in quotes unless you wish the quotes painted as part of the identifying text. If no text is entered the default is to use the attribute name for Auto Design.

**Prompt Length** - Enter the length the control that will be autodesigned on a display or update window. If this is not specified, the length of the attribute (see above) will be used.

**List Title** - This Auto Design field is used to enter the text you want to identify this attribute when it is displayed in a list box created through Auto Design. The entered text will be the column heading on the list box for that attribute .

**List Length** - Enter the length the control that will be autodesigned on a list box. If this is not specified, the length of the attribute (see above) will be used.

The following fields are modifiable for all attributes.

**Updateable** - This check box should be clicked on to allow the attribute to be updated. If it is not an update constraint will be automatic.

**Initial Value** - Specify a value to initialize this attribute to when the entity is created.

**Derivation Operation** - If this is a derived attribute, select the operation to be invoked during retrieval and setting of this attribute from the drop-down combo box.

**Priority** - Enter the priority number if this attribute is to be part of a sorting sequence. Click the Ascending or Descending radio button to indicate how the attribute is sorted.

**Specified** - Enter the text for a Message Bar message if you want to override the default inherited message.

Once the fields for the attribute detail have been updated, press **OK** to accept the changes or **Cancel** to abort. If you updating more than one attribute for an entity at a time, you can cycle through the Object Attribute Detail windows for all the attributes for the current entity by pressing the **Prev** and **Next** buttons.

## Creating an Attribute

Attributes created within the Zeidon Object tool are either work attributes or derived attributes. That is, they have no tie back to any entity in the associated data model entity in the model. Attributes are created in one of the following ways:

- Right click in the list box of the Object Attribute List dialog box (see [Viewing Attributes](#) ). From the pop-up menu presented select New Work Attribute. This will bring up the Object Attribute Detail dialog box.
- While on the Object Attribute Detail dialog box, press the New button. This refreshes the dialog box so that details for the new attribute may be entered.

Enter the necessary fields for the attribute as described in [Updating an Attribute](#) and when finished, press the **OK** button.

## Deleting an Attribute

Attributes may be deleted in either of the following two ways:

- From the Object Attribute List dialog box, highlight the attribute you wish deleted by single-clicking on it. Right mouse click in the window and select **Delete** from the pop-up menu presented. See [Viewing Attributes](#) for details on how to bring up the Object Attribute List dialog box.
- From the Object Attribute Detail dialog box press the **Delete** button. See [Updating an Attribute](#) for details on how to bring up this dialog box.



## Re-including Entity Attributes

Attributes may be re-included into an entity from the data model. This may necessary if attributes were inadvertently deleted from the entity. From the Object Attribute List dialog box right mouse click in the list box and select **Add All ER Attributes** from the pop-up menu. See [Viewing Attributes](#) for details on how to bring up the Object Attribute List dialog box. Note that all attributes from the data model are refreshed and any work attributes which were added are kept.

## LOD Operations

A LOD can have many operations defined for it. In order to assign an operation to a LOD, a Source file must be assigned to the LOD. The Source file is where the operation is maintained. Multiple source files can be assigned to a LOD and thus different operations for a LOD may be from different source files.

### See also:

[Creating a New LOD Source File](#)

[Updating a LOD Source File](#)

[Deleting a LOD Source File](#)

[Creating a New Operation](#)

[Updating an Operation](#)

[Editing an Operation Source](#)

[Deleting an Operation](#)

[Writing a Derived Attribute Operation](#)

[Writing an Entity Constraint Operation](#)

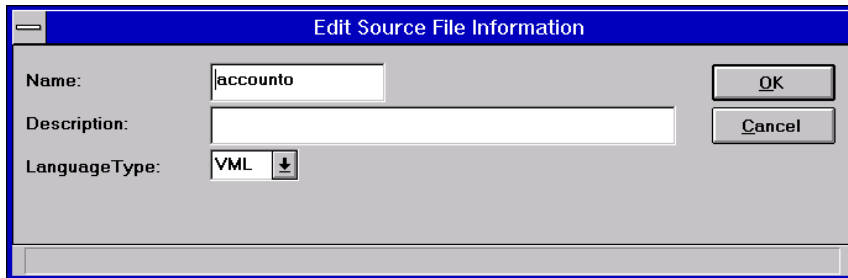
[Writing an Object Constraint Operation](#)

[Writing a Transformation Operation](#)

## Creating a New LOD Source File

To create a source file for a LOD:

1. From the Logical Object Definition window, select **Source Files** from the **LOD** pull-down menu. This will open the Source File Maintenance window.
2. In the Source File Maintenance window, press the **New** button to open the Edit Source File Information window.



The screenshot shows a dialog box titled "Edit Source File Information". It has a blue title bar with a minus button on the left. The dialog contains three input fields: "Name:" with the text "accounto", "Description:" which is empty, and "LanguageType:" with a dropdown menu showing "VML" and a downward arrow. To the right of these fields are two buttons: "OK" and "Cancel".

3. Enter a name for the source file in the edit box.  
NOTE: the name must follow any naming conventions for the operating system and must be unique with respect to this LPLR.
4. If desired, enter a short description for the source file.
5. Select the language type for the source file using the combo-box.
6. Press **OK** to create the file or **Cancel** to abort.

## Updating a LOD Source File

To update a LOD source file:

1. From the Logical Object Definition window, select **Source Files** from the **LOD** pull-down menu. This will open the Source File Maintenance window.
2. In the Source File Maintenance window, double-click on the source file in the list box. This will open the Edit Source File Information window.
3. Update the fields of this window as described in [Creating a New LOD Source File](#).

## Deleting a LOD Source File

**NOTE:** you should only attempt to delete a source file if it contains no operations for the dialog. If it contains operations for the dialog, you should first delete these operations as described below in the section [Deleting an Operation](#).

---

To delete a source file for a dialog:

1. From the Logical Object Definition window, open the Source File Maintenance dialog by selecting the **Source File** item under the **LOD** pull-down menu.
2. Using the Source File Name list box, single-click on the file to be deleted. This will highlight the source file in the list box.
3. Press **Delete** to delete the source file or press **Cancel** to abort.

## Creating a New Operation

**NOTE:** There must be a source file assigned to the dialog in order to create an operation. If no source file exists, follow the directions in [Creating a New LOD Source File](#).

---

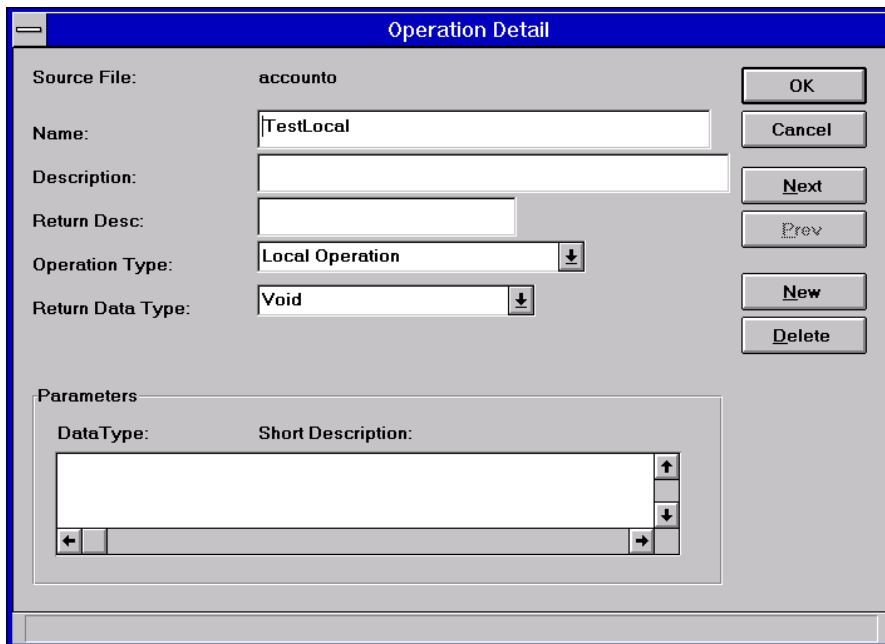
To create a new operation to be added to the source file and assigned to the LOD:

1. From the Logical Object Definition window, select **Operations** from the **LOD** menu. The Operation Maintenance window is displayed.
2. Press the **New** button. The Select Source File for Operation window is displayed.
3. Select the source file where the operation code will be stored by single-clicking on it in the Source File list box and then press **OK**. The Operation Detail window is displayed.
4. Complete the operation details as described in [Updating an Operation](#).
5. If you want to enter the Editor to edit the newly created operation, highlight the operation and select the **Edit** button in the Operation Detail window.
6. Select **OK** to complete the create and to return to the Operation Maintenance window or press **Cancel** to drop it.

## Updating an Operation

To update a LOD operation:

1. From the Logical Object Definition window, select **Operations** from the **LOD** menu. The Operation Maintenance window is displayed.
2. Double-click the operation to be updated from the operation list. The Operation Detail dialog box is displayed.



3. In the Operation Detail window, enter a name and, optionally, a description for the operation, as well as the following information:

**Operation Type** - There are different types of operations which may be assigned to a LOD. Select the operation type from the Operation Type combo box. The choices are:

**Object Constraint Operation** - May be invoked at Activate, Activate Empty, Commit and Drop Instance.

**Entity Constraint Operation** - May be invoked at Accept, Create, Include, Cancel, Delete, Exclude of a specified entity(s).

**Transformation Operation** - May be invoked from any other Zeidon operations.

**Derived Attribute Operation** - Invoked during the setting and retrieval of a specified attribute(s).

**Local Operation** - May be invoked from other Zeidon operations within the same source file.

**Return Data Type** - Use the combo-box to select the data type for the return value.

**Return Description** - Enter an optional description of the return value for the operation.

4. If desired, you can modify, add to and delete the default parameters by doing the following:

**Modify a Parameter** - Select the parameter by single-clicking on it in the Parameters list box. With the cursor in the list box, single-click the right mouse button to display a pop-up menu. Select **Parameter Detail** and the Parameter Maintenance dialog is presented. From this dialog, you can change:

**Description** - Enter a description for the parameter in the edit box provided.

**Data Type** - Specify the data type for the parameter by using the pull-down combo box.

**Returned Value** - If the parameter is to be modified by the operation and returned to the caller, choose **Y** (yes). Otherwise, choose **N** (no).

**Unsigned** - If the parameter is a numeric value that is to be treated as an unsigned value, choose **Y** (yes). Otherwise, choose **N** (no).

If there is more than one parameter you wish to update, press **Next** or **Prev** to cycle through the other parameters. Otherwise, you can press **OK** to accept the modifications or you can press **Cancel** to abort.

**Add a New Parameter** - Single-click the right mouse button while the cursor is in Parameters list box. Select the **New** item from the pop-up menu presented. The Parameter Maintenance dialog is presented. Complete the fields of this dialog as described immediately above.

**Delete a Parameter** - Delete a parameter in one of two ways.

- From the Operation Detail window, select the parameter from the Parameters list box by single-clicking on it. With the cursor in the list box, single-click the right mouse button to present the pop-up menu and select the **Delete** item.
  - From the Parameter Maintenance dialog, press **Delete**.
5. If you want to enter the Editor to edit the operation, highlight the operation and select the **Edit** button in the Operation Detail window.
  6. If you wish to update the details of several operations in one session, you can use the **Next** and **Prev** buttons in the Operation Detail dialog to cycle through the operation list for the dialog. When you are done with the updates, press **OK** to accept them or press **Cancel**



## Editing an Operation Source

To edit the source for an existing operation:

1. From the Logical Object Definition window, select **Operations** from the **Dialog** pull-down menu. The Operation Maintenance window is displayed.
2. Single-click on the operation in the Operation Name list to highlight it.
3. Press the **Edit** button to enter the system editor.

## Deleting an Operation

## Writing a Derived Attribute Operation

When you first enter the Editor after specifying a new derived attribute, you will be presented with an operation shell structured as follows.

```
DERIVED ATTRIBUTE OPERATION

OperationName (VIEW ViewtoInstance BASED ON LOD ObjectName,

               STRING ( 32 ) InternalEntityStructure,

               STRING ( 32 ) InternalAttribStructure,

               SHORT GetOrSetFlag )

CASE GetOrSetFlag

OF  zDERIVED_GET:

    /* end zDERIVED_GET */

OF zDERIVED_SET:

    /* end zDERIVED_SET */

END /* case */

END
```

The two OF statements allow you to specify logic that will be executed on both the retrieval of the value for the derived attribute (the GET) or on the setting of the value for the derived attribute (the SET). In most cases, you will only have GET logic and can delete the second OF or leave it empty.

The first parameter passed in is the view to the object instance for which the derived attribute was requested. The ObjectName will be filled in with the actual name of the LOD. The view name will be filled in with "ViewtoInstance" and can be altered or used as is. The second two parameters are two pointers that you do not reference directly but are used in the operations to store or retrieve a value. The last parameter identifies whether the operation is triggered by either the GET or SET event.

The most common case is zDERIVED\_GET, where an attribute value is set as a result of an algorithm. You can use any attributes from the entity in the main object for the operation or attribute from any object for which you can get a view by name. The derived operation is triggered whenever the attribute is referenced.

The following example presents a derived attribute for the GET case. The derived attribute is for an invoice total, which is computed by summing the line item amounts and applying a discount factor.

```
DERIVED ATTRIBUTE OPERATION

CalculateTotalForInvoice (VIEW vCustomer BASED ON LOD Customer,

                          STRING ( 32 ) InternalEntityStructure,

                          STRING ( 32 ) InternalAttribStructure,

                          SHORT GetOrSetFlag )
```

```

    DECIMAL InvoiceTotal

    VIEW vCustomerTBased on LOD Customer

    CASE GetOrSetFlag

    OF zDERIVED_GET:

        Create View From View (vCustomerT, vCustomer)

    InvoiceTotal = 0

    FOR EACH vCustomerT.Line_Item

        InvoiceTotal = InvoiceTotal + vCustomerT.Line_Item.Amount

    END

    InvoiceTotal = InvoiceTotal * (1.00 - vCustomerT.Invoice.Discount)

    StoreValueInRecord ( vCustomer,

        InternalEntityStructure,

        InternalAttribStructure,

        InvoiceTotal,

        0 )

        DropView (vCustomerT)

    /* end zDERIVED_GET */

    OF zDERIVED_SET:

        /* end zDERIVED_SET */

    END /* case */

END

```

## Writing an Entity Constraint Operation

When you first enter the Editor after specifying a new entity constraint, you will be presented with an operation shell structured as follows.

```
ENTITY CONSTRAINT OPERATION

OperationName (VIEW ViewtoInstance BASED ON LOD ObjectName,

               STRING ( 32 ) sEntityName,

               SHORT Event,

               SHORT State )

CASE Event

OF  zECE_ACCEPT:

    /* end zECE_ACCEPT */

OF  zECE_CANCEL:

    /* end zECE_CANCEL */

OF  zECE_CREATE:

    /* end zECE_CREATE */

OF  zECE_DELETE:

    /* end zECE_DELETE */

OF  zECE_EXCLUDE:

    /* end zECE_EXCLUDE */

OF  zECE_INCLUDE:

    /* end zECE_INCLUDE */

END /* case */

END
```

The six OF statements allow you to specify logic that will be executed on the six possible events that occur for an entity. The most common event is the ACCEPT event, which is triggered when other processing, usually a dialog, requests that changes to the subobject starting with the entity be accepted. The other events occur on cancel, create, delete, include and exclude activity on the entity.

The first parameter passed is the view to the object instance for which the event was triggered. The ObjectName will be filled in with the actual name of the LOD. The view name will be filled in with "ViewtoInstance" and can be altered or used as is. The second parameter identifies the entity name for which the event was triggered. The third parameter identifies the event itself and the fourth parameter is currently not used.

During constraint processing if you detect a constraint violation that should cause the action to be terminated, you should send an error message to the message object using either the operation `MessageSend`, which will present the message to the operator and return with an abort condition by exiting with a return code of `zCONSTRAINT_VIOLATION`.

The following example presents an entity constraint that is triggered when an include is issued for the `Cassette` entity. It checks to see if more than 10 movies are trying to be rented and returns an error if the condition is met.

ENTITY CONSTRAINT OPERATION

MovieSelection (VIEW vCustomer BASED ON LOD Customer,

    STRING ( 32 ) sEntityName,

    SHORT Event,

    SHORT State )

Integer NumberOfMoviesRented

CASE Event

OF zECE\_ACCEPT:

    /\* end zECE\_ACCEPT \*/

OF zECE\_CANCEL:

    /\* end zECE\_CANCEL \*/

OF zECE\_CREATE:

    /\* end zECE\_CREATE \*/

OF zECE\_DELETE:

    /\* end zECE\_DELETE \*/

OF zECE\_EXCLUDE:

    /\* end zECE\_EXCLUDE \*/

OF zECE\_INCLUDE:

    NumberOfMoviesRented = 0

    FOR EACH vCustomer.Cassette WITHIN vCustomer.Customer

        NumberOfMoviesRented = NumberOfMoviesRented + 1

    END

    IF NumberOfMoviesRented >= 10

        MessageSend( vCustomer, "CV001", "Customer Validation",

```
limit.", "You currently have 10 movies selected for rent. 10 is the
```

```
zMSGQ_OBJECT_CONSTRAINT_ERROR, 0 )
```

```
RETURN zCONSTRAINT_VIOLATION
```

```
END
```

```
/* end zECE_INCLUDE */
```

```
END /* case */
```

```
END
```

## Writing an Object Constraint Operation

When you first enter the Editor after specifying a new object constraint, you will be presented with an operation shell structured as follows.

```
OBJECT CONSTRAINT OPERATION

OperationName (VIEW ViewtoInstance BASED ON LOD ObjectName,

              SHORT Event,

              SHORT State )

CASE Event

OF  ZOCE_ACTIVATE:

    /* end ZOCE_ACTIVATE */

OF  ZOCE_EMPTY:

    /* end ZOCE_EMPTY */

OF  ZOCE_COMMIT:

    /* end ZOCE_COMMIT */

OF  ZOCE_DROPOI:

    /* end ZOCE_DROPOI */

END /* case */

END
```

The four OF statements allow you to specify logic that will be executed on the four possible events that occur for an object. The most common event is the COMMIT event, which is triggered when other processing, usually a dialog, requests that changes to the object are committed to a database or file. The ACTIVATE event is triggered when other processing activates an object instance from the database or a file. The EMPTY event is triggered when other processing activates an object instance as new or empty. The DROPOI event is triggered when other processing issues the DropObjectInstance operation for this object.

The first parameter passed is the view to the object instance for which the event was triggered. The ObjectName will be filled in with the actual name of the LOD. The view name will be filled in with "ViewtoInstance" and can be altered or used as is. The second parameter identifies the event itself. The fourth parameter is used to identify special states that may be necessary to handle some situations. The four possible states are as follows:

**ZOCE\_STATE\_FILE** - Indicates that the activate or commit is for a file, instead of a database.

**ZOCE\_STATE\_NOI** - On the activate of an object instance for a file, it indicates that the file could not be found to open, but that an option on the Activate operation indicated that it was OK to have no object instance.

**ZOCE\_STATE\_SHUTDOWN** - On a drop of an object instance, it indicates that the drop occurred during the



shutdown of a task for which the view was created. This occurs automatically on task shutdown so that any active object instances can be handled properly.

**zOCE\_STATE\_SYSTEM** - On the activate of an object instance, it indicates that the Activate operation requested the object instance be tied to the system task, instead of the application task.

During constraint processing if you detect a constraint violation that should cause the action to be terminated, you should send an error message to the message object using the operation MessageSend, which will likely present the message to the operator, and return with an abort condition by exiting with a return code of zCONSTRAINT\_VIOLATION.

The following example presents an object constraint that is triggered on a commit of the Customer object. It is the same constraint specified on an earlier example for an entity include. It checks to see if more than 10 movies are trying to be rented and returns an error if the condition is met. In practice, if two different events were to trigger the same constraint, that constraint would be defined as a local operation and would be executed from the operation for each event.

#### OBJECT CONSTRAINT OPERATION

```
CommitCustomer (VIEW vCustomer BASED ON LOD Customer,

    SHORT Event,

    SHORT State )

CASE Event

OF  zOCE_ACTIVATE:

    /* end zOCE_ACTIVATE */

OF zOCE_EMPTY:

    /* end zOCE_EMPTY */

OF zOCE_COMMIT:

    /* end zOCE_COMMIT */

OF zOCE_DROPOI:

    NumberOfMoviesRented = 0

    FOR EACH vCustomer.Cassette WITHIN vCustomer.Customer

        NumberOfMoviesRented = NumberOfMoviesRented + 1

    END

    IF NumberOfMoviesRented >= 10

        MessageSend( vCustomer, "CV001", "Customer Validation",

            "You currently have 10 movies selected for rent. 10 is the
limit.",

            zMSGQ_OBJECT_CONSTRAINT_ERROR, 0 )
```

```
        RETURN zCONSTRAINT_VIOLATION

    END

    /* end zOCE_DROPOI */

END /* case */
```

## Writing a Transformation Operation

When you first enter the Editor after specifying a new transformation, you will be presented with an operation shell structured as follows.

```
TRANSFORMATION OPERATION
```

```
OperationName (VIEW ViewtoInstance BASED ON LOD ObjectName,  
  
    param2,  
  
    etc.  )
```

```
END
```

The shell is basically empty, because a transformation has no specific structure. Its structure is completely up to you. The first parameter passed is the view to the object instance for which the event was triggered. The ObjectName will be filled in with the actual name of the LOD. The view name will be filled in with "ViewtoInstance" and can be altered or used as is. The additional parameters are generated based on the number of parameters you defined when the transformation operation was defined.

The following example is a transformation that builds an invoice subobject when an external process, probably a dialog operation, issues a call to the operation Build\_Invoice.

```
TRANSFORMATION OPERATION
```

```
Build_Invoice (VIEW vCustomer BASED ON LOD Customer )
```

```
    IF vCustomer.Invoice EXISTS
```

```
        DELETE ENTITY vCustomer.Invoice
```

```
    END
```

```
    FOR EACH vCustomer.RentalInformation
```

```
        CREATE ENTITY vCustomer.Line_Item
```

```
        INCLUDE vCustomer.InvoiceRental FROM vCustomer.RentalInformation
```

```
    END
```

```
END
```

## Specifying the Technical Environment of a Zeidon Object

To specify the technical environment of a Zeidon Object:

1. Select the **Technical Environment** option from the **Utilities** menu on the Logical Object Definition window.
2. In the List of Technical Environments dialog box either:
  - select the desired technical environment from the list box with a single-click and press the **Save with TE** button, or
  - double-click the desired technical environment.

This saves the LOD and includes the technical environment information. Subsequent saves of the LOD will be for that technical environment. If you want to undo the specification of a technical environment for the LOD press, the **Save with No TE** button. Subsequent saves will then not contain technical environment information.

## Zeidon Data Locking

Zeidon's philosophy of locking is to avoid using database locks across long transactions (business transactions). Zeidon does use database locks, but only across short transactions where there is no performance problems. This occurs during the process of the commit of an object instance. At the beginning of the commit, a database start transaction is issued and continues during all the reading and writing of the individual entities within the object instance being committed. At the end of the commit, the database transaction is terminated. The database management system issues database locks within the execution of the transaction and releases them upon transaction termination.

Zeidon does not use those database locks during long transactions primarily because of the performance problems that can occur when database locks are held over a long period of time. An additional reason for avoiding database locks, however, is that the physical level at which database locking occurs is inconsistent with the logical transactions around which an application is organized and around which locks are best defined. Thus, if the long transaction is "Maintaining Prospect Information", a logical lock around the transaction is easier to understand and control than individual database locks on all the pieces of the transaction. It is also easier to avoid deadlocks during long transactions when individual database locks are avoided.

Zeidon implements two kinds of protection against simultaneous updates of data in the database, commonly called optimistic locking and pessimistic locking. Together with the condition of not using any locks, there are three different locking options as follows.

- **No Locking** - No locking will be implemented over long transactions. If this option is selected, either no locking should be necessary for the application or the application should perform its own locking. After an object instance is activated, it can be committed to the database at any time. Zeidon will not protect against simultaneous updates.
- **Optimistic Locking** - When this option is selected, no actual locks are issued even though the option protects against simultaneous updates. After an object instance is activated, any database locks that may have been issued during the Activate operation are released. Simultaneous updates are prevented by comparing the original state of an object against the database when the object is committed. During activate of an object instance, its original state is captured and routed with the object instance to all its destinations. When changes to the object instance are committed to the database, the original state of the object instance is compared to the data in the database and if it does not match (i.e. another transaction has already modified some of the data contained in the object instance), the Commit transaction is rejected. Two different kinds of compares can be made depending on Optimistic Locking options explained in [Optimistic Locking](#).
- **Pessimistic Locking** - When an object instance is activated, one or more logical locks are issued. If any other Activate transaction is issued against the same data, that transaction is either rejected or the returned object instance is made a read-only instance, as determined by Pessimistic Locking options explained in [Pessimistic Locking Options](#). When the object instance is committed, the logical locks are removed.

Zeidon's objective is to control locking options in the LOD, where a data administrator can determine current settings without having to search through application programming logic and where he can also set and modify those options through the high-level definition of the LOD. The only programming consideration occurs with Pessimistic Locking as explained in [Pessimistic Locking Options](#).

### See also:

[Optimistic Locking](#)

[Pessimistic Locking Options](#)

## Optimistic Locking

Optimistic locking is selected by specifying one of two optimistic locking options in the LOD. As explained in [Zeidon Data Locking](#), selecting either option will not result in any actual lock to be issued, but will instead cause the original state of the object instance to be saved and later compared against the database during Commit processing. If the data content of the original state does not match the database, the Commit transaction is rejected with a return code of -5 (zLOCK\_ERROR).

To select optimistic locking for a LOD, from the main window of the LOD tool, select **LOD Detail...** from the **LOD** menu. This will present the LOD Detail window. Select the desired optimistic locking option from the Data Locking combo-box.

The optimistic locking options are:

- **Optimistic Locking With Partial Compare** - will cause the compare to be made only between entities in the object instance that have been Create and/or Update permission and between keys of entities that don't have Create/Update permission. If the comparison shows changes then the Commit transaction is rejected. However, if the data in an Includable entity for the original state of the object instance is different from the data for that entity in the database, the Commit transaction is accepted.
- **Optimistic Locking With Full Compare** - will cause the compare to be made against the data in all entities for the original state of the object instance. This is an unusual situation, however, because it is common that included data is being modified by other transactions.

In addition to selecting Optimistic locking on the LOD, you must specify the SingleForUpdate option on all Activate statements for object instances for that LOD that are to be updated. If an Activate has any locking specified for the associated LOD and does not have SingleForUpdate specified (ex., `ACTIVATE vCustomer Single`), then a read-only object instance will be returned and a later Commit will result in an error.

## Pessimistic Locking Options

Pessimistic Locking is selected by specifying one of two pessimistic locking options in the LOD and optionally in one or more of the subordinate entities to the LOD. Selecting either option will cause one or more logical locks to be issued against the database at the time the object instance is activated. You would normally request the lock only for the root of the LOD and not for subordinate entities. This is usually adequate because subordinate entities to the LOD are usually unique to a particular object instance or are includable and not locked. In fact, locking is usually determined by the entity type as follows.

- **Attributive** - Attributive entities are unique under a parent up to the root entity. Thus an attributive entity would never appear within two different root instances for a LOD and the lock on the root would be adequate to lock the attributive entity.
- **Associative** - Associative entities are also unique under a parent up to the root entity. Thus the lock on the root would normally be adequate to lock the attributive entity. This only exception to this would be if the Associative entity were modified through an inverse view of the relationship in a LOD with a different root.
- **Primary** - Primary entities are normally only included and excluded as subordinate entities within a LOD and thus do not need to be locked for LODs where they are not the root.

Note that Pessimistic Locking actually locks the root entity of the LOD. Thus if two LODs allow the same root to be updated (an unusual situation), simultaneous processing of an instance of either LOD with the same root would be allowed unless an entity level pessimistic lock was set on both root entities.

If an associative entity could be processed through an inverse relationship in a different LOD, a pessimistic lock should also be specified for that entity in each LOD. A pessimistic lock for a subordinate entity might also be specified for a fundamental subordinate entity in the unusual case where that entity could be modified within the LOD.

For the discussion of the following locking options, we will use the term *locked entity* to refer to a root entity instance of a LOD with pessimistic locking or to a subordinate entity instance that also has pessimistic locking specified for it in the LOD.

Pessimistic locking requires that the Activate function within the application code indicate update intent using the SingleForUpdate parameter of the Activate statement. If optimistic locking or no locking is specified, then an object instance can be activated through any Activate statement, modified and later committed to the database. But if pessimistic locking is specified, then you must specify the SingleForUpdate parameter of the Activate statement or your Commit statement will be rejected. In addition, any Activate transaction containing a locked entity will be rejected or will return a read-only object instance as explained further below.

To select pessimistic locking for a LOD from the main window of the LOD tool, select **LOD Detail...** from the **LOD** menu. This will present the LOD Detail window. Select the desired pessimistic locking option from the Data Locking combo-box.

To select optimistic locking for an entity from the main window of the LOD tool, open the Entity Detail window as described in [Updating an Entity](#) and then select the desired pessimistic locking option from the Data Locking combo-box.

The pessimistic locking options are:

- **Pessimistic Locking Exclusive** - will lock out any Activates to an object instance containing a locked entity. In other words, a pessimistic lock not only protects against simultaneous updates but keeps any

function from reading the data while it is being processed. This is not the common situation. If an entity is locked exclusively, any other Activate containing that entity is rejected.

- **Pessimistic Locking Non-Exclusive** - will not lock out all Activates to the same object instance but will protect against their being Activated for update. If an entity is locked nonexclusively, any Activate with SingleForUpdate containing that entity will be rejected with a return code of -5 (zLOCK\_ERROR). Any Activate without SingleForUpdate containing that entity will return a read-only object instance.

In addition to selecting Pessimistic locking on the LOD, you must specify the SingleForUpdate option on all Activate statements for object instances for that LOD that are to be updated. If an Activate has any locking specified for the associated LOD and does not have SingleForUpdate specified (ex., ACTIVATE vCustomer Single), then a read-only object instance will be returned and a later Commit will result in an error.



## Overriding Generated SQL

If you want to override the SQL normally generated through a qualification object (the result of an ACTIVATE... WHERE... statement in VML) and specify the SQL SELECT by hand, you can use a technique called OpenSQL. OpenSQL is a way to override the generated SQL SELECT qualification with your own SQL statements. The user is responsible for making sure that the syntax of the specified SQL matches the supported database. Only SELECT statements are supported for OpenSQL.

The OpenSQL SELECT statement is specified using the qualification object that is passed as an argument to ActivateObjectInstance. Thus the user has full control over when OpenSQL is used. The following discussion assumes that the reader is familiar with the Zeidon system object KZDBHQUA (the qualification object).

To specify the OpenSQL when activating a specific entity, create an EntitySpec entity in the qualification object and set the attribute EntitySpec.EntityName to the name of the desired entity. Then set the attribute EntitySpec.OpenSQL to the OpenSQL SELECT statement. The last thing to do is to set the attribute EntitySpec.OpenSQL\_AttributeList to list all the attributes that are being selected. Note that they must be listed in the same order as the columns are listed in the OpenSQL SELECT statement.

Example: The user wants to activate the Dept object and use OpenSQL when activating the root entity (Dept). The user would create a qualification object similar to the following:

ENTITY: EntitySpec

<u>ATTRIBUTES:</u>	<u>VALUES:</u>
--------------------	----------------

EntityName	Dept
------------	------

OpenSQL	select NAME from DEPT where DEPT.NO > 555
---------	---

OpenSQL_AttributeList	Name
-----------------------	------

When activating the entity Dept the DB-Handler will use the OpenSQL instead of the generated SQL.

When specifying OpenSQL for child entities (all non-root entities) you can indicate that you want to use the attribute value of a parent entity in the SELECT statement. Use a '@' followed by the Entity.Attribute qualifier. The DB-Handler will replace the qualifier with the current value of the attribute when the child entity is loaded.

Example: The user wants load the Employee entity using OpenSQL. The qualification object would look similar to:

ENTITY: EntitySpec

<u>ATTRIBUTES:</u>	<u>VALUES:</u>
--------------------	----------------

EntityName	Employee
------------	----------

OpenSQL	select name, position from employee
---------	-------------------------------------

where fk\_deptname = @Dept.Name

OpenSQL\_AttributeList Name, Position

When activating the Employees for department 'Human Resources' the resulting SQL would look like:

select name, position from employee where fk\_deptname = 'Human Resources'

**NOTE:** The attribute names in OpenSQL\_AttributeList are case sensitive.

---

**NOTE:** To use the literal '@' in OpenSQL specify it by using '@@'.

---

## Committing Multiple Object Instances

Committing multiple object instances allows multiple object instances to be committed as a single transaction. This means if there is a failure in the commit of any object instance all will be backed out.

Committing multiple object instances in VML involves three operations:

- `AddViewToViewCluster` which adds a view to the cluster,
- `CommitMultipleObjectInstances` which commits all of the object instances whose views have been added to the cluster using the operation `AddViewToViewCluster`,
- `CreateViewCluster` which initializes an array of views.

The parameters for the three operations are as follows:

- `CreateViewCluster` - Parameter 1: An integer which will point to the `ViewCluster`,
- `AddViewToViewCluster` - Parameter 1: An integer which will point to the `ViewCluster` , Parameter 2: The View to be added to the `ViewCluster`,
- `CommitMultipleObjectInstances` - Parameter 1: An integer which will point to the `ViewCluster`, Parameter 2: An integer variable that will hold the cluster number of the view that causes an error if an error occurs.

The following restrictions apply when committing multiple object instances:

- They must be from the same database.
-

```
AddViewToViewCluster ( ViewCluster, vView2 )  
AddViewToViewCluster ( ViewCluster, vView3 )  
AddViewToViewCluster ( ViewCluster, vView4 )  
CommitMultipleObjectInstances ( ViewCluster, ErrorIndex )
```

END

# Domains

Overview

Starting the Domain Tool

Domain Groups

Domains

Contexts

Table Domains

Algorithm Domains

Format Domains

Expression Domains

## Domains - Overview

The Domain tool is used to create and maintain the domain components of an application. Domains define the set of all valid values an attribute can take, both internally (as the data is stored) and externally (as the data is presented).

Domains have three purposes:

- data validation and conversion
- formatting data for presentation
- comparison and arithmetic operations.

Domains are reusable, which means that a single domain can be assigned to any number of attributes. The validity of a value assigned to an attribute may vary according to the circumstances surrounding its assignment. For example, the date format mm/dd/yy is valid in the United States while dd/mm/yy may be valid in other countries. To support this, one or more domain contexts may be specified for a domain, each defining different external formatting criteria.

Domains are categorized into 4 types: table, algorithm, expression and format.

**Table** - Typically, these are used in situations where the end user must choose or specify a value from a pre-determined table (list) of valid values as in the case of drop-down combo boxes or groups of radio buttons. Each table entry is defined by specifying a valid internal value (i.e. stored value). You can also specify an optional different external value (i.e. presented value) for each table entry.

**Algorithm** - These are used in any situation where the validation or presentation of the data requires the implementation of some specified operation.

**Expression** - Typically, these are used in situations where no data validation is required beyond checking the data type and perhaps the length (e.g. strings) or precision (e.g. decimals) as is the case with simple names, integers or decimals.

**Format** - These domains are used in situations where validation and presentation can be controlled by a user defined template or mask that describes a valid value on a character by character basis. That is, they are useful for cases such as phone numbers where the attribute must be formatted in a pre-defined manner for presentation.

An application is likely to have a large number of domains. Therefore, to facilitate the management of domains, Zeidon organizes domains into domain groups.

## Starting the Domain Tool

To start the Domain tool, double-click on the Domains icon in the Zeidon program group. If there are no domains in your LPLR a New Domain dialog box is presented where you may start by creating a domain. Otherwise, the Open Domain dialog box is presented. From this window you can create a new domain, update an existing domain, delete a domain, or exit the Domain tool.

## Domain Groups

Domain groups are used to manage domains. You can organize your domains into groups in any way that makes sense for your application with only condition. If a particular domain group contains any algorithm domains, the associated operations must all reside in one source file. That is, a domain group can have at most one source file associated with it. Typically, it is most beneficial to organize those domains that are likely to be maintained at the same time together into the same domain group.

### See also:

[Creating a Domain Group](#)

[Updating a Domain Group](#)

[Deleting a Domain Group](#)



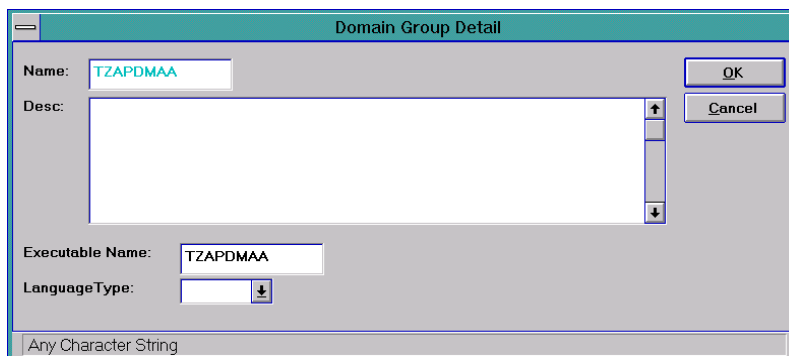
## Creating a Domain Group

We only create a new domain group when creating a new domain. New domains and domain groups are created with the New Domain window. To open this window, do one of the following:

- From the Domain Maintenance window, select **New** from the **File** pull-down menu.
- From the Open Domain window, press the **New** button.

To create a new domain group from the New Domain window, do the following:

1. In the New Domain dialog, right mouse click in the Domain Group list box to display a pop-up menu. Select the **New** item. This opens the Domain Group Detail dialog window.



The screenshot shows the 'Domain Group Detail' dialog box. It has a title bar with a minus sign. Inside, there are two input fields: 'Name' with the text 'TZAPDMAA' and 'Desc' which is a multi-line text area. To the right of these fields are 'OK' and 'Cancel' buttons. Below the 'Name' field is an 'Executable Name' field with the text 'TZAPDMAA'. Below that is a 'LanguageType' field with a drop-down arrow. At the bottom of the dialog is a status bar that says 'Any Character String'.

2. Enter a name for the domain group and an optional description in the text box and multi-line edit box provided.  
NOTE: The name cannot be longer than 8 characters and must be unique to the LPLR.
3. If the domain group is to contain algorithm domains, enter the name of the DLL that will contain the domain group in the edit box. Also, select the language type from the drop-down combo box.
4. Press **OK** to create the domain group or **Cancel** to abort.

## Updating a Domain Group

Existing domain groups are maintained with the Domain Group Detail dialog. This window can be opened in one of three ways:

- To update the domain group for the current domain in the Domain Maintenance window, select the **Group Detail** item from the **Domain Group** pull-down menu.
- Or, to update a domain group other than the one for the current domain in the Domain Maintenance window, select the **Group List** item from the **Domain Group** pull-down menu. This will open the Domain Groups dialog. Double-click on the desired domain group in the list box. Alternatively, highlight the domain group by single-clicking on it and then right-mouse click in the list box. Select **Group Detail** from the pop-up menu.
- Finally, to update a domain group while in the process of creating a new domain, highlight the desired domain group in the Domain Group list in the New Domain dialog. With the cursor inside of the Domain Group list, right mouse click to produce a pop-up menu. Select the **Group Detail** item.

Once the Domain Group Detail dialog has been opened, you can modify the fields as described in [Creating a Domain Group](#).

**NOTE:** if you change the language type for a domain group already containing algorithm domains, you will need to recreate all of the associated operations in the new language.

---

## Deleting a Domain Group

From the Domain Maintenance dialog, select the **Group List** item under the **DomainGroup** pull-down menu to open the Domain Groups dialog. Select the domain group to be deleted by single-clicking on the group name. Then press the **Delete** button.

**NOTE:** use caution! Deleting a domain group will automatically delete all of the domains in the group.

---

## Domains

Certain properties of a domain, such as Type and Data Type must be defined. These essential properties are discussed in [Domain Basics](#). Other properties of a domain depend on which type of domain is being defined. We will discuss each domain type separately.

Also, every domain has to have at least one domain context. Domain contexts control such things as data validation and formatting. A default context is automatically created when a domain is created. If you want a domain to have contexts other than its default context, you should follow the procedures for creating and updating domain contexts in [Contexts](#).

### See also:

[Creating a Domain](#)

[Updating a Domain](#)

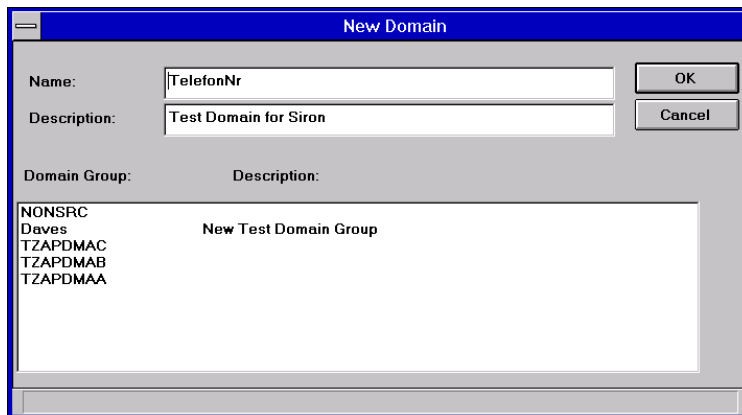
[Domain Basics](#)

[Deleting a Domain](#)

## Creating a Domain

To create a new domain, do the following:

1. From the Open Domain File dialog box, press the **New** button or from the Domain Maintenance window, select **New** from the **File** pull-down menu. In either case, the New Domain dialog box is presented.



Domain Group:	Description:
NONSRC	
Daves	
TZAPDMAC	
TZAPDMAB	
TZAPDMAA	
	New Test Domain Group

2. Enter a name for the new domain and an optional description in the corresponding text boxes.  
NOTE: you must enter a name for the new domain before trying to proceed.
3. Use the Domain Group list box to select a domain group for the new domain by single-clicking on the group name.  
NOTE: if no domain group is present, right mouse click in the Domain Group list box and select **New** from the resulting pop-up menu. Then follow the instructions for creating a new domain group in [Creating a Domain Group](#).
4. Pressing **OK** transfers you to the **Domain Maintenance** window where you can then define the domain information as described in [Domain Basics](#). Or, you can press **Cancel**.

## Updating a Domain

From the Domain Maintenance window, select **Open** from the **File** pull-down menu to open the Open Domain dialog box. Use the list box to select a domain either by double-clicking on the desired domain in the list or by highlighting the domain to be opened and pressing **OK**. The Domain Maintenance window then displays the current information for the selected domain.

## Domain Basics

1. Open or create a domain to be updated as described in [Creating a Domain](#). The following Domain Maintenance window is presented:

Domain Maintenance

File Domain DomainGroup Help

Domain Group: TZAPDMAC

Name: Decimal Type: Expression

Desc: Characters '0-9' precision ( 0.00 )

Data Type: Decimal ( 15 digits )

Maximum String Lth: Decimal Precision: 2

Context Name: Default: DECIMAL DECIMAL0 DECIMAL1 DECIMAL2 DECIMAL3 DECIMAL4 DecimalComma

Operation: zdmDecimal

Prev Next

Edit...

2. Update the name and description in the edit box and multi-line edit box provided.
3. From the Type combo box, select one of the following domain types:

**Table** - The set of valid values for the attribute are defined in a table. Useful for data that will be displayed using drop-down combo boxes or in groups of radio buttons.

**Algorithm** - Invokes a user supplied operation to handle data validation and presentation.

**Format** - Attribute values are presented and input by way of a user defined template (mask) that specifies the formatting on a character by character basis. Useful for things like phone numbers, for instance.

**Expression** - The definition of the domain is made up entirely of the Data Type, Maximum String Lth and Decimal Precision. (Generic strings, names, integers, simple decimals).

4. Select the internal storage type from one of the following in the Data Type combo box:

**String** - Null terminated string. Maximum string length must be specified.

**Number** - An integer in the range  $\pm 2,147,483,647$ .

**Decimal** - A decimal value with a specified precision.

**Date and Time** - Date and time of day.

**BLOb** - Binary Large Object.

5. If the data type is a decimal, enter the number of decimal precision required in the Decimal text box. Note that this field controls the number of digits to the right of decimal that are actually stored internally. You can control the number of digits displayed externally by specifying that separately in the context specifications for the domain.
6. If the data type is a string, you can must specify the maximum length in the Maximum String Lth edit box.
7. You can also add a context to this domain as described [Contexts](#).
8. Restricted is used when a table domain has multiple Contexts. If Restricted is selected, then the validation for an attribute using the domain is restricted to the values in the context used in setting the attribute value. If Restricted is not selected, then the validation for an attribute using the domain is restricted to the values in any context for the domain. For example, consider a domain having two contexts, TypeA and TypeB, with TypeA allowing internal values "1" and "2" and TypeB allowing internal values "2" and "3". If a value of "3" were set using context TypeA, then a validation error would be returned if Restricted is selected for the domain. But, if Restricted is not selected then no validation error would occur, because "3" is allowed for context TypeB. Again, non-Restricted allows an internal value to be specified that is defined in any of the contexts for the domain. Of course, this has no affect on a windowing interface using a combo box, because in that case, only the values for the specified context or default context are listed.
9. Press **OK** to accept the updates or press **Cancel**.



## Deleting a Domain

If you are currently in the Domain Maintenance window, select **Delete** from the **File** pull-down menu. This will open the Open Domain dialog. From the Open Domain dialog box, highlight the domain to be deleted by single-clicking on it and then press the **Delete** button.

**NOTE:** if the domain you are deleting is an operation type domain, you must edit the source file to remove the associated operation from the file. This will not happen automatically.

---

## Contexts

Domain contexts are used to control validation, formatting, comparison, and arithmetic within a domain. Every domain must have at least one context. When you create a domain, a default context is automatically created with the same name as the domain. You can create additional contexts for the domain, each defining different external formatting criteria, along with the associated validation and data conversion information. The default context is used whenever the domain has no other overriding context specified.

### See also:

[Creating and Updating Contexts](#)

[Deleting a Context](#)

[Setting a Default Context](#)

## Creating and Updating Contexts

To create a new context from the Domain Maintenance window, position the cursor in the Context list box and single-click the right mouse button. Select **New** from the pop-up menu presented.

To update an existing context, select a context from the Context list box by doing one of the following:

- Single-click on the desired context to highlight it. Then, with the cursor in the Context list box, single-click the right mouse button. Select **Context Detail** from the pop-up menu presented.
- Or, double-click on the desired context in the Context list box.

Regardless if you are creating a new context or updating an existing one, the following Context dialog box is presented.

The screenshot shows a 'Context' dialog box with the following elements:

- Title Bar:** Context
- Name:** A single-line text input field.
- Desc:** A multi-line text input field.
- Format/Range:** A single-line text input field.
- Precision:** A single-line text input field.
- Message Bar:** A section containing 'Inherited:' and 'Specified:' labels with corresponding input fields.
- Internal Stored Value:** A large multi-line text input field.
- External Display Value:** A large multi-line text input field.
- Buttons:** OK, Cancel, Next, Prev, New, and Delete.
- Status Bar:** QuinSoft Name

**NOTE:** you must enter a Name for the context in the corresponding text box. This is the only field in this dialog that must be completed. This will be used to identify the context to override the default context at the various points at which the attribute value is set.

---

If desired, enter a description for the context in the multi-line edit box.

If the data type for the domain is decimal, use the Precision edit box to enter the number of digits to the right of the decimal you want to be displayed. Note that this specification affect only how the attribute will be displayed. It does not affect how many digits will be actually stored. (The internal storage precision is specified in the Decimal Precision edit box in the Domain Maintenance window as discussed in [Domain Basics](#).) If the precision specified for the context exceeds that which has been specified for the domain, then the attribute will be displayed according to the precision defined for the domain.

The rest of the context specification depends on what the domain type is. Please refer to [Table Domains](#), [Algorithm Domains](#), [Format Domains](#) on page 61 or [Expression Domains](#) for more details.

## Deleting a Context

You can delete a context in one of two ways.

- First, from the Domain Maintenance window, select a context from the Context Name list box by double-clicking on it. This opens the Context window in which you can press **Delete**.
- Alternatively, from the Domain Maintenance window, highlight a context in the Context Name list box by single-clicking on it. Then, with the cursor positioned in the Context Name list box, single-click with the right mouse button to open up a pop-up menu. From this pop-up, select **Delete**.

**NOTE:** You can not delete the default context for a domain!

---

## Setting a Default Context

From the Domain Maintenance window, highlight the context you want to set as the default context in the Context Name list box. Then, position the cursor in the Context Name list box and single-click the right mouse button. Select **Set Default Context** from the pop-up menu presented.

## Table Domains

Table domains allow you to define a table of valid data values, how they are actually stored (their internal values) and to how they are displayed (their external value). Each table entry represents one data element and is made up an internal and an optional external value.

### See also:

[Domain Specification](#)

[Context Specification](#)

## Domain Specification

As with all domains, begin the domain specification as described in [Domains](#).

Next, in the Domain Maintenance window, use the Restricted combo box to select whether or not this domain is to be restricted.

**Y** - Only the specified context table will be used for attribute assignments or comparisons.

**N or Null** - Other context tables for this domain will be searched for attribute assignments or comparisons if the search on the specified context table is unsuccessful.

## Context Specification

The tables (lists) that define a table domain are defined at the context level. That is, a separate table is defined for each context. By choosing different contexts, this allows you to present the same data in different ways, depending on the situation.

As with any context, begin the context specification as described in [Contexts](#). Next, you must define or edit the entries in the table for the context.

### See also:

[Creating Context Table Entries](#)

[Updating Context Table Entries](#)

[Deleting Context Table Entries](#)

[Defining Dynamic Table Domains](#)



## Creating Context Table Entries

From the Domain Maintenance window select the appropriate context in the Context Name list box by double-clicking on it. In the Context window, position the cursor within the list box at the bottom of the window and single-click the right mouse button. Select **New Table Item** from the pop-up menu and in the Table Entry dialog box presented, enter the internal and external values of the table entry.

**NOTE:** you specify one value per entry if only data validation is required and two values if validation and conversion are required.

---

Press the **OK** button when you are done.

## Updating Context Table Entries

As described in [Contexts](#), open the Context window for the related context. In the list box at the bottom of the window, position the cursor on the table entry to be updated and double-click. The Table Entry dialog box is presented and the information may be updated. To save the updates, press **OK**.

## Deleting Context Table Entries

As described in [Contexts](#), open the **Context** window for the related context. Deleting a context table item can then be done in two ways:

- Position the cursor on the table entry to be deleted within the list box at the bottom of the Context window and press the right mouse button. Select **Delete Table Item** from the pop-up menu.
- Alternatively, position the cursor on the table entry to be deleted within the list box at the bottom of the Context window and double-click. In the Table Entry dialog box presented, press **Delete**.

## Defining Dynamic Table Domains

Dynamic table domains are supported automatically by Zeidon to allow the User of an application to specify table values, instead of defining them in the domain as a part of the application definition. The entities necessary in the data model and the associated LOD can be generated from the Data Model tool, as described in that section. To enable your application to use dynamic table domains, you must do the following:

1. Using the Data Model tool, create the domain table entities in the data model and in the associated LOD (it is named DOMAINT) by selecting the **Create Domain Entities + LOD** item from the **Utilities** pull-down menu.
2. Using the Technical Environment tool, create the TE with the new domain table entities by first selecting the **Physical From ER, Init Tables/Records** and then the **Physical From ER, Init Relationships** items from the **Synchronize** pull-down menu. When you are done, save the TE by selecting the **Save Model** item from the **File** pull-down menu.
3. Using the Object tool, save the DOMAINT LOD (which was created by the Data Model tool) with the new TE by first opening the DOMAINT LOD as described in [Opening a LOD for Update](#). Once the LOD is loaded, select the **Technical Environment** item from the **Utilities** pull-down menu.
4. Using the Domain tool, for each domain group that will contain a dynamic table domain, add the operation, `zdmDynamicTable` as described in [Maintaining Operations](#) and set the Executable Name for the domain group to be `TZAPDMAA` as described in [Creating a Domain Group](#).
5. To change a normal table domain to a dynamic table domain, use the Domain tool to select the operation, `zdmDynamicTable`. If you deselect the operation, the domain will be altered back to a normal table domain.

## Algorithm Domains

Domain operations are necessary when the set of all valid values for an attribute cannot be defined through a table or other pre-defined domain type. If you assign an operation to a domain, the operation is triggered whenever an attribute value is set or retrieved, or when a compare or arithmetic operation is executed on an attribute using the domain. An algorithm domain must have exactly one operation assigned to it.

If the data type of your domain is Number or Decimal, you may wish to assign one of Zeidon's internal operations which will allow you to take advantage of their range specification capabilities. Range specification allows you to further delineate what values are valid for the attribute.

### **See also:**

[Maintaining Operations](#)

[Domain Specification](#)

[Context Specification](#)

## Maintaining Operations

Zeidon provides a limited number of pre-defined operations that you can attach to an algorithm domain. In addition, you can also use the Domain tool to create and maintain your own operations which can then be assigned to your domain.

### See also:

[Creating a New Operation](#)

[Editing an Operation](#)

## Creating a New Operation

To create a new operation:

1. From the Domain Maintenance window, select **Operations** under the **Domain Group** pull-down menu to open the Operation List Within Domain Group dialog.
2. Press **New** to open the New Operation dialog.
3. Enter a Name for the operation in the corresponding text box.  
NOTE: the name must not have any spaces in it!
4. You may also enter an optional Description for the operation.
5. Press **OK** to create the operation or press **Cancel** to abort. In either case, you will return to the Operation List Within Domain Group dialog. Press **Close** to return to the Domain Maintenance window.

The above procedure will create a operation stub in the source file corresponding to the current Domain Group. To provide functionality to the operation, you must edit this operation. For more information on editing operations and linking them to a domain refer to [Editing an Operation](#) and [Domain Specification](#).

## Editing an Operation

From the Domain Maintenance window, use the Operation combo box to select the desired operation. Then select **Edit** from the **Domain** pull-down menu.



## Domain Specification

As with all domains, begin the domain specification as described in [Domains](#).

Next, you must assign an operation to the domain.

To assign an operation using the Domain Maintenance window:

1. Select **Group Detail** from the **Domain Group** pull-down menu to open the Domain Group Detail window.
2. If the domain group for the domain does not have a DLL name and language type specified, update the domain group details as described in [Domain Groups](#).  
NOTE: both a DLL name and a language type must be specified.
3. In the Domain Group Detail window, press **OK** to return to the Domain Maintenance window.
4. From the Domain Maintenance window, select the desired operation from the Operation combo box.

## Context Specification

As with any context, begin the context specification as described in [Contexts](#).

If the data type of your domain is Number or Decimal, you can further restrict the valid values for that context by specifying a range of valid values. The range values entered can be used by any operation defined for the domain. In that case, the operation must have the logic necessary to process the range values. More commonly, you will use one of the standard Zeidon operations: `zdmInteger` or `zdmDecimal`.

**NOTE:** When using standard Zeidon operations in conjunction with range editing, the domain group containing the domain cannot contain other operations. This is because the domain group must include the DLL name for the standard Zeidon domains.

---

To use one of the standard Zeidon operations to specify a range:

1. From the Domain Maintenance window, select **Group Detail** from the **Domain Group** pull-down menu to open the Domain Group Detail window.
2. In the Domain Group Detail window, specify the DLL name, TZAPDMAA, if it has not already been defined and then press **OK** to return to the Domain Maintenance window.
3. For the domain group containing the domain, add the operation `zdmInteger` or `zdmDecimal`, if they are not already defined. For more information on adding operations see [Creating a New Operation](#).
4. For the domain, specify a domain type of Algorithm.
5. For the domain, select the operation `zdmInteger` or `zdmDecimal` from the Operation drop-down combo box.
6. Double click on the appropriate context name in the Context Name list box to open the corresponding Context window.
7. For the context of the domain, specify in the Range/Format edit box each range value as an operator (<, <=, =, >, >=), separated by semicolons.

For example: >34,50; <= 3.400,00; >23.45

You can specify the range for decimal domains using either the American standard where periods denote the decimal location and commas denote thousands and etc. or you can use the European standard where commas denote the decimal location and periods denote thousands and etc.

## Format Domains

A format domain is rather like an algorithm domain in that every time an attribute is accessed or presented that is defined as a format domain, a special internal Zeidon operation is executed. This operation controls the validation and presentation of format domains and it is controlled by entering a template (or mask) that describes the formatting on a character by character basis. Since the same data may be presented in different ways in different situations, the formatting masks are defined for each context within a format domain.

### See also:

[Domain Specification](#)

[Context Specification](#)

## Domain Specification

For a format domain, no further domain specification beyond that described in [Domain Basics](#) is required.

## Context Specification

A formatting template is defined in the Format/Range edit box in the Context window for your format domain.

Each character in the format template directly represents one character to be displayed (with the exception of escape codes, which are prefixed by a '\' character). If the character is one of the following edit types, any one of the corresponding legal characters will be permitted. Otherwise, the character is considered to be a constant place holder. The format string is used in the data string to presentation string conversion, as well as the presentation string to data string conversion.

The format characters are:

- 9 - digits from 0 to 9
- A - alphabetic
- N - alphanumeric
- X - ascii
- U - upper case
- L - lower case
- H - hex digit 0 - F

For example, to specify a format for phone number (U.S. standard) you would enter:

(999) 999-9999

where the parentheses, spaces and the dash will act as place holders.

**NOTE:** Format domains support only attribute types of string and should have no operation associated with them.

---

## Expression Domains

Expression domains are the simplest domains in that they provide no validation or formatting beyond that which is specified by the data type, string length and decimal precision as described in [Domain Basics](#). If you wish, you can create more than one context for the domain as described in [Creating and Updating Contexts](#).

# Global Operations

Overview

Starting the Operations Tool

Operation Groups

Creating an Operation

Updating an Operation

Deleting an Operation

## **Global Operations - Overview**

The Global Operation tool is used to create and maintain global operations; that is, operations not tied to a dialog, LOD or domain. One may think of Zeidon global operations as being the equivalent of general operations in a non-object oriented environment.



## **Starting the Operations Tool**

To start the Operations tool, double-click on the Operations icon in the Zeidon program group. This will open the Open Operation window. From this window you can create a new domain, update an existing domain, delete a domain, or exit the Domain tool.

## Operation Groups

Operation groups are used to manage operations. You can organized your operations into groups in any way that makes sense for your application with only condition. An operation group can have at most one source file associated with it. Typically, it is most beneficial to organize those operations that are likely to be maintained at the same time together into the same operation group.

### **See also:**

[Creating an Operation Group](#)

[Updating an Operation Group](#)

[Deleting an Operation Group](#)

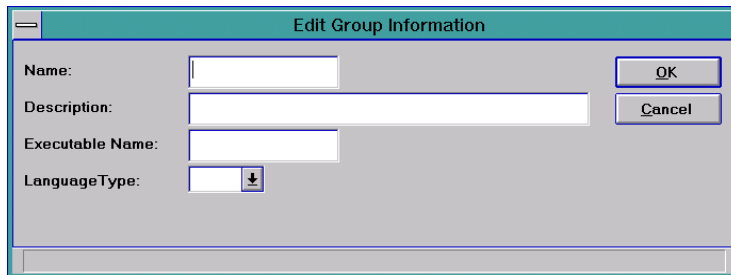
## Creating an Operation Group

We only create a new operation group when creating a new operation. New operations and operation groups are created with the New Operation window. To open this window, do one of the following:

- From the Operation window, select **New** from the **File** pull-down menu.
- From the Open Operation window, press the **New** button.

To create a new operation group from the New Operation window, do the following:

1. In the New Operation dialog, right mouse click in the Operation Group list box to display a pop-up menu. Select the **New** item. This opens the Edit Group Information dialog window.



The screenshot shows a dialog box titled "Edit Group Information". It has a light gray background and a blue border. Inside, there are four labeled input fields: "Name:" (a small text box), "Description:" (a larger text box), "Executable Name:" (a text box), and "LanguageType:" (a drop-down menu with a downward arrow). To the right of these fields are two buttons: "OK" and "Cancel".

2. Enter a name for the operation group and an optional description in the text boxes provided.  
NOTE: The name cannot be longer than 8 characters and must be unique to the LPLR.
3. Enter the name of the DLL that will contain the operation group in the edit box. Also, select the language type from the drop-down combo box.
4. Press **OK** to create the operation group or **Cancel** to abort.



## Deleting an Operation Group

From the Operation dialog, select the **Group List** item under the **OperationGroup** pull-down menu to open the Operation Groups dialog. Select the operation group to be deleted by single-clicking on the group name. Then press the **Delete** button.


**NOTE:** use caution! Deleting a operation group will automatically delete all of the operations in the group.

---

## Creating an Operation

To create a new operation, do the following:

1. From the Open Operation File dialog box, press the **New** button or from the Operation window, select **New** from the **File** pull-down menu. In either case, the New Global Operation dialog box is presented.

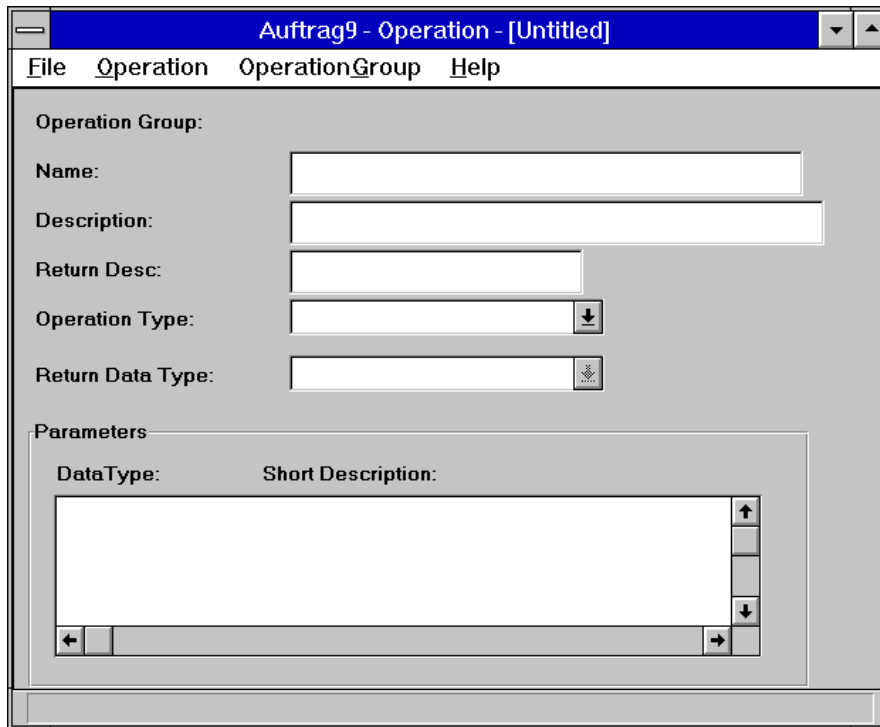


2. Enter a name for the new operation and an optional description in the corresponding text boxes.  
NOTE: you must enter a name for the new operation before trying to proceed.
3. Use the Operation Group list box to select a operation group for the new operation by single-clicking on the group name.  
NOTE: if no operation group is present, right mouse click in the Operation Group list box and select **New** from the resulting pop-up menu. Then follow the instructions for creating a new operation group in [Creating an Operation Group](#).
4. Pressing **OK** transfers you to the **Operation** window where you can then define the operation information as described in [Updating an Operation](#). Or, you can press **Cancel**.

## Updating an Operation

To update an operation:

1. Select **Open** from the **File** pull-down menu from the Operation window to open the Open Operation dialog box. Use the list box to select a operation either by double-clicking on the desired operation in the list or by highlighting the operation to be opened and pressing **OK**. The Operation window then displays the current information for the selected operation.



2. In the Operation window, enter a name and, optionally, a description for the operation, as well as the following information:

**Operation Type** - There are different types of operations which may be defined. Select the operation type from the Operation Type combo box. The choices are:

**Global Operation** - An operation called from another source file.

**Local Operation** - An operation called from within the source file.

**ShortcutKey Operation** - An operation triggered by a shortcut key.

**Return Data Type** - Use the combo-box to select the data type for the return value.

**Return Description** - Enter an optional description of the return value for the operation.

3. If desired, you can modify, add to and delete the default parameters by doing the following:

**Modify a Parameter** - Select the parameter by single-clicking on it in the Parameters list box. With the cursor in the list box, single-click the right mouse button to display a pop-up menu. Select **Parameter Detail** and the Parameter Maintenance dialog is presented. From this dialog, you can change:

**Description** - Enter a description for the parameter in the edit box provided.

**Data Type** - Specify the data type for the parameter by using the pull-down combo box.

**Returned Value** - If the parameter is to be modified by the operation and returned to the caller, choose **Y** (yes). Otherwise, choose **N** (no).

**Unsigned** - If the parameter is a numeric value that is to be treated as an unsigned value, choose **Y** (yes). Otherwise, choose **N** (no).

If there is more than one parameter you wish to update, press **Next** or **Prev** to cycle through the other parameters. Otherwise, you can press **OK** to accept the modifications or you can press **Cancel** to abort.

**Add a New Parameter** - Single-click the right mouse button while the cursor is in Parameters list box. Select the **New** item from the pop-up menu presented. The Parameter Maintenance dialog is presented. Complete the fields of this dialog as described immediately above.

**Delete a Parameter** - Delete a parameter in one of two ways.

- From the Operation window, select the parameter from the Parameters list box by single-clicking on it. With the cursor in the list box, single-click the right mouse button to present the pop-up menu and select the **Delete** item.
  - From the Parameter Maintenance dialog, press **Delete**.
4. If you want to enter the Editor to edit the operation, highlight the operation and select the **Edit** button in the Operation window.
  5. When you are done with the updates, press **OK** to accept them or press **Cancel**



## Deleting an Operation

If you are currently in the Operation window, select **Delete** from the **File** pull-down menu. This will open the Open Operation dialog. From the Open Operation dialog box, highlight the Operation to be deleted by single-clicking on it and then press the **Delete** button.

**NOTE:** you must edit the source file to remove the associated operation from the file. This will not happen automatically.

---

# Dialogs

Overview

Starting Zeidon's Dialog Tool

Creating a Zeidon Dialog

Opening a Zeidon Dialog for Update

Dialog Details

Registered Views

Dialog Operations

Windows

Window Actions

Menus

Shortcut Keys

Window Controls

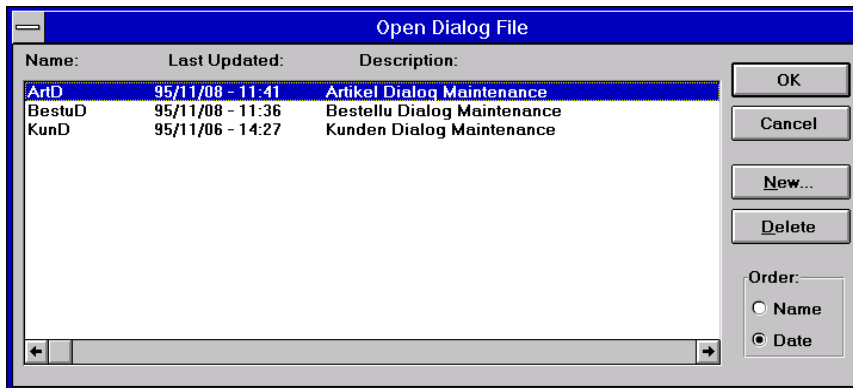
Autodesign

## **Dialogs - Overview**

Zeidon's Dialog tool is used to create and maintain windowing dialogs. A windowing dialog is a group of related windows which interact together to perform some application functionality. The Zeidon Dialog tool allows you to create dialog windows, paint the window controls, define actions for the control and window events, and specify data mapping from and to the windows from Zeidon Objects. In addition, since many dialogs are focused around the manipulation of data, the tool has the powerful capability of auto-designing a complete dialog based upon Zeidon Objects.

## Starting Zeidon's Dialog Tool

To start the Dialog tool, double-click on the Dialog icon in the Zeidon program group. A Open Dialog File dialog box will be presented where you can choose to create, update or delete dialogs.



## Creating a Zeidon Dialog

You begin creating a new dialog by one of the following:

- Select **New** from the **File** pull-down menu of the Dialog window.
- From the Open Dialog dialog box press the **New** button.

In either case, a new Dialog dialog box is presented.



The image shows a standard Windows-style dialog box titled "New Dialog". It features a blue title bar. The main area has a light gray background. On the left, there are two labels: "Name:" and "Description:". Next to "Name:" is a single-line text input field. Next to "Description:" is a multi-line text input field. On the right side of the dialog, there are two buttons: "OK" and "Cancel", stacked vertically.

Enter a name and, optionally, a description of the dialog in the corresponding text boxes and press **OK**.

A dialog box is presented which is the Window Image. This is where you paint your windows controls, define menus, specify mapping and define the actions for the window and its associated controls. The window name is initialized to the dialog name. The dialog is now open for update.

## Opening a Zeidon Dialog for Update

To open a dialog for update by doing one of the following:

- Select **Open** from the **File** pull-down menu of the Dialog window.
- From the Open Dialog window, select the dialog to be opened by clicking on it and then pressing the **OK** button, or by double-clicking on the dialog.

If the dialog opened has only one window defined for it, a window design dialog box for that window will be presented along with a Window List dialog box. Otherwise, only the Window List dialog box is presented.

When a dialog is open it may or may not be open for update. A Zeidon Dialog will not be updateable unless it has been checked out to your workstation using the Workstation Administration tool or you have created it on your workstation and have not yet checked it into the repository.

## Dialog Details

To specify or update the dialog detail information, select the **Dialog Detail** item from the **Dialog** pull-down menu of the Dialog window. The Dialog Detail dialog box is presented.



The screenshot shows a standard Windows-style dialog box titled "Dialog Detail". It has a blue title bar. The main area is light gray and contains three labeled input fields. The first field is "DLL Name:" followed by an empty text box. The second field is "Description:" followed by a text box containing the text "Artikel Dialog Maintenance". The third field is "Primary Window:" followed by a text box containing "AutoDsgnLoadWindow" and a small downward-pointing arrow icon, indicating a dropdown menu. To the right of these fields are two buttons: "OK" and "Cancel".

The following may be modified:

**DLL Name** - Enter the name of the executable DLL containing the operations for this dialog.

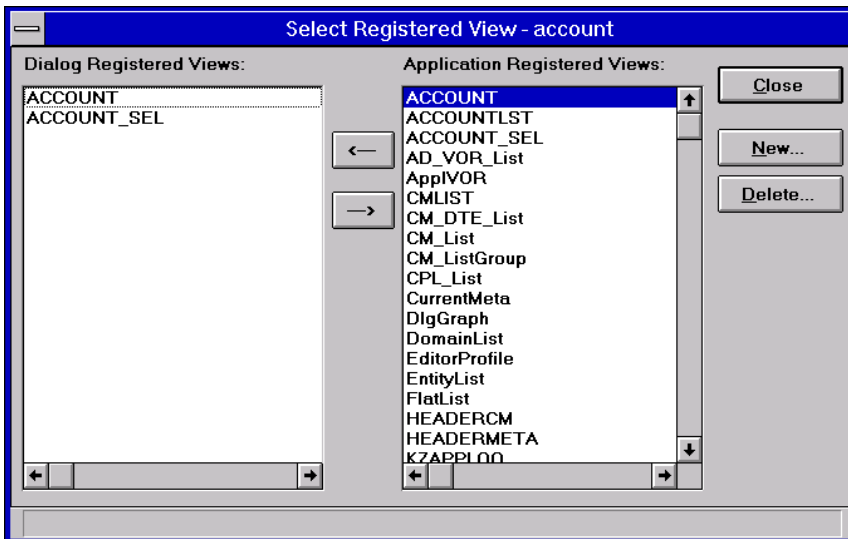
**Description** - Enter a short description for the dialog in the corresponding text box. This field is optional

**Primary Window** - From the drop-down combo box select, the window which will be the first window, or dialog box, presented when this dialog is started from an icon.

## Registered Views

For a window control, such as a list box, to display data from an object instance, a name must exist to identify which object instance and which view to that object instance are to be used by the control. All object views that are referenced within a window must have names assigned to them that can be referenced by each window control. In Zeidon, these names are called registered views. All names, or views, must be registered to the application. To use a view within a dialog, that view must be registered to the dialog.

To list and maintain registered views, select the **Registered Views** item from the **Dialog** pull-down menu of the Dialog window. This presents the Select Registered View dialog box.



The dialog box has two list boxes, one listing views registered to the Dialog and the other listing views registered to the Application. When a LOD is created, a view of that object is registered to the application with the same name as the LOD.

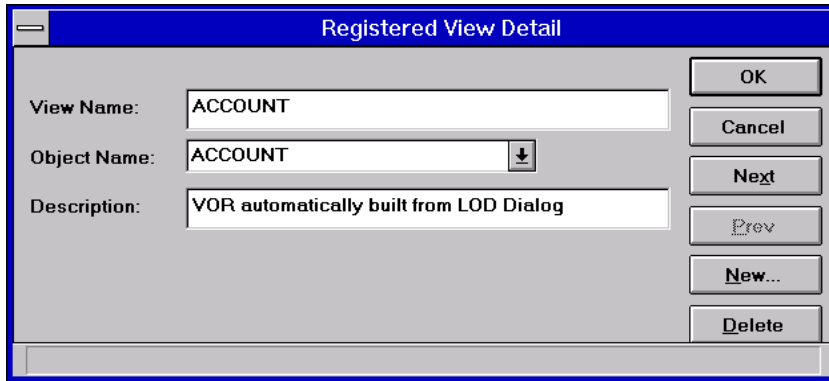
### See also:

- [Updating a View Registered to an Application](#)
- [Adding a Registered View to an Application](#)
- [Deleting a View Registered to an Application](#)
- [Registering a View to a Dialog](#)
- [Removing a Registered View from a Dialog](#)



## Updating a View Registered to an Application

To update a view which is registered to an application, open the Select Registered View dialog box as described in the preceding section. From the Select Registered View dialog box, double-click on the view to be updated in the Application **Registered Views** list box. The Registered View Detail dialog box is presented.



The image shows a dialog box titled "Registered View Detail". It has a blue title bar with a minus sign icon on the left. The dialog contains three input fields on the left and a vertical stack of buttons on the right. The "View Name" field contains the text "ACCOUNT". The "Object Name" field contains "ACCOUNT" and has a small downward arrow icon to its right. The "Description" field contains the text "VOR automatically built from LOD Dialog". The buttons on the right, from top to bottom, are "OK", "Cancel", "Next", "Prev", "New...", and "Delete".

The following may be modified:

**View Name** - Provide a name to identify this view.

**Object Name** - From the drop-down combo box select the Zeidon Object which this is to be a view of.

**Description** - Provide an optional description of the view.

## Adding a Registered View to an Application

To add a new registered view to an application, do one of the following:

- Open the Select Registered View dialog box by selecting the **Registered Views** item from the **Dialog** pull-down menu of the Dialog window. Press the **New** button.
- From the Registered View Detail dialog box, press **New**.

In either case, an empty Registered View Detail dialog box is presented. Complete the fields of this dialog box as described in [Updating a View Registered to an Application](#)

## Deleting a View Registered to an Application

To delete a view registered to an application in one of the following ways:

- Open the Select Registered View dialog box by selecting the **Registered Views** item from the **Dialog** pull-down menu of the Dialog window. Select the application view to be deleted by single-clicking on it in the Application Registered Views list box and then press the **Delete** button.
- Open the Registered View Detail dialog by double-clicking on the selected view in the Application Registered Views list box in the Select Registered View window. From the Registered View Detail dialog box press the **Delete** button.

## Registering a View to a Dialog

1. From the Dialog window, select the **Registered Views** item under the **Dialog** pull-down menu. This will open the Select Registered View dialog box.
2. Select the view to be assigned by single-clicking on the view in the Application Registered Views list box. This will highlight the view in the list box.
3. Press the arrow button (<-) pointing to the Dialog Registered Views list box.

## **Removing a Registered View from a Dialog**

1. Open the Select Registered View dialog box as described in the previous section.
2. Select the view to be unregistered by single-clicking on the view the Dialog Registered Views list box. This will highlight the view in the list box.
3. Press the arrow button (->) pointing to the Application Registered Views list box.

## Dialog Operations

A Zeidon Dialog can have many operations defined for it. In order to assign an operation to a dialog, the dialog must have a Source file assigned to it. The Source file is where the operation is maintained. Multiple source files can be assigned to a dialog and thus different operations for a dialog may be maintained in different source files.

### **See also:**

[Creating a New Dialog Source File](#)

[Updating a Dialog Source File](#)

[Deleting a Dialog Source File](#)

[Creating a New Operation](#)

[Updating Operation Details](#)

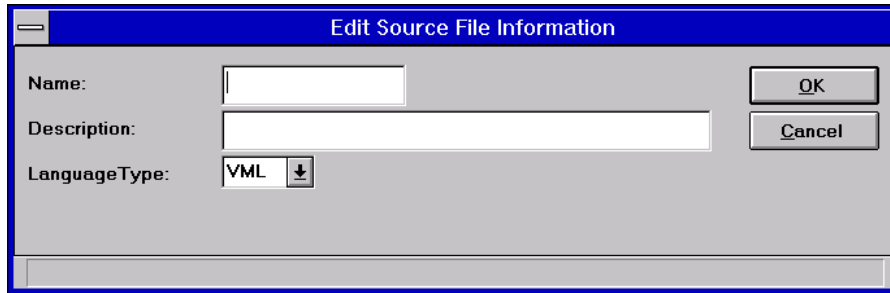
[Editing an Operation Source](#)

[Deleting an Operation](#)

## Creating a New Dialog Source File

To create a source file for a dialog:

1. From the Dialog window, select the **Source Files** item under the **Dialog** pull-down menu. This will open the Source File Maintenance dialog.
2. In the Source File Maintenance dialog, press **New** to open the Edit Source File Information dialog.



The screenshot shows a dialog box titled "Edit Source File Information". It has a blue title bar. Inside, there are three labels: "Name:", "Description:", and "LanguageType:". The "Name:" label is followed by a text input box. The "Description:" label is followed by a larger text input box. The "LanguageType:" label is followed by a dropdown menu showing "VML" and a downward arrow. To the right of the input boxes are two buttons: "OK" and "Cancel".

3. Enter a name for the source file in the edit box.  
NOTE: the name must follow any naming conventions for the operating system and must be unique with respect to this LPLR.
4. If desired, enter a short description for the source file.
5. Select the language type for the source file using the combo-box.
6. Press **OK** to create the file or **Cancel** to abort.

## Updating a Dialog Source File

To update a dialog source file:

1. From the Dialog window, select the **Source Files** item from the **Dialog** pull-down menu. This will open the Source File Maintenance dialog window.
2. In the Source File Name list box, double-click on the name of the source file to be updated. This will open the Edit Source File Information dialog.
3. Update the fields of this window as described above in [Creating a New Dialog Source File](#).



## Deleting a Dialog Source File

**NOTE:** you should only attempt to delete a source file if it contains no operations for the dialog. If it contains operations for the dialog, you should first delete these operations as described in [Deleting an Operation](#).

---

To delete a source file for a dialog:

1. From the Dialog window, open the Source File Maintenance dialog by selecting the **Source File** item under the **Dialog** pull-down menu.
2. Using the Source File Name list box, single-click on the file to be deleted. This will highlight the source file in the list box.
3. Press **Delete** to delete the source file or press **Cancel** to abort.

## Creating a New Operation

**NOTE:** There must be a source file assigned to the dialog in order to create an operation. If no source file exists, follow the directions in [Creating a New Dialog Source File](#).

---

To create a new operation for the dialog:

1. From the Dialog window, select **Operations** from the **Dialog** pull-down menu. The Operation Maintenance window is displayed.
2. Press the **New** button. The Select Source File for Operation dialog box is displayed.
3. Using the list box, select a source file by single-clicking on the file name and pressing **OK**. Alternatively, you can press **Cancel** to abort. If you select **OK**, the Operation Detail window is presented.
4. Complete the operation details as described in [Updating Operation Details](#).
5. If you want to enter the Editor to edit the newly created operation, highlight the operation and select the **Edit** button in the Operation Detail window.
6. Select **OK** to complete the create and to return to the Operation Maintenance window or press **Cancel** to drop it.

## Updating Operation Details

To update a dialog operation detail:

1. From the Dialog window, select **Operations** from the **Dialog** pull-down menu. The Operation Maintenance window is displayed.
2. Double-click the operation to be updated from the operation list. The Operation Detail dialog box is displayed.

The screenshot shows the 'Operation Detail' dialog box. It has a title bar with the text 'Operation Detail'. Inside the dialog, there are several input fields and buttons. The 'Source File:' field contains the text 'ArtD'. Below it are fields for 'Name:', 'Description:', 'Return Desc:', 'Operation Type:', and 'Return Data Type:'. The 'Operation Type:' and 'Return Data Type:' fields are dropdown menus. To the right of these fields are buttons for 'OK', 'Cancel', 'Next', 'Prev', 'New', and 'Delete'. At the bottom of the dialog is a 'Parameters' section. It contains a table with two columns: 'DataType:' and 'Short Description:'. Below the table are four small buttons: a left arrow, a right arrow, an up arrow, and a down arrow.

3. In the Operation Detail window, enter a name and, optionally, a description for the operation, as well as the following information:

**Operation Type** - There are different types of operations which may be assigned to a Dialog. Select the operation type from the Operation Type combo box. The choices are:

**Dialog Operation** - May be invoked by an action(s) defined for window and/or control events.

**ShortcutKey Operation** - May be invoked for a defined shortcut key.

**Local Operation** - May be invoked from other Zeidon operations within the same source file

**Return Data Type** - Use the combo-box to select the data type for the return value.

**Return Description** - Enter an optional description of the return value for the operation.

4. If desired, you can modify, add to and delete the default parameters by doing the following:

**Modify a Parameter** - Select the parameter by single-clicking on it in the Parameters list box. With the cursor in the list box, single-click the right mouse button to display a pop-up menu. Select **Parameter Detail** and the Parameter Maintenance dialog is presented. From this dialog, you can change:

**Description** - Enter a description for the parameter in the edit box provided.

**Data Type** - Specify the data type for the parameter by using the pull-down combo box.

**Returned Value** - If the parameter is to be modified by the operation and returned to the caller, choose **Y** (yes). Otherwise, choose **N** (no).

**Unsigned** - If the parameter is a numeric value that is to be treated as an unsigned value, choose **Y** (yes). Otherwise, choose **N** (no).

If there is more than one parameter you wish to update, press **Next** or **Prev** to cycle through the other parameters. Otherwise, you can press **OK** to accept the modifications or you can press **Cancel** to abort.

**Add a New Parameter** - Single-click the right mouse button while the cursor is in Parameters list box. Select the **New** item from the pop-up menu presented. The Parameter Maintenance dialog is presented. Complete the fields of this dialog as described immediately above.

**Delete a Parameter** - Delete a parameter in one of two ways.

- From the Operation Detail window, select the parameter from the Parameters list box by single-clicking on it. With the cursor in the list box, single-click the right mouse button to present the pop-up menu and select the **Delete** item.
  - From the Parameter Maintenance dialog, press **Delete**.
5. If you want to enter the Editor to edit the operation, highlight the operation and select the **Edit** button in the Operation Detail window.
  6. If you wish to update the details of several operations in one session, you can use the **Next** and **Prev** buttons in the Operation Detail dialog to cycle through the operation list for the dialog. When you are done with the updates, press **OK** to accept them or press **Cancel**.

## Editing an Operation Source

To edit the source for an existing operation:

1. From the Dialog window, select **Operations** from the **Dialog** pull-down menu. The Operation Maintenance window is displayed.
2. Single-click on the operation in the Operation Name list to highlight it.
3. Press the **Edit** button to enter the system editor.

## Deleting an Operation

To delete a dialog operation:

1. From the Dialog window, select **Operations** from the **Dialog** pull-down menu. The Operation Maintenance window is displayed.
2. Select the operation from the list to be deleted by single-clicking on it.
3. Press the **Delete** button.

# Windows

Windows provide the interface between user and data. As such, they are a primary component of a Zeidon Dialog. A window itself is comprised of many components, including controls such as edit boxes and buttons, menu items and events such as hot keys. Windows can also have actions which define an activity that can be assigned to menu items, control events or window events.

## See also:

[Creating a Window](#)

[Updating Window Detail Information](#)

[Copying Windows](#)

[Merging Windows](#)

[Deleting Windows](#)

## Creating a Window

When a new dialog is created, the Dialog tool creates an initial window. To create new windows after the initial window is created:

1. If the Window List dialog box is not displayed, open it by selecting the **Window List** item from the **Window** pull-down menu in the Dialog window.
2. Right mouse click in the Window List dialog box and select the **New** item from the pop-up menu.

A Window Image is presented where you can paint and design the new window as described in the following sections of this chapter.



## Updating Window Detail Information

Detail information for a window includes its name, caption, style, and events. This information is defined in the Window Detail dialog box. To update the window detail information:

1. If the Window List is not displayed, open it by selecting the **Window List** item under the **Dialog** pull-down menu in the Dialog window.
2. In the Window List window, select the window to be updated by single-clicking on its name.
3. The Window Detail dialog box can now be opened in two ways:
  - Select the **Window Detail** item from the **Window** pull-down menu of the Dialog window.
  - From the Window List dialog, box right mouse click and select the **Window Detail** item from the pop-up menu.

In either case, the Window Detail dialog box will be presented.

**NOTE:** if the Window List dialog box is not displayed and you select the **Window Detail** item from the **Dialog** menu item, the detail information display in the Window Detail dialog will be for the window whose Window Image had focus most recently.

---

4. In the Window Detail dialog box the following information may be updated or defined:

**Name** - Specify a name to identify the window.

**Caption** - Enter text that is to be displayed in the caption bar of the window.

**Default Button** - Enter the tag name of the button on this window that is to be the default. The window's default button is pushed when the user presses the enter key.

**Style** - Select the Window Style from the drop-down combo box. The choices are:

**Dialog box** - Standard dialog box frame with system menu and caption Bar. It has no menu and is non-sizable.

**Frame** - Simple frame with no caption or menu.

**Primary Window** - Sizable window with system menu, caption bar, minimize and maximize button, and a possible application menu bar.

**Sizable Dialog Box** - Sizable dialog box.

**Toolbox per CW** - Stay on top dialog box.

**Actions for Events** - The list box lists the events for a window and any action which is assigned to that window action. How actions for the window events may be defined is described in [Window Actions](#).

5. Once the fields of the Window Detail dialog have been updated, press **Close**.

## Copying Windows

There are two methods to copy windows depending on whether or not the window to be copied is in the current dialog or not.

### See also:

[Copying Within a Dialog](#)

[Copying From Another Dialog](#)

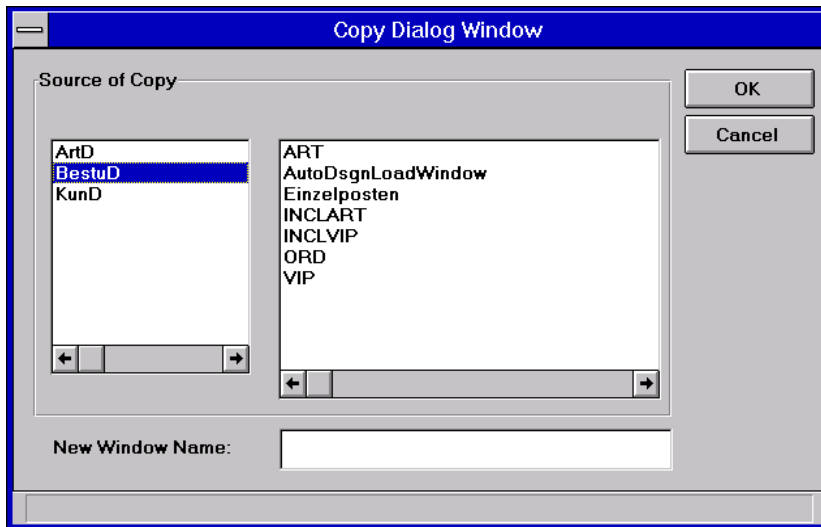
## Copying Within a Dialog

To copy a window in the current dialog:

1. Bring up the Window List dialog box, if it is not already open, by selecting **Window List** from the **Dialog** pull-down menu of the Dialog window.
2. In the window list, highlight the window to be copied and press the right mouse button. From the pop-menu select **Copy**. The window will be copied and assigned a default name.

## Copying From Another Dialog

1. Select the **Copy Dialog Window** item from the **Utilities** pull-down menu of the Dialog window. This presents the Copy Dialog Window dialog box.



2. Select the dialog which contains the window you wish to copy from the Dialog list box by single-clicking on it. The Window list box will now list all windows from that dialog.
3. Select the desired window with a single-click. The window will be highlighted.
4. Enter a New Window Name for the copy in the text box provided.
5. Press the **OK** button to complete the copy or press **Cancel** to abort.

## Merging Windows

Merging a source window into a target window will copy all the window controls from the source into the target while retaining all the current controls of the target. A merge will not affect the source window in any way.

To merge one window into another:

1. Select the target dialog window you wish to merge into by either single-clicking on the window in the Window List dialog box, or by single-clicking on the window's Window Image.
2. Select the **Merge Dialog Window** item from the **Utilities** pull-down menu of the Dialog window. The Copy Dialog Window dialog box is presented.
3. Select the dialog which contains the source window you wish to copy from by single-clicking on the dialog name in the Dialog list box. The Window list box will now list all windows from that dialog.
4. Select the desired source window with a single-click and press the **OK** button.

## Deleting Windows

To delete a window:

1. Bring up the Window List dialog box, if it is not already presented, by selecting **Window List** from the **Dialog** pull-down menu of the Dialog window.
2. In the window list, highlight the window to be deleted by single-clicking on it.
3. With the cursor position in the window list, single-click the right mouse button. From the pop-up menu presented, select **Delete**.

## Window Actions

Window actions define the activity that can be assigned to menu options, window events or control events. Any action can be tied to multiple options or events within a window providing a level of reusability within a window.

### See also:

[Assigning an Action to a Window Event](#)

[Removing an Assigned Action from a Window Event](#)

[Creating Window Actions](#)

[Updating Window Actions](#)

[Deleting Window Actions](#)

[Disabling Window Actions](#)



## Assigning an Action to a Window Event

A Window's events are listed in the Window Detail dialog box. To open this dialog box refer to [Updating Window Detail Information](#).

Window events include:

**Post-Build** - After the window has been built.

**Pre-Build** - Prior to the creation of a window.

**Receive Focus** - When focus is given to the window.

**Refresh Window** - When a request for a refresh of the window has occurred. (Refresh resets the contents of the window controls from the current object instance values.)

**Return from Subwindow** - When a subwindow closes.

**System Close** - When a close message is received by the window, (e.g. Alt-F4, or the close option from the window's system menu.)

To assign an Action to a window event:

1. Double-click on the event in the list box. A Select Action dialog box is presented.
2. Double-click on the desired action from the list or single-click on it to select the action and press the **OK** button.

See [Creating Window Actions](#) for details on how to add actions to a window.

## Removing an Assigned Action from a Window Event

Window actions are un-assigned from a window event through the Window Detail dialog box. To present this dialog box refer to [Updating Window Detail Information](#).

From that dialog box:

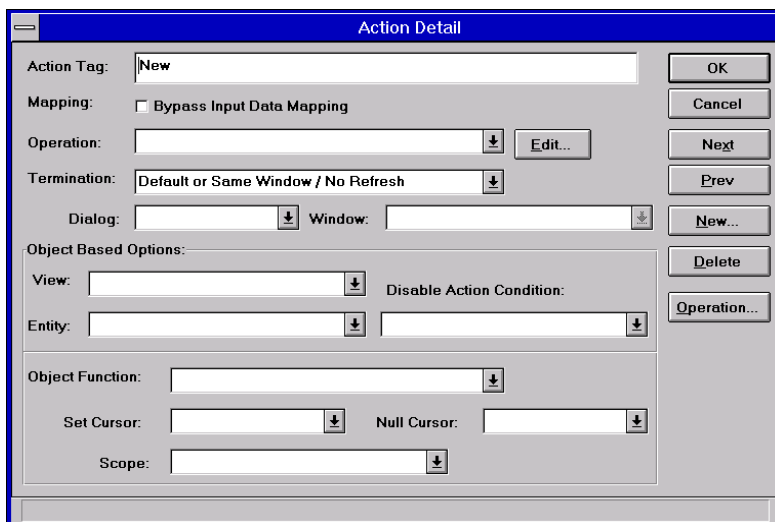
1. Double-click on the event in the list box. A Select Action dialog box is presented in which the associated action is already highlighted in the list box.
2. Press the **Remove** button.

## Creating Window Actions

The dialog box where a window action is created is the Action Detail dialog box. There are a number of ways to open the Action Detail dialog box:

- Select the dialog window you wish to create an action for by either clicking on the window in the Window List dialog box, or clicking on the window's Window Image. Select the **Action List** item from the **Window** pull-down menu of the Dialog window. The Action List dialog box is presented. Press the **New** button.
- Right mouse on the dialog window you wish to create an action for in the Window List dialog box. Select Actions List from the pop-up menu. The Action List dialog box is presented. Press the **New** button.
- Press the **New** button in the Select Action dialog box. This dialog box is presented while assigning actions to events, such as a window event or a control (push button, radio button, list box, etc.) event. For details on assigning an action to a window event, see [Assigning an Action to a Window Event](#). Assigning actions to control events is discussed in the individual controls topics.

In any case, the Action Detail window is opened.



The screenshot shows the 'Action Detail' dialog box. It has a blue title bar and a grey body. The 'Action Tag' field is set to 'New'. The 'Mapping' section has a checkbox for 'Bypass Input Data Mapping'. The 'Operation' field is empty with a dropdown arrow and an 'Edit...' button. The 'Termination' field is set to 'Default or Same Window / No Refresh' with a dropdown arrow. Below these are 'Dialog' and 'Window' fields, both with dropdown arrows. The 'Object Based Options' section contains 'View' and 'Entity' fields with dropdown arrows, and a 'Disable Action Condition' field. The 'Object Function' field has a dropdown arrow. The 'Set Cursor' and 'Null Cursor' fields have dropdown arrows. The 'Scope' field has a dropdown arrow. On the right side, there are buttons for 'OK', 'Cancel', 'Next', 'Prev', 'New...', 'Delete', and 'Operation...'.

**See also:**

[Action Details](#)

## Action Details

When the Action Detail dialog box is presented, detail information regarding that action can be specified as follows:

**Action Tag** - Key in a name to identify the action in the text box provide.

**Mapping** - Click this check box if you wish mapping from the window to be bypassed when this action is invoked.

**Operation** - If needed, select an operation from the pull-down list to be invoked with this action. Once an operation has been selected, it may be edited by pressing the **Edit** button.

**Termination** - Select, from the pull-down list, the window processing which occurs at the end of the action. The options include:

- Default or Same Window / No Refresh - Stay on the current window, but don't refresh the window.
- Same Window / Refresh - Stay on the current window and refresh.
- Subwindow / Modal - Start the specified modal window.
- Subwindow / Modeless - Start the specified modeless window.
- Set Focus or Start Modeless Window - Set focus to the specified modeless window and if it is not found start it.
- Set Focus or Start Modeless Subwindow - Set focus to the specified modeless subwindow and if it is not found start it.
- Replace Window / Modal - Replace the current window with the specified modal window.
- Replace Window / Modeless - Replace the current window with the specified modeless window.
- Return / No Refresh - Return to the parent window without refreshing it.
- Return / Refresh - Return to the parent window and refresh it.
- Return Top / No Refresh - Return to the top level parent window of the dialog without refreshing it.
- Return Top / Refresh - Return to the top level parent window of the dialog and refresh it.
- Exit Dialog Task - End the dialog.

**Dialog and Window** - If the window processing selected is a Subwindow, Replace Window, or Set Focus or Start Modeless types, specify the Window name and the Dialog name that control is to be passed to.

**NOTE:** a DialoT10 0 0 10 60.70 Tm/TT1 1 Tf-0.0382ot be specifi038if the Window is in the current 1 Tf.10 87 766.95 Tm/TT.

- Stay When Null
- Return When Null
- Scope - Select the scoping entity for the subsequent cursor positioning from the Scope pull-down list.

**Operation** - click this button to bring up the Operation Maintenance dialog box. For details on operation maintenance see [Creating a New Operation](#).

Once the action details have been completed, press **OK** to complete the creation or press **Cancel** to abort.

## Updating Window Actions

The dialog box where a window action is updated is the Action Detail dialog box. There are a number of ways to open it:

- Select the dialog window containing the action you wish to update, by either clicking on the window in the Window List dialog box or by clicking on the window's Window Image. Select the **Action List** item from the **Window** Pull-down menu of the Dialog window. The Action List dialog box is presented. Double-click on the action to be updated.
- Right mouse click on the dialog window you wish to create an action for in the Window List dialog box. Select **Action List** from the pop-up menu. The Action List dialog box is presented. Double-click on the action to be updated.
- In the course of assigning actions to events, such as a window event or a control event (e.g. push button, radio button, list box), the Select Action dialog box will sometimes be presented. Double-click on the action to be updated in the Select Action dialog box. For detailed on assigning an action to a window event see [Assigning an Action to a Window Event](#).

When the Action Detail dialog box is presented, detail information regarding that action can be specified as defined in [Creating Window Actions](#).

**NOTE:** when updating several actions at once, you can spool through the current list of actions while in the Action Detail dialog by pressing the **Next** and **Prev** buttons.

---

## Deleting Window Actions

A window action can be deleted from either the Action List or Action Detail dialog boxes:

- In the Action List dialog box, single-click on the action to be deleted and press the **Delete** button.
- In the Action Detail dialog box simply press the **Delete** button.

Refer to [Updating Window Actions](#) for details on how to bring up these dialog boxes.



## Disabling Window Actions

During the execution of an application, an action can be dynamically disabled from an operation. When an action is disabled, any event tied to the action will not cause the action to be executed. Further, menu items and several types of controls are automatically grayed when an action to which they are tied is disabled.

To disable an action, execute the operation `DisableAction`. This operation also allows you to enable a previously disabled action. For more details, see the Zeidon Programmer's Reference.

## Menus

Zeidon supports two kinds of menus: pull-down menus and pop-up menus. Logically the same, the difference is that pull-down menus appear along the menu bar at the top of a window, while pop-up menus appear only in response to a right mouse click in some field within the window. Whereas a window has only one menu bar (or none), there may be several different pop-up menus associated with it.

### See also:

[Creating Menus for a Window](#)

[Updating Window Menus](#)

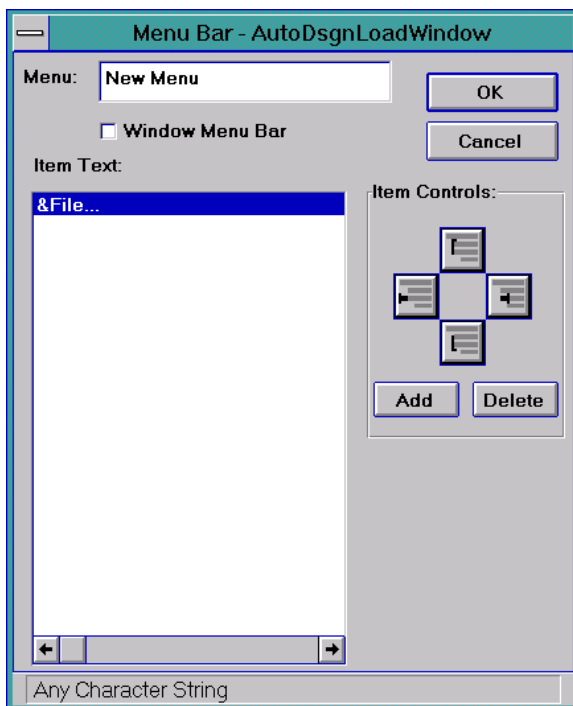
## Creating Menus for a Window

Window menus are created from the Menu List dialog box. There are a number of ways to open this dialog box:

- Select the window you wish to create a menu for by either clicking on the window in the Window List dialog box, or clicking on the window's Window Image. Select the **Menu List** item from the **Window** pull-down menu of the Dialog window.
- Right mouse click on the window you wish to create a menu for in the Window List dialog box. Select **Menu List** from the pop-up menu.
- Select the window you wish to create a menu for by either clicking on the window in the Window List dialog box, or clicking on the window's Window Image. Press the following Menu Icon from the Toolbar:



In any case, the Menu List dialog box is opened. In the Menu List dialog, press the **New** button to open the Menu Bar dialog box.



In the Menu Bar window, the menu name can be modified by typing in the text box provided. If the menu is to be the menu bar for the window, as opposed to a pop-up menu, click on the Window Menu Bar check box. Adding items to the menu is described in [Updating Window Menus](#).

## Updating Window Menus

Window menus are updated in the Menu Bar dialog box. The Menu Bar dialog is opened in the process of creating a new window menu as discussed in the previous section. For existing menus, there are a number of ways to present this dialog box:

- Select the window you wish to create an menu for by either clicking on the window in the Window List dialog box, or clicking on the window's Window Image. Select the **Menu List** item from the **Window** pull-down menu of the Dialog window. The Menu List dialog box is presented. Double-click the menu you wish to update from the list.
- Right mouse click on the window whose menu you wish to update in the Window List dialog box. Select **Menu List** from the pop-up menu. The Menu List dialog box is presented. Double-click the menu you wish to update from the list.
- Select the window you wish to create an menu for by either clicking on the window in the Window List dialog box, or clicking on the window's Window Image. Press the Menu Icon from the Toolbar. Double-click the menu you wish to update from the list.
- In the Window Image of the desired window, click any menu item on the menu bar.

In any event, the Menu Bar dialog box is presented.

### See also:

[Updating Menu Items](#)  
[Item Details](#)

## Updating Menu Items

In the Menu Bar dialog box, the following can be done:

**Adding an Item** - To Add an item to the menu list, press the **Add** button. The new item will be positioned one level lower than the currently highlighted item in the list box. If there are already items in that level the new item will be positioned last for that level.

**Moving an Item** - The item's position can be adjusted up and down, within the parent item, by pressing the icons in the Item Control control group to the right of the list box. The top most item will move the item up and the bottom will move it down. The left and right buttons will move the item to a higher and lower level in the menu, respectively.

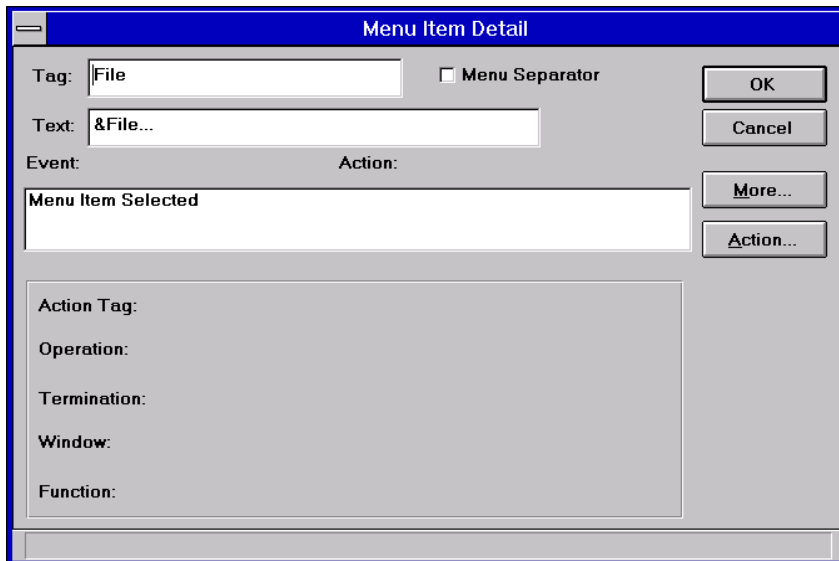
**Deleting an Item** - To delete a menu item, highlight the item to be deleted in the Item list and press the **Delete** button.

**NOTE:** sub-menu items of a deleted menu item will be deleted as well.

---

## Item Details

To define detail information of the menu items, double-click the desired item on the list box in the Menu Bar dialog box. The Menu Item Detail dialog box is presented.



The screenshot shows the 'Menu Item Detail' dialog box. It has a blue title bar. Inside, there's a 'Tag' text box containing 'File', a 'Menu Separator' checkbox, a 'Text' text box containing '&File...', an 'Event' text box containing 'Menu Item Selected', and an 'Action' text box containing 'Menu Item Selected'. To the right of these fields are four buttons: 'OK', 'Cancel', 'More...', and 'Action...'. Below these fields is a large text area with labels for 'Action Tag:', 'Operation:', 'Termination:', 'Window:', and 'Function:'.

The following may be specified for the menu item:

**Tag** - Enter a name for this menu item. Menu items are identified by this tag in procedural code when they are being grayed, checked on, etc.

**Text** - Enter the text as you wish it to appear. If you wish to have an accelerator for the menu item, key in an ampersand (&) before the desired accelerator character.

**Menu Separator** - Click this check box on if you want a separator line between this menu item and the one above it.

**Action** - To assign an action to be initiated when the menu item is selected, double-click on Menu Item Selected. If an action is already defined and you wish to update it, either double-click on Menu Item Selected or press the **Action** button. In either case, the Select Action dialog box is presented. Double-click on the desired action from the list or single-click to select the action and press the **OK** button. See [Creating Window Actions](#) for details on how to add actions to a window.

**NOTE:** if a menu item has an action assigned to it and if in the process of execution, that action is disabled by the application, the menu item will automatically be grayed. See [Disabling Window Actions](#) for more details.

---

Once the fields of the Menu Item Detail window have been updated, press **OK**.

## Shortcut Keys

A shortcut key is a keyboard key or combination of keys that is linked to a particular window action. Shortcut keys are useful for initiating actions without using the mouse. Providing shortcut keys allows experienced users a faster way of doing certain tasks.

### See also:

[Creating Window Shortcut Keys](#)

[Updating Window Shortcut Keys](#)

[Deleting Window Shortcut Keys](#)

## Creating Window Shortcut Keys

To create a shortcut key:

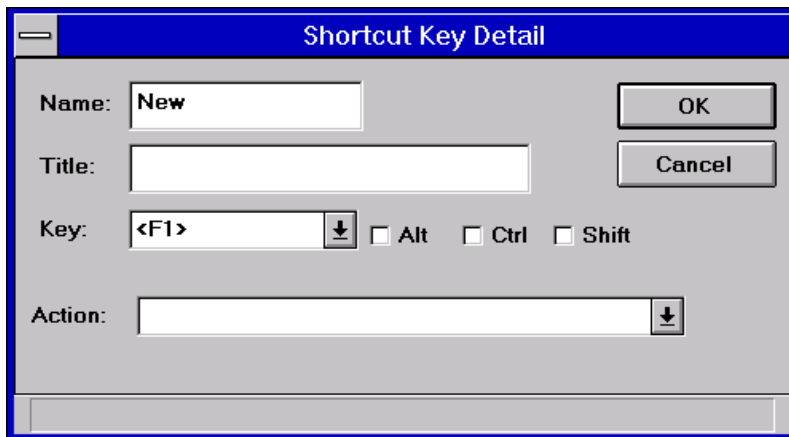
1. Select the window you wish to create a shortcut key for by either clicking on the window in the Window List dialog box, or clicking on the window's Window Image.
2. Select the **Shortcut List** item from the **Window** pull-down menu of the Dialog window. This brings up the Shortcut Key List dialog box.
3. Press the New Button. This presents the Shortcut Key Detail dialog box where the shortcut key information may be defined as described in [Updating Window Shortcut Keys](#).



## Updating Window Shortcut Keys

To update a shortcut key:

1. Select the window you wish to create a shortcut key for by either clicking on the window in the Window List dialog box, or clicking on the window's Window Image.
2. Select the **Shortcut List** item from the **Window** pull-down menu of the Dialog window. This brings up the Shortcut Key List dialog box.
3. Double-click the desired key from the list box. This presents the Shortcut Key Detail dialog box.



4. Update the following fields:

**Name** - Enter a name to identify the shortcut key.

**Title** - Enter the text for the key combination. This text will be displayed on any menu items which invoke the same action as this shortcut key.

**Key** - From the Key drop-down list, selects the keyboard character which will comprise the shortcut key.

**Alt** - Click on this check box if Alt is to be part of the shortcut key.

**Ctrl** - Click on this check box if Ctrl is to be part of the shortcut key.

**Shift** - Click on this check box if Shift is to be part of the shortcut key.

**Action** - From the Action drop-down combo box, select the action to be invoked when the shortcut key is used.

5. When the fields in the Shortcut Key Detail window have been updated, press **OK**.

## Deleting Window Shortcut Keys

To delete a shortcut key:

1. Select the window containing the shortcut key to be deleted by either clicking on the window in the Window List dialog box, or clicking on the window's Window Image.
2. Select the **Shortcut List** item from the **Window** pull-down menu of the Dialog window. This brings up the Shortcut Key List dialog box.
3. Highlight the desired key on the list box with a single-click and press the **Delete** button.

# Window Controls

Window controls are visible components of a window that display information or that allow for operator action. Zeidon window controls include, but are not restricted to, the following: Text, Edit boxes (single and Multi-Line), Buttons (Push, Radio), Check Boxes, List Boxes, Combo Boxes, Group Boxes, Spread Sheets, Dynamic Information Lines, Graphics, Bitmap Push Buttons, and Bitmap Controls.

## See also:

[The Toolbar and Control Palette](#)

[Setting up the Control Palette](#)

[Painting Window Controls](#)

[Selecting Controls](#)

[Moving Controls](#)

[Copying Controls](#)

[Updating Controls](#)

[Deleting Controls](#)

[Data Mapping for Controls](#)

[Aligning Controls](#)

[Spacing Controls](#)

[Sizing Controls](#)

[Assigning Tabbing Order](#)

[Disabling and Graying Controls](#)

[Common Details for Controls](#)

[Drag and Drop Between Controls](#)

[Controls within Group Boxes](#)



[Text Boxes](#)



[Edit Boxes](#)



[Multi-line Edit Boxes](#)



[Radio Buttons](#)



[Check Boxes](#)



[Push Buttons](#)



[Combo Boxes](#)



[List Boxes](#)



[Group Boxes](#)



[Dynamic Information Lines](#)



[Spread Sheets](#)



[Outliner Controls](#)



[Bitmap Push Buttons](#)



Bitmaps



Tabs

## The Toolbar and Control Palette

Controls are maintained primarily with the Dialog window toolbar and the Control Palette. The toolbar, a series of buttons displayed beneath the pull-down menus in the Dialog window, allow quick access to various pull-down menu items related to window controls. Such things include:



**Deleting Controls**



**Sizing Controls**



**Aligning Controls**



**Spacing Controls**

The Control Palette window contains a set of buttons that control the effect control painting within a window image. The choices include:



**Text**



**Edit**



**Multi-line Edit**



**Spread Sheet**



**Tab**



**Bitmap Push Button**



**Radio Buttons**



**Check Boxes**



**Combo Boxes**



**List Boxes**



**Group Boxes**



**Message Bar**



**Bitmap Controls**

Using the toolbar and Controls Palette to paint window controls are discussed in detail in the following sections. Further, what buttons (or icons) are displayed in the Controls Palette can be customized to your preference as is discussed in the following section.

## Setting up the Control Palette

The buttons (or icons) presented in the Control Palette may be specified according to each user's preference.

To specify which icons are presented in the Control Palette:

1. Select the **Control Palette** item from the **Options** pull-down menu of the Dialog window. This presents the Select Control Palette Icons dialog box.
2. Single-click on items from the Controls Not On Palette list box to select icons for the Control Palette. You can multi-select from this list box.
3. Press the arrow button (<-) pointing towards the Controls On Palette list box.

To de-select icons from the Painter Toolbar:

1. Single-click on items from the Controls On Palette list box to select icons you wish removed from the painter toolbar. You can multi-select from this list box.
2. Press the arrow button (->) pointing towards the Control Not On Palette list box.

## Painting Window Controls

All controls are painted and organized in a window image in a similar manner, regardless of type.

To paint a control in a window:

1. Bring up the window image, if not already visible, by double-clicking on the desired window from the list of the Window List dialog box.
2. From the Control Palette window, single-click on the icon for the control type to be painted. See the preceding section for details on displaying Control icons on the Control Toolbar.
3. Move the cursor, now a cross-hair, to the window image. With the cursor positioned at a corner of the control's desired location, left-mouse drag the cursor towards the opposite corner and release the mouse when the desired size of the control is reached.

You may continue to paint additional controls of the same type until the cursor is returned to its original shape. To reset the cursor, single-click anywhere on the window image or click the arrow cursor icon from the Control Palette.

## Selecting Controls

Various dialog tool procedures require you to select one or more control images in a window image. You select a single control by simply clicking on the desired control within the window image.

You can select multiple controls in a number of ways:

- While holding down the Ctrl key, single-click on the controls you want selected.
- Position the cursor in the window image and use a mouse drag to block an area around the controls you want selected, then release. The resulting rectangle must completely enclose the control or it will not be selected. Use a right or left mouse drag when the initial position of the rectangle is in an open area of the window image. Use a right mouse drag if the initial position of the cursor is inside a control. This is referred to as a rubber band select.

You can use a combination of the rubber band select with the Ctrl key. That is, you can select one or more controls with either method, then rubber band select while holding down the Ctrl key to select additional controls.



## Moving Controls

To move controls painted on the window image:

1. Obtain focus on the control to be moved by clicking anywhere within the control image.
2. Mouse drag the image to the desired location of the window image and release or, while holding down the shift key, use the keyboard arrow keys to reposition the control in small increments.

If multiple controls have been selected they can be moved at the same time. See [Selecting Controls](#) for details on selecting more than one control at a time.

## Copying Controls

To copy controls on the window image:

1. Obtain focus on the control to be moved by clicking anywhere within the control image.
2. While holding down the Ctrl key, mouse drag the control to the desired location and release.

If multiple controls have been selected they can be copied at the same time. See [Selecting Controls](#) for details on selecting more than one control at a time.


## Updating Controls

To update a control double-click on the control image. Though much of the information on controls is common across control types, the dialog box presented will vary according to the control type selected for update. Updating each control type is discussed in its own topic.

## Deleting Controls

To delete controls from the window image:

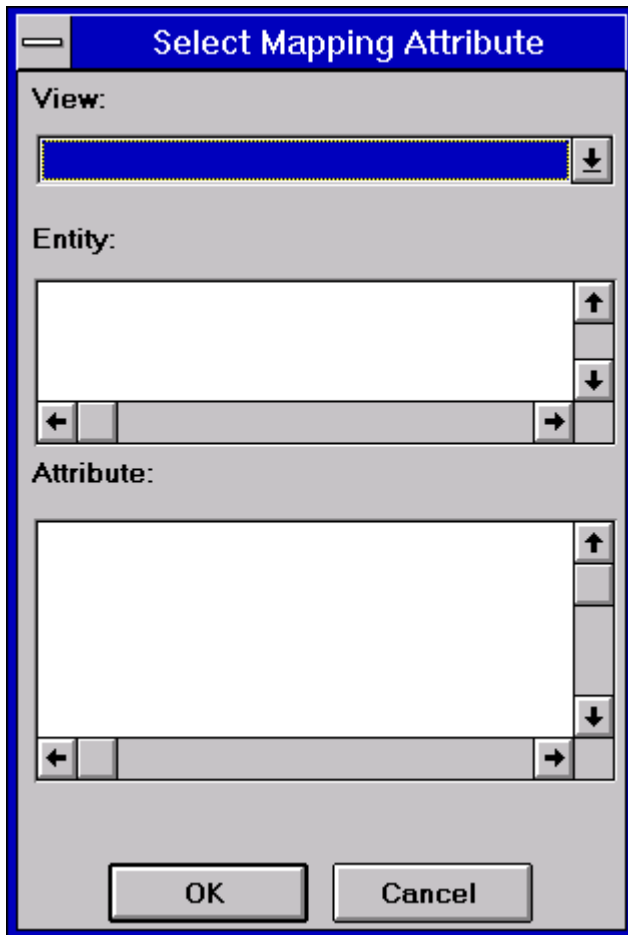
1. Obtain focus on the control to be deleted by clicking anywhere within the control image.

2. Click on the Delete icon, , or press the **Delete** key.

It is possible to delete more than one control at a time by selecting multiple controls. See [Selecting Controls](#) for details on multiple selection of controls.

## Data Mapping for Controls

While updating controls, it is often necessary to specify data mapping criteria. The selection of data mapping occurs in the Select Mapping Attribute dialog box. How this dialog box is brought up can vary from control to control but typically, double-clicking on a control will open a dialog box that contains either a **Mapping** or **Data** button. Pressing these buttons then presents the Select Mapping Attribute.



To select the mapping attribute:

1. From the View drop-down list, select the view to the object instance which contains the desired entity and attribute.
2. Click on the entity containing the desired attribute in the Entity list box.
3. Select the desired Attribute from the Attribute list box by either double-clicking on it or by single-clicking on it and then pressing **OK**.

## Aligning Controls

Controls can be aligned according to their left, right, top or bottom margins. To align controls:

1. Select the controls to be aligned by clicking anywhere within the controls while holding down the keyboard Control key.

**NOTE:** the control which is selected last is the control to which the others will be aligned.

---

2. Press the desired alignment icon from the control toolbar. The choices are:



- Align Bottoms



- Align Tops



- Align Left Sides



- Align Right Sides

## Spacing Controls

Controls can be spaced evenly in either the horizontal or vertical direction. To evenly space controls:

1. Select the controls to be spaced by clicking anywhere within the controls while holding down the keyboard Control key.
2. Press the desired spacing icon from the control toolbar. The choices are:



- Space Horizontally



- Space Vertically

## Sizing Controls

Controls can be sized individually, or adjusted to the size of another control.

To size a control individually:

1. Obtain focus on the control to be sized by clicking anywhere within the control image.
2. Position the cursor on the sizable frame of the control and mouse drag to the desired size. Or, while holding down the Ctrl key, use the keyboard arrow keys to resize the control in small increments.

To adjust the size of a control to match the size of another control:

1. Select the controls to be sized by clicking anywhere within the controls while holding down the keyboard Control key.

**NOTE:** the control which is selected last is the control to which the others will be sized.

---

2. Press the desired sizing icon from the control toolbar. Icons are available to adjust controls to the same width, length, or both. The choices are:



- Size Heights  
Width



- Size Widths



- Size Height and  
Width



## Assigning Tabbing Order

Within a window, pressing Tab will change the focus from one control to another within the current window. Repeated Tabs will cycle through some or all of the window's controls. The order in which focus is passed from control to control is called the *tabbing order* for the window.

Tabbing order is defined for a control by one of the following:

- Left mouse drag a rectangle around the control and press the right mouse button while holding down the Shift key.
- Right mouse drag a rectangle around the control while holding down the Shift key and then release.

**NOTE:** the second method is necessary if the tabbing is being assigned to a control within a group box, because left mouse dragging moves the group box.

---

The sequence in which the tabs are assigned defines the tabbing order. That is, the first assigned is the first in tabbing order and so on. Also, if a control has had a tab assigned, and then you re-assign a tab to that control, the previously assigned tab order is lost.

You can assign tabbing to multiple controls by dragging the rectangle over more than one control at once. Done this way, the tabbing order begins with the upper left and increases going towards the lower right.

**NOTE:** tabbing from a control within a group box, to a control outside the group box, and then back to a control within the group box is not supported.

---

## Disabling and Graying Controls

Controls can be disabled dynamically during application execution. When a control is disabled, all events for the control are also disabled. When disabled, a control's appearance is altered. Generally, the control appears to be "grayed out".

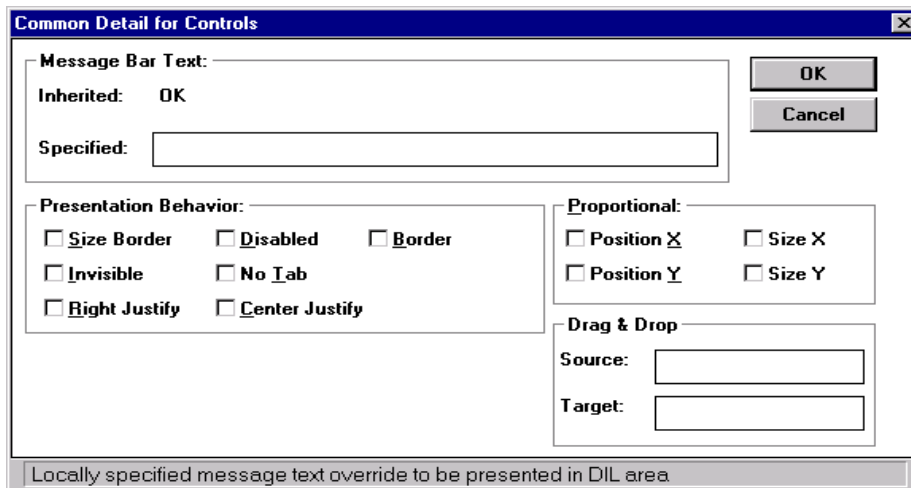
Controls can be disabled in one of two ways:

- **Disable a Control** - To disable a specific control, use the Zeidon operation SetControlState.  
NOTE: if you disable a control using this operation, the only way you can enable it is to call the operation again. See the Zeidon Programmer's Reference for more information.
- **Disable an Action** - If an action is disabled, some types of controls are automatically disabled provided they are tied to that action.

## Common Details for Controls

All controls have certain characteristics in common. These include colors, proportional sizing specification, and presentation behavior. For each control, this common information is specified on the Common Detail For Controls dialog box. This dialog box is usually accessed from the control type dialog box by pressing the **Common** push button.

To update the common details for a specific control, open the control type dialog for the control by double-clicking on the control in the window image.



The dialog box titled "Common Detail for Controls" has a blue title bar with a close button. It contains several sections: "Message Bar Text" with an "Inherited" field showing "OK" and a "Specified" text box; "Presentation Behavior" with checkboxes for "Size Border", "Disabled", "Border", "Invisible", "No Tab", "Right Justify", and "Center Justify"; "Proportional" with checkboxes for "Position X", "Size X", "Position Y", and "Size Y"; and "Drag & Drop" with "Source" and "Target" text boxes. "OK" and "Cancel" buttons are in the top right. A status bar at the bottom reads "Locally specified message text override to be presented in DIL area".

Maintain the common control detail information as follows:

**Message Bar Text** - Enter text in the Specified edit box which you wish to appear in the default Message Bar Item when this control has focus. If left blank, the inherited string is displayed. If an attribute mapping has been specified for this control, then the inherited string is taken from the context of the domain for the mapping attribute. That is, it is taken the Message Bar definition for that context. If no attribute mapping has been specified for the control, the control tag is used.

**Presentation Behavior** - Select the presentation style for the control from the following:

- **Size Border** - The control will have a sizable border.
- **Disabled** - The control will be disabled.
- **Border** - The control will have a standard (non-sizable) border.
- **Invisible** - The control will not be displayed.
- **No Tab** - The control will not be assigned tabbing.
- **Right Justify** - The text within the control is right justified.

**Proportional** - Specify how you wish the control to respond to sizing of the window or dialog box it is within. Select from the following:

- **Size X** - Increase/decrease the size of the control along the X axis in proportion to the increase/decrease in size of its parent window or dialog

box along the X axis. That is, if the parent window is doubled in length, double the length of this control.

- **Size Y** - Increase/decrease the size of the control along the Y axis in proportion to the increase/decrease in size of its parent window or dialog box along the Y axis. That is, if the parent window is doubled in height, double the height of this control.
- **Position X** - Reposition the control along the X axis in proportion to the changes in size of its parent window or dialog box. For example, if the control was originally centered along the X axis, keep it centered, if it was originally positioned 3/4's the length of the parent, reposition it to 3/4's the length of the parent.
- **Position Y** - Reposition the control along the Y axis in proportion to the changes in size of its parent window or dialog box. For example, if the control was originally centered along the Y axis, keep it centered, if it was originally positioned 3/4's the length of the parent, reposition it to 3/4's the length of the parent.

**Drag & Drop** - The Source and Target edit boxes define drag and drop functionality involving the control. For more information on defining drag and drops, see [Drag and Drop Between Controls](#).

When the fields have been updated, press **OK**.

## Drag and Drop Between Controls

Drag and drops between controls are defined in the Common Detail for Controls windows for the source and target controls for the drag and drop. For more information on opening the Common Detail for Controls window, see the preceding section.

A source control for a drag and drop is defined by specifying a name for the drag and drop in the Source edit box in the Common Detail for Control window for the source control.

Only those controls that have a Left Mouse DropTarget or Left Mouse DropSource event defined for them can possibly be targets for a drag and drop. To define such a control as a target, the Left Mouse DropSource or DropTarget event must have an assigned action. The DropSource event notifies the source control that the drop has occurred. Similarly, the DropTarget event notified the target control that the drop has occurred. A common use of Drag & Drop is to move (or copy) items from one listbox to another. An alternative implementation for this is to use buttons with left and right arrows. To implement Drag & Drop with the same action as used by the buttons, the DropSource event should be used rather than the DropTarget event. In particular, the Auto Gray functionality is equivalent for the arrows implementation and the DropSource implementation. The Drop Target event may be recommended for situations in which the source requires specialized processing depending on the target of the Drop. For more information on window actions see Window Actions.

To tie a source control together with a target (i.e. to allow a target control to accept a drop from one or more source controls) you can use one of two methods:

- You can specify an action operation for the Left Mouse QueryDrop event for the target control. This operation must query some internal information set up by the Left Mouse BeginDrop event for a source control. If the control is to accept a drop, the operation must return a TRUE (value of 1). Otherwise, the operation must return a FALSE (value of 0).
- You can identify the source control by the name specified in the Drag & Drop Source edit box in the Common Detail for Controls window for the source control. Enter this same name in the Drag & Drop Target edit box in the Common Detail for Controls window for the target control.

Both methods will allow a control to be the source for drag and drops to multiple target controls. Also, the first technique will allow a control to be the target of drag and drops from multiple source controls.

**NOTE:** the second technique will not allow a control to be the target for more than one source control.

---

## Controls within Group Boxes

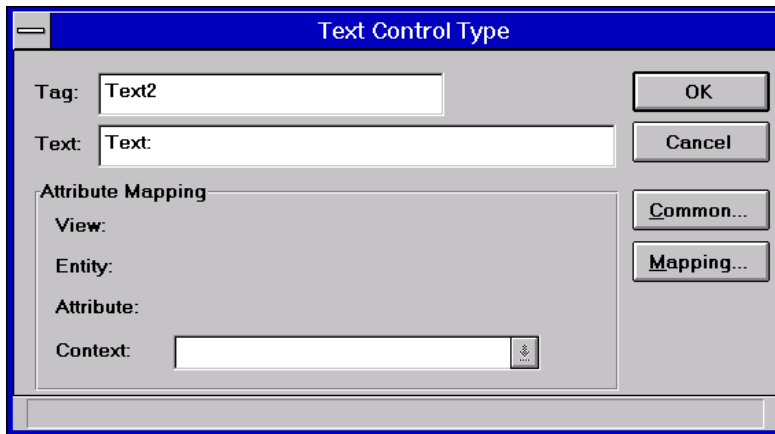
Once a group box has been painted within a window image, other controls, such as edit boxes, text boxes and other controls may be painted inside the group box. If the group box is then moved, the controls painted within it are moved along with it. Likewise, if the group box is deleted, all the controls within it are deleted as well. However, if controls exist on the window image and a group box is painted over those controls, the controls are not considered associated with that group box, and subsequent manipulation of the group box, such as moving or deleting it, do not affect the controls it was painted around.

If it is desired to have existing controls associated with a new group box, you can do so by moving the existing controls into the group box by using a drag and drop with the mouse.

## Text Boxes

Text controls are controls on a window which display text information. The text display is not modifiable and you cannot get focus on the control.

To update a text control, double-click on text control in the window image to present the Text Control Type dialog box.

The image shows a dialog box titled "Text Control Type". It has a blue title bar. Inside, there are several input fields and buttons. The "Tag:" field contains "Text2". The "Text:" field contains "Text:". Below these is a section titled "Attribute Mapping" which contains fields for "View:", "Entity:", "Attribute:", and "Context:". The "Context:" field is a text box with a small drop-down arrow on its right. To the right of the input fields are four buttons: "OK", "Cancel", "Common...", and "Mapping...".

The text control is maintained as followed:

**Tag** - Enter a name to uniquely identify this text control within this window. The control can then be referenced in procedural code, (e.g., for graying).

**Text** - Enter the text you wish to appear in this control.

**Mapping** - You can specify an entity attribute whose value is to be displayed in the control. Press the **Mapping** button to specify the object instance mapping. Refer to [Data Mapping for Controls](#) for details on control data mapping.

**NOTE:** Mapping and text are mutually exclusive. That is, you can either specify text to be displayed or you specify an attribute mapping but you can not do both.

---

**Context** - If attribute mapping is specified, select the domain context for the attribute from the combo box drop-down. If left blank, the default context is used.

**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

When the fields have been updated, press **OK**.

## Edit Boxes

Edit controls are controls which allow the user to key in a single line of text.

To update an edit control, double-click on the edit control in the window image to present the Edit Control Type dialog box.

The edit control is maintained as followed:

**Tag** - Enter a name to uniquely identify this edit control within this window. The control can then be referenced in procedural code, (e.g., for graying).

**Type** - Select the editing characteristics you want for the edit box. The choices are:

**Upper/Lower Case** - Editing accepts upper and lower case.

**Lower** - Editing accepts alphanumeric, but converts upper case alpha characters to lower case.

**Upper** - Editing accepts alphanumeric, but converts lower case alpha characters to upper case.

**Password** - Keystrokes are displayed in the edit control as asterisks.

**Mapping** - Press the **Mapping** button to specify object instance mapping for this control. Refer to [Data Mapping for Controls](#) for more details.

**Context** - If attribute mapping is specified, select the domain context for the attribute from the combo box drop-down. If left blank the default context is used.


**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

When the fields have been updated, press **OK**.



## Multi-line Edit Boxes

Multi-line Edit controls are controls which allow the user to key in multiple lines of text. They support tabbing, page breaks, carriage return, and wrap around editing. In most other ways they are similar to Edit controls. Double-clicking on a multi-line edit control in the window image presents the Edit Control Type dialog box

since the specifications for a multi-line edit control are the same as an edit control. Refer to  [Edit Boxes](#) for the specification details.

## Radio Buttons

Radio buttons provide a single choice from a limited set of mutually exclusive choices. From a group of radio buttons, only one can be selected at a time. Since radio buttons are mutually exclusive within a group, radio buttons can only be painted within a radio group box. The same toolbar icon is used to paint radio group boxes and radio buttons. The tool paints a radio group box first. A radio button is painted only if the area clicked and mouse dragged is within a radio group.

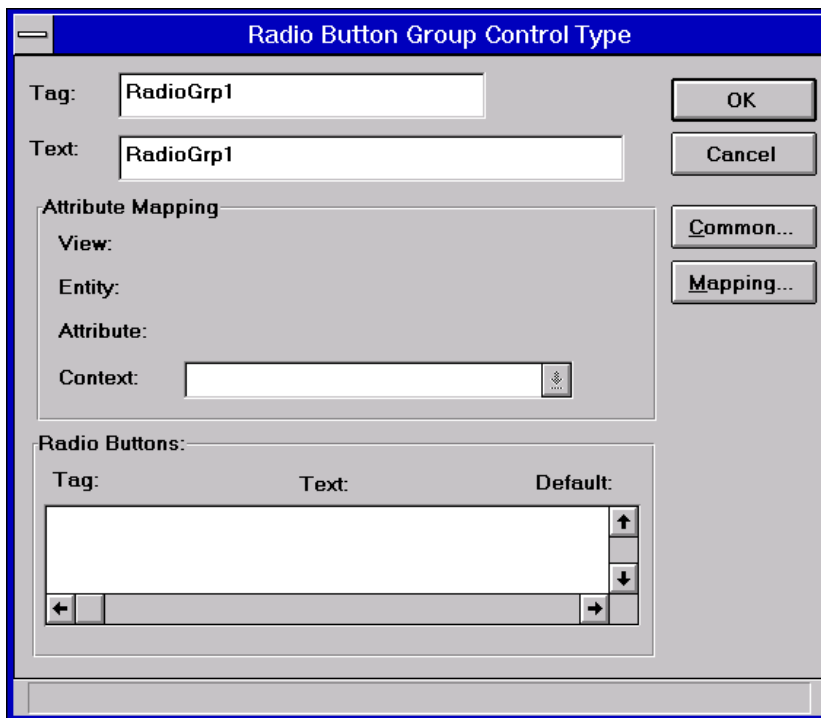
### See also:

[Updating Radio Button Groups](#)

[Updating Radio Buttons](#)

## Updating Radio Button Groups

To update a radio button group, double-click on the radio group box with focus. The Radio Button Group Control Type dialog box is presented.



The dialog box is titled "Radio Button Group Control Type". It contains several sections and controls:

- Tag:** A text field containing "RadioGrp1".
- Text:** A text field containing "RadioGrp1".
- Attribute Mapping:** A section with labels for "View:", "Entity:", "Attribute:", and "Context:". The "Context:" label is followed by a text field and a small icon.
- Radio Buttons:** A section with a table-like structure. The header row has "Tag:", "Text:", and "Default:". Below the header is a large empty text area. To the right of this area are four small buttons: an up arrow, a down arrow, a left arrow, and a right arrow.
- Buttons:** On the right side of the dialog, there are four buttons: "OK", "Cancel", "Common...", and "Mapping...".

The fields of Radio Button Group control are maintained as followed:

**Tag** - Enter a name to uniquely identify this control within this window. The control can then be referenced in procedural code (e.g., for specifying defaults).

**Text** - Enter the text you want displayed to identify the group box.

**Mapping** - If desired, press the **Mapping** button to specify object instance mapping for this control. The attribute mapping specified will apply to all radio buttons within the group. Refer to [Data Mapping for Controls](#) for more details.

**Context** - If an attribute mapping is specified, select the domain context for the attribute from the combo box drop-down. If left blank the default context is used.

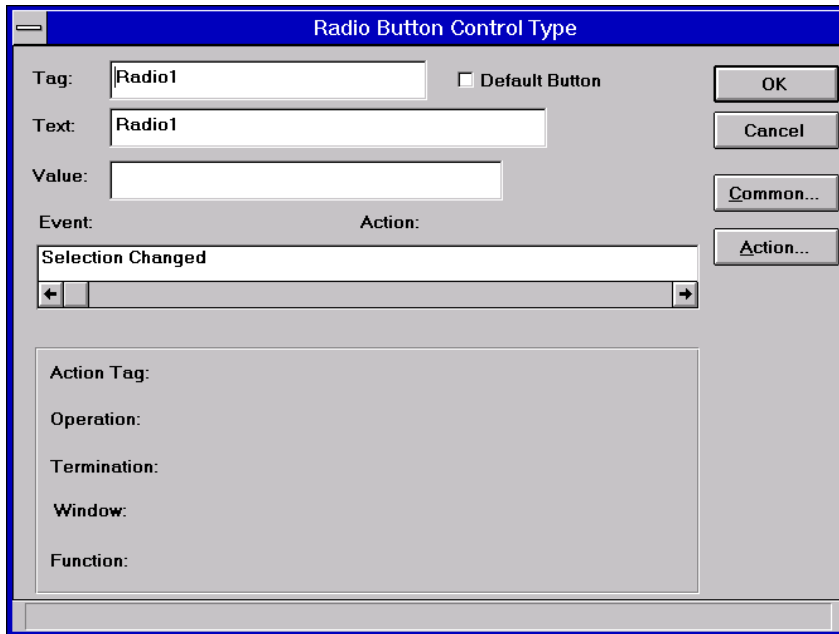
**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

When the fields are updated, press **OK**.

## Updating Radio Buttons

The Radio Button Group Control Type dialog box (see above) lists the radio buttons of the group.

To update a radio button in the group, double-click on the button in the list box of the Radio Button Group Control Type dialog box. This will open the Radio Button Control Type dialog box. Alternatively, double-clicking on a radio button with focus in the window image, will open the same Radio Button Control Type dialog box.

The image shows a dialog box titled "Radio Button Control Type". It has a blue title bar. Inside, there are several fields and buttons. At the top, there's a "Tag:" field with "Radio1" entered, and a "Default Button" checkbox which is unchecked. Below that is a "Text:" field also containing "Radio1". Underneath is a "Value:" field which is empty. To the right of these fields are four buttons: "OK", "Cancel", "Common...", and "Action...". Below the "Value:" field, there are two labels, "Event:" and "Action:", followed by a list box containing "Selection Changed". Below the list box are left and right arrow buttons. At the bottom of the dialog is a large rectangular area with labels for "Action Tag:", "Operation:", "Termination:", "Window:", and "Function:", each followed by a text input field.

The Radio Button control is maintained as followed:

**Tag** - Enter a name to uniquely identify this text control within this window. The control can then be referenced in procedural code, (e.g., for graying).

**Text** - Enter the text you wish to appear for the button.

**Default Button** - Click on this check box if this radio button is to be the default for the group.

**Value** - If mapping was specified for the radio group, key in the value to be mapped to the mapping attribute if this radio button is selected.

**Event Action** - Double-click the Selection Changed event line if an action is to be invoked for this event. The Select Action dialog box is presented. Double-click on the desired action from the list or single-click on the action to highlight it and press the **OK** button. If you need to specify a new action, refer to [Creating Window Actions](#) for details on how to add actions to a window. Press the **Remove** button to remove the action from this control event.

**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

**Action** - Press this button if you wish to update the specified Action for the radio button. This will present the Action Detail dialog box. Refer to [Creating Window Actions](#) for details on updating actions.

**NOTE:** if a radio button has an action assigned to it and if in the process of execution, that action is disabled by the application, the button will automatically be grayed. See [Disabling Window Actions](#) for more details.

---

Once the fields have been updated, press **OK**.

## Check Boxes

Check Boxes indicate individual choices which are either selected (turned on) or not selected (turned off).

To update a check box control, double-click on the check box control in the window image. This opens the Check Box Control Type dialog box.

The screenshot shows the 'Check Box Control Type' dialog box. It has a blue title bar with the text 'Check Box Control Type'. The main area is gray and contains several sections. At the top, there are two text boxes: 'Tag:' with the value 'Check1' and 'Text:' with the value 'Check1'. To the right of these are 'OK' and 'Cancel' buttons. Below the 'Text' box is the 'Attribute Mapping' section, which includes labels for 'View:', 'Entity:', and 'Attribute:', and a 'Context:' text box with a drop-down arrow. To the right of this section are buttons for 'Common...', 'Mapping...', and 'Action...'. Below the 'Attribute Mapping' section is the 'Attribute Value' section, which includes labels for 'Checked:' and 'Unchecked:', and text boxes containing '1' and an empty box respectively. At the bottom is the 'Event' section, which has a table with two columns: 'Event' and 'Action'. The 'Event' column contains 'SelectionChanged' and the 'Action' column is empty. There are left and right arrow buttons at the bottom of the table.

The check box control is maintained as followed:

**Tag** - Enter a name to uniquely identify this control within this window. The control can then be referenced in procedural code (e.g., for graying).

**Text** - Enter the text you wish to appear in this control.

**Mapping** - Press the **Mapping** button to specify object instance mapping for this control. Refer to [Data Mapping for Controls](#) for details.

**Context** - If attribute mapping is specified, select the domain context for the attribute from the drop-down combo box. If left blank the default context is used.

**Attribute Value** - Enter the value which will be mapped to the specified mapping attribute when the check box is checked and unchecked.

**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

**Event Action** - Double-click the SelectionChanged event line to select an action to be invoked for this event. The Select Action dialog box is presented. Double-click on the desired action from the list or single-click on the action to highlight it and press the **OK** button. If you need to specify a new action, refer to [Creating Window Actions](#) for details on how to add actions to a window. Press the **Remove** button to remove the action from this control event.

**Action** - Press this button if you wish to update the specified action for the Selection Changed Event. This will present the Action Detail dialog box. Refer to [Creating Window Actions](#) for details on updating actions.

**NOTE:** if a check box has an action assigned to it and if in the process of execution, that action is disabled by the application, the check box will automatically be grayed. See [Disabling Window Actions](#) for more details.

---

When the fields in the dialog box have been updated, press **OK**.

## Push Buttons

To update a push button control, double-click on the push button control in the window image. This will open the Push Button Control Type.

The screenshot shows a dialog box titled "Push Button Control Type". It has a standard Windows-style title bar. Inside the dialog, there are several input fields and buttons. The "Tag" field contains "Push1". The "Text" field also contains "Push1". The "Event" field contains "ButtonPress". The "Action" field is empty. To the right of these fields are four buttons: "OK", "Cancel", "Common...", and "Action...". Below these fields is a large text area with labels: "Action Tag:", "Operation:", "Termination:", "Window:", and "Function:". The text area is currently empty.

A push button control is maintained as followed:

**Tag** - Enter a name to uniquely identify this control within this window. The control can then be referenced in procedural code, (e.g., for graying).

**Text** - Enter the text you wish to appear in this control.

**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

**Event Action** - Double-click the ButtonPress event line to select an action to be invoked for this event. The Select Action dialog box is presented. Double-click on the desired action from the list or single-click on the action to highlight it and press the **OK** button. If you need to specify a new action, refer to [Creating Window Actions](#) for details on how to add actions to a window. Press the **Remove** button to remove the action from this control event.

**Action** - Press this button if you wish to update the specified Action for the ButtonPress Event. This will present the Action Detail dialog box. Refer to [Creating Window Actions](#) for details on updating actions.

**NOTE:** if a push button has an action assigned to it and if in the process of execution, that action is disabled



by the application, the button will automatically be grayed. See [Disabling Window Actions](#) for more details.

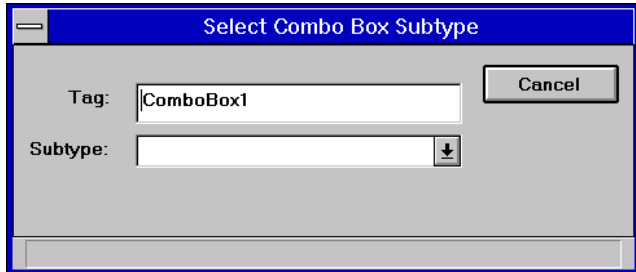
---

When the fields of the dialog window have been updated, press **OK**.

## Combo Boxes

A combo box is a edit or static control with an attached interdependent list displaying possible choices.

To define and update combo boxes, double-click on the combo box in the window image. If the type of combo box has not yet been specified, a Select Combo Box Subtype dialog box appears.



Initialize the combo box by specifying the following:

**Tag** - Enter a name to identify the combo box.

**Subtype** - Select the subtype from the drop-down combo box. The choices are:

**Domain** - The listed items are the context table items of a domain. The attribute is set from the selected domain item.

**Automatic Include** - The combo box selection triggers an include from a specified source entity to a specified target entity.

**Set Foreign Key** - The combo box selection triggers the setting of a foreign key attribute in a specified target entity from a source entity.

**Select Source Entity Only** - The selection only marks the list entity as selected. Procedural code may then be invoked to process data based upon the entity being marked as selected.

**Select Foreign Key and Select Source** - The selection triggers the setting of the foreign key and marks the list entity as selected.

Selecting a subtype presents one of two dialog boxes, depending upon the subtype selection. Follow the instructions in one of the following sections.

### See also:

[Domain Type Combo Boxes](#)

[Other Combo Boxes](#)

## Domain Type Combo Boxes

If Domain is chosen in the Select Combo Box Subtype dialog box, then a Combo Box Control Type - Domain dialog box is presented.

To complete the update or create of the combo box, update the following fields of the Combo Box Control Type - Domain dialog box:

**Tag** - Enter a name to uniquely identify this control within this window.

**Subtype** - Select the combo box subtype from the drop-down list. Subtype was first specified when first updating the combo box. Selecting a subtype other than domain presents a Combo Box Control Type - Select dialog box.

**List Style** - Select the combo box style from the one of the following:

- **Drop-down List** - Contains a static control and list box (presented when the drop-down arrow is pressed). Selections are made by clicking from the list only. Entered character keys position and map the list box entry to the static control on matches.
- **Drop-down** - Contains an edit control and list box (presented when the drop-down arrow is pressed). Selections are made from the list or keyed in the edit box. Entered character keys also position the list box if matches occur.
- **Simple** - Contains an edit control and associated list box. Selections are made from the list or keyed in the edit box. Entered character keys also position the list box if matches occur.

**Sort Presented List** - Click on this check box if the presented list is to be sorted alphabetically.

**Mapping** - Press this button to present the Select Mapping Attribute dialog box. (Refer to [Data Mapping for Controls](#) for details on attribute selection). The value of the attribute selected will be displayed in the edit or static control. The context table items for that attribute's domain will be presented in the list box.

**Context** - Select the context of the attribute's domain to use. If no context is specified the default context will be used.

**Event Action** - Double-click on an event to specify or update an action to be invoked for that event. The Select Action dialog box is presented. Double-click on the desired action from the list or single-click to select the action and press the **OK** button. If you need to specify a new action, refer to [Creating Window Actions](#) for details on how to add actions to a window. Press the **Remove** button to remove the action from this control event. Events for a combo box are:

- **Select End** - occurs when the drop-down list has been closed (valid for Drop-down types only).
- **Select Change** - occurs whenever a selection is made or entered which is different from the displayed value.
- **DblClick** - occurs whenever you double-click on a simple combo box (valid for Simple types only).

**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

When the fields have been updated, press **OK**.

## Other Combo Boxes

If Automatic Include, Set Foreign Key, Select Source Entity Only, or Set Foreign Key and Select Source is the chosen subtype in the Select Combo Box Subtype dialog box, then a Combo Box Control Type - Select dialog box is presented.

The screenshot shows the 'Combo Box Control Type - Select' dialog box. It contains the following fields and controls:

- Tag:** ComboBox1
- Type:** Automatic Include (dropdown)
- Style:** Drop Down List (dropdown)
- ☐ Sort Presented List
- ☐ No Null Entry In List
- List Box Entity Mapping:**
  - Entity:** (dropdown)
  - Scope:** (dropdown)
  - ☐ Scope Object Instance
- List Box Mapping:**
  - View:** (empty)
  - Entity:** (empty)
  - Attribute:** (empty)
- Event:** SelectChange, SelectEnd, CloseUp (Obsolete), OnClick (Simple)
- Action:** (empty)
- Buttons:** OK, Cancel, Common..., Map Edit..., Map List...

To complete the update or create of the combo box, update the following fields in the Combo Box Control Type - Select dialog box, as described above for domain subtypes: Tag, Subtype, Style, Sort Presented List, Event Action, and Common.

Defining attribute mappings is different than the previous case. The **Map List** button presents a Select Mapping Attribute dialog box. The attribute selected identifies what is to be displayed in the list box control of the combo box. For an Automatic Include, this also identifies the entity which is to be the source of the include. In the case of a Set Foreign Key, or Set Foreign Key and Select Source subtype, this identifies the attribute which is used to set the foreign key. For a Select Source Entity Only subtype, this identifies which entity will be marked as selected. Specify the list entity and the scoping entity for the list in the List Box Entity Mapping group box back in the Combo Box Control Type - Select dialog. Select the entity from the Entity drop-down list and the scoping entity from the Scope drop-down list.

The **Map Edit** button presents a Select Mapping Attribute dialog box. The attribute selected identifies what is displayed in the edit or static control portion of the combo box. For an Automatic Include, this identifies the entity which is to be the target of the include. In the case of a Set Foreign Key, or Set Foreign Key and Select Source subtype, this identifies the attribute which is to be set to the foreign key.

When the fields have been updated, press **OK**.

## List Boxes

List boxes are used to display choices to users. They may be classified as to whether they allow the selection of a single item or multiple items.

In Zeidon, a list box group is painted first and then the line items of the list are painted next. The same toolbar icon is used to paint both. Paint a list box group first and size it appropriately. Next, paint the list items by clicking and dragging inside the list box group area.

**NOTE:** it is probably desirable to paint the list items within the list box starting in the upper left corner and working right.

---

### See also:

[Updating List Box Groups](#)

[Updating List Box Line Items](#)

[Drag and Drop on the Same Listbox](#)

## Updating List Box Groups

To update a list box group, double-click on the list box group with focus. This will open the List Box Control Type dialog box.

The screenshot shows the 'List Box Control Type' dialog box. It has a title bar with the text 'List Box Control Type'. The dialog is divided into several sections. At the top, there is a 'Tag' field with the text 'ListBox1'. Below this is a 'List Style' dropdown menu currently set to 'Single', and a checkbox labeled 'Use Integral Height'. To the right of these are three buttons: 'OK', 'Cancel', and 'Common...'. Below the 'List Style' section is a 'Mapping Entity' section containing two dropdown menus, 'View' and 'Entity'. Below that is a 'Scope' section with three radio buttons: 'Parent Entity' (which is selected), 'Specific Entity', and 'Object Instance'. To the right of the radio buttons is a 'Specific Entity' dropdown menu and a checkbox labeled 'Present Null List if Scoping Entity is Not Selected'. At the bottom is an 'Event' section with a list box containing four items: 'Pick (Single Click)', 'Enter (Double Click)', 'PickNew (Single Click New)', and 'EnterNew (Double Click New)'. There are navigation arrows on the right side of the list box.

The following are maintainable fields for a list box control:

**Tag** - Enter a name to uniquely identify the list box within this window.

**List Style** - From the drop-down list, select the style type of the list box. The choices are:

- **Single** - Only one item on the list may be selected at a time.
- **Multiple** - Allows for the multiple selection of items. Clicking acts as a toggle, selecting items that are unselected and unselecting selected items.
- **Extended** - Allows for the multiple selection of items. Clicking sets selection on and marks it as an anchor item. Shift + click then selects all items between itself (including itself) and the anchored item. Ctrl + click toggles the select state of items in the list individually.
- **Single / Hilite Cursor Pos** - Only one item selected at a time. The entity with cursor position is highlighted when the list is presented.

**Use Integral Height** - Allow the operating system to adjust the height of the list box based upon the best fit of list box items. That is, the list box height will be set so that no item in the list box is only partially displayed.

**View** - In the Mapping Entity group box, select the view containing the entity listed from the View drop-down list.

**Entity** - Select the entity listed from the Entity drop-down list.

**Scope** - In the Scope group box, select the radio button which describes the scoping that is to occur for the list. The choices are:

- **Parent Entity** - Select this radio button if scoping is within the parent only.
- **Specific Entity** - Select this radio button if scoping occurs at a higher level than the parent entity.
- **Object Instance** - Select this radio button if scoping is to occur across multiple roots of the Object Instance.

**Specific Entity** - Select the scoping entity from the drop-down list if Specific Entity was selected for scoping.

**Present Null List if Scoping Entity is Not Selected** - Check this if you want the list box presented with a null list when the specified scoping entity specified has not been selected.

**Event Action** - The events for a list box are listed in this list box. Double-click on an event to specify or update an action to be invoked for that event. The Select Action dialog box is presented. Double-click on the desired action from the list or single-click to select the action and press the **OK** button. If you need to specify a new action, refer to [Creating Window Actions](#) for details on how to add actions to a window. Press the **Remove** button to remove the action from this control event.

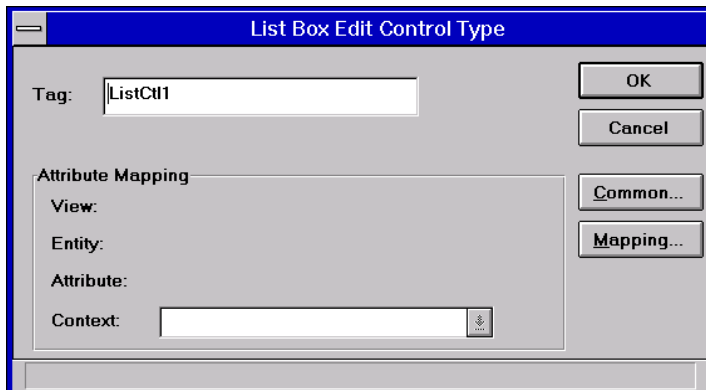
**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

When the fields of the List Box Control Type window have been completed, press **OK**.



## Updating List Box Line Items

To update a line item in a list box group, double-click on a list box line item with focus in the window image. The List Box Edit Control Type dialog box is presented.



Specify the following information for the list box edit controls:

**Tag** - This identifies the list box edit item.

**Mapping** - Press this button to present the Select Mapping Attribute dialog box. The value of the attribute selected will be displayed in the list box edit control. Refer to [Data Mapping for Controls](#) for details on attribute selection.

**Context** - Select the context of the attribute's domain to use. If no context is specified the default context will be used.

**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

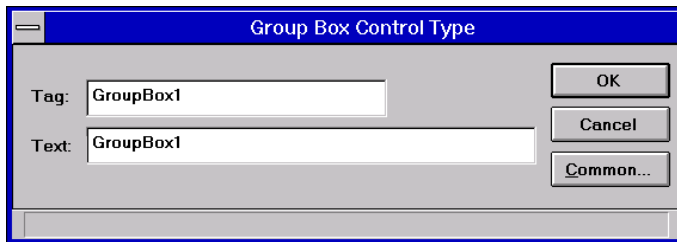
When the fields of the List Box Edit Control Type window have been completed, press **OK**.

## **Drag and Drop on the Same Listbox**

If you want to reorder entities under an immediate parent by drag and dropping on the same list box, you can do so automatically using the Drag and Drop functionality of the Common Control window. From the list box detail, select the Common button to take you to the Common Detail for Controls window. Under the Drag & Drop section, enter the same name (any name will do as long as it is not used in any other Drag and Drop definition) for both the Source and the Target values.

## Group Boxes

Group boxes, though considered controls, simply provide visual grouping for related controls. A group box is a rectangular frame with a label in the upper left corner. To update group boxes, double-click on the group box in the window image. This brings up the Group Box Control Type dialog Box.



Data which may be specified for a group box control include:

**Tag** - Enter a name to uniquely identify the group box within this window. box.

**Text** - Enter the label that is to appear in the upper left corner of the rectangular frame.

**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

When the fields have been completed, press **OK**.

## Dynamic Information Lines

The Dynamic Information Line (DIL), also referred to as the Message Bar Line, is an optional component of a window. The DIL is used to display brief messages to the user. Typically these include indicating the current state of the application and explaining menu and control items as the user focuses on them. The messages displayed in the DIL are called message items.

In Zeidon, a message bar is painted first and then message bar item controls are painted next. The same toolbar icon is used to paint both. After the message bar has been painted and sized appropriately, paint the message bar item control(s) by clicking and dragging inside the message bar. If the cursor is double-clicked on the message bar with focus the Message Bar Control Type dialog box is presented.



Specify the following information for the message bar:

**Tag** - Enter a name to identify the message bar

**Default Item** - Enter the number to identify the default message item. If a DIL message is sent without an item identifier, this is the item which will display the message.

If the cursor is double-clicked on a message bar item with focus the Message Bar Item Control Type dialog box is presented.

Specifications for the message bar item are as follows:

**Text** - Enter any text which you wish to initially appear in the item control.

**Text Style** - Select how you wish the messages displayed within the item. The choices are: Left Justified, Centered and Right Justified.

Once all the information in the Message Bar Item Control Type dialog box has been specified, press **OK**.

## Spread Sheets

Spread sheet controls are lists whose rows can contain edit boxes, combo boxes and push buttons. In Zeidon, a spread sheet group is painted first and then the individual items in the row are painted. Different toolbar icons are used for painting the spread sheet and its items. Paint a spread sheet group first, using the spread sheet icon, and size it appropriately. Next, click on the desired control (edit box, combo box, or push button) and paint the control items by clicking and dragging inside the spread sheet area.

**NOTE:** it is probably desirable to paint the controls within the spread sheet group starting in the upper left corner of the group box and working right.

---

### See also:

[Spread Sheet Area Details](#)

## Spread Sheet Area Details

If the cursor is double-clicked on a spread sheet area with focus the Update Spread Sheet Control dialog box is presented.

The screenshot shows the 'Update Spreadsheet Control' dialog box. It has a title bar with the text 'Update Spreadsheet Control'. Inside the dialog, there are several fields and buttons. The 'Tag' field contains the text 'SS1'. Below it, the 'Mapping Entity' field contains 'ACCOUNT' and the 'Entity' field contains 'TRANSACTION'. To the right of these fields are three buttons: 'OK', 'Cancel', and 'Common...'. Below the 'Mapping Entity' and 'Entity' fields is a 'Flags' section with a checkbox labeled 'No Integral Height'. At the bottom of the dialog is a list box titled 'Events / Related Actions' which contains two items: 'Cell Selected' and 'ResetText'. Below the list box is a button labeled 'PushButton1'.

Specify the following information for the spread sheet control:

**Tag** - Enter a name to uniquely identify the spread sheet within this window.

**View** - Select the view containing the entity to be listed from the View drop-down list.

**Entity** - Select the entity listed from the Entity drop-down list.

**Scope** - From the Scope drop-down list, select the scoping entity, if applicable.

**Events/ Related Actions** - The events for a spread sheet are listed in this list box. To add an action to an event click on the event and press the **Select Action** button. The Action List dialog box is presented. Double-click on the desired action from the list or single-click to select the action and press the **OK** button. Double-click on an event to specify or update an action to be invoked for that event. If you need to specify a new action, refer to [Creating Window Actions](#) for details on how to add actions to a window. Press the **Remove Action** button to remove the action from this control event or the **Update Action** button to update it.

**Hide Row Header** - If this check box is left unselected, a column is displayed listing the line numbers for the rows. If this check box is selected, this row header column will be hidden from view.

**No Integral Height** - If this check box is left unselected, the system will adjust the height of the spread sheet based upon the best fit of spread sheet items. That is, the spread sheet height will be set so that no row in the spread sheet is only partially displayed in the vertical direction.

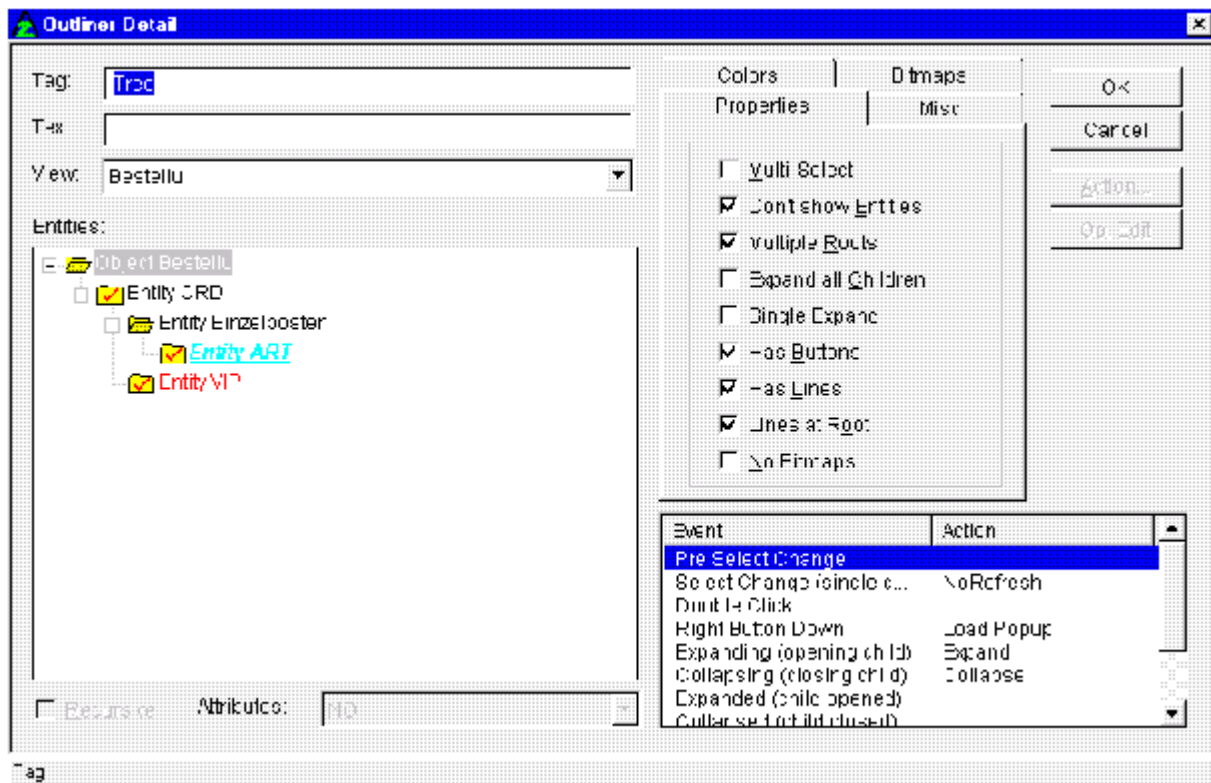
**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

If you double-click on any item within the spread sheet group the corresponding control type update dialog box will be displayed. Update the individual controls as documented for that control type.

## Outliner Controls

Outliner controls are used to display hierarchical data. More specifically, outliners are used to display instances of objects.

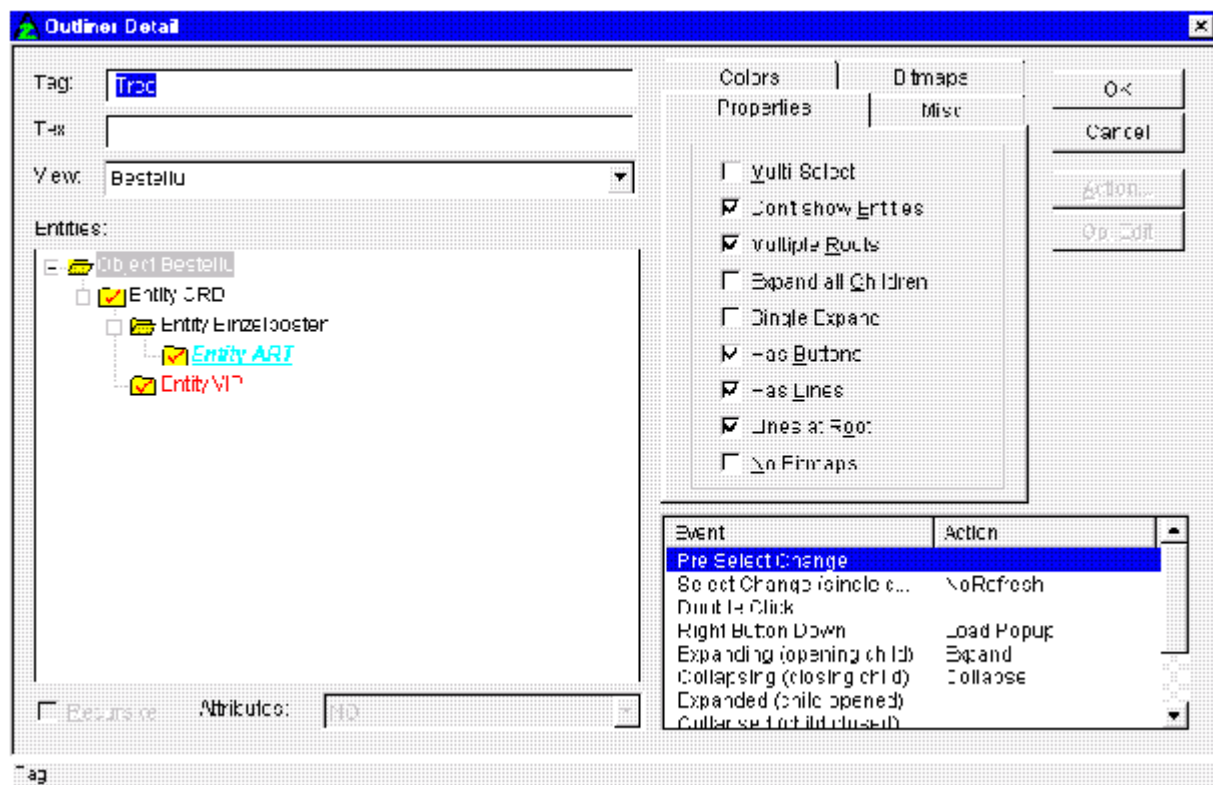
The Outliner control has been completely revised. It is **no longer downwards compatible**.



### See also:

- [Design Time of the Outliner Control](#)
- [Mapping Entities](#)
- [Outliner Control Properties](#)
- [Colors and Fonts](#)
- [Bitmap Property](#)
- [Default Values for New Outliner Controls](#)
- [Outliner Functions](#)

## Design Time of the Outliner Control




### See also:

- [Mapping Entities](#)
- [Outliner Control Properties](#)
- [Colors and Fonts](#)
- [Bitmap Property](#)
- [Default Values for New Outliner Controls](#)
- [Outliner Functions](#)



## Mapping Entities

- at runtime the Outliner only displays those entities which own the following bitmap at design-time  

- mapping an entity ensues via CTRL + left mouse-click. If the entity has already been selected to be displayed in the Outliner you may undo this selection via a second.
- As a novelty child-entities may be mapped, without having their direct parent on display. You may manage this by setting scope entities (see also: Property Multiple Roots).

Example: in the example stated above the entities "ORD", "ART" and "VIP" shall be mapped. Zeidon generates the following scope entities:

"ORD" ⇨ Scope = Object Scope

"ART" ⇨ Scope = "ORD"

"VIP" ⇨ Scope = "ORD"

in a second example only the entities "ART" and "VIP" are to be mapped.

"ART" ⇨ Scope = Object Scope

"VIP" ⇨ Scope = Object Scope

### See also:

[Design Time of the Outliner Control](#)

[Outliner Control Properties](#)

[Colors and Fonts](#)

[Bitmap Property](#)

[Default Values for New Outliner Controls](#)

[Outliner Functions](#)

## Outliner Control Properties

The Outliner offers a variety of properties which may be set at runtime.

The operation `OL_SetCtrlState` serves to set these during runtime.

### **Multi-Select:**

The new Outliner supports multi-selection via mouse and keyboard according to Windows standards. Analogous to the listbox-type "Extended" there is a differentiation of two possibilities:

#### Selection via control:

When selecting via Outliner control the select flag will be set in the view for the mapped entity.

#### Selection via code:

If the user applies the function `SetSelectStateOfEntity` to select entities in the view the control will display these only after a refresh.

Please note that the application of sets is generally not supported by the Outliner.

If multi-select is not set as property, the Outliner behaves like a listbox type "Single / Hilite Cursor Pos", i.e.:

#### Selection via control:

The selection in the Outliner may not be removed, thus there always exists exactly one mapped entity where the select flag in the view is set.

#### Selection via code:

If cursor movements occur in the mapped entities the entity where the cursor is positioned will always be displayed as selected after a refresh. All other select flags will then be removed.

### **Don't Show Entities:**

This property states whether the entity names should be hidden at display.

If this property is not set, the text to be displayed originates from the entity name + 1 blank + value of the mapped attribute.

### **Multiple Roots:**

Each entity which shall be displayed in the Outliner is saved by Zeidon with a scope entity. For all entities without parent applies that their scope is always the object scope.

If the user activates the 'Property Multiple Roots' the entities will be displayed with the object scope.

If this property is inactive Zeidon will only display the entities which belong to the root entity where the cursor is positioned.

### **Expand all Children:**

Via 'Expand all Children' the Outliner knows if the Outliner items should be opened or closed at the first loading.

In the old Outliner this property worked for each refresh. This is no longer the case. Now, the Outliner saves its expand status before refreshing and opens all Outliner items after the refresh, which have previously been opened.

If you wish to suppress this behavior please set the required expand status via the operation OL\_SetCtrlState before refreshing.

### **Single Expand:**

,Single Expand' is only accepted by the Outliner if the following properties are set:

Multi-Select = FALSE (or inactive)

Expand all Children = FALSE (or inactive)

The property ,Single Expand' ensures that only one Outliner item is open. A single click is sufficient to open an item, the double-click is not necessary.

If you click on an item which is already open, it will be closed.

### **Has Buttons, Has Lines, Lines at Root:**

In ,Has Buttons' the Outliner displays the plus- (+) and minus (-) – buttons next to the items which own child items.

In ,Has Lines' lines are used to indicate the hierarchy.

,Lines at Root' applies lines to connect the Outliner items with the root. This property is ignored if ,Has Lines' is not set.

### **No Bitmaps:**

For each mapped entity one default and one selected bitmap may be defined.

If one Outliner shall not display bitmaps this property must be set.

Please note that bitmaps may only be set at design- and runtime, if 'No Bitmap' is inactive.

### **Position X, Position Y, Size X, Size Y, Invisible, No Tab, Full Client:**

Analogous to the other Zeidon Controls these properties are also supported.

,Position X' or ,Position Y' respectively cause the Outliner to adapt its position to the enlarged window according to the activated coordinates.

,Size X' or ,Size Y' respectively ensure that the control's size is altered to the required coordinates.

,Invisible' hides the Outliner.

,No Tab' suppresses the selection of the Outliners via the TAB key.

,Full Client‘ causes the Outliner to take up the complete window on condition that ,Size X‘ and ,Size Y‘ are active.

### **Disable As Read Only:**

This enables/disables the editing of the Outliner item.

If this property is inactive there will be a switch into edit-mode at every selection of an Outliner item.

Please note that key attributes are generally not editable. This is also valid for other attributes whose values are not editable due to the LOD.

### **See also:**

[Design Time of the Outliner Control](#)

[Mapping Entities](#)

[Colors and Fonts](#)

[Bitmap Property](#)

[Default Values for New Outliner Controls](#)

[Outliner Functions](#)

## Colors and Fonts

The Outliner control supports setting the background colors, text colors and fonts.

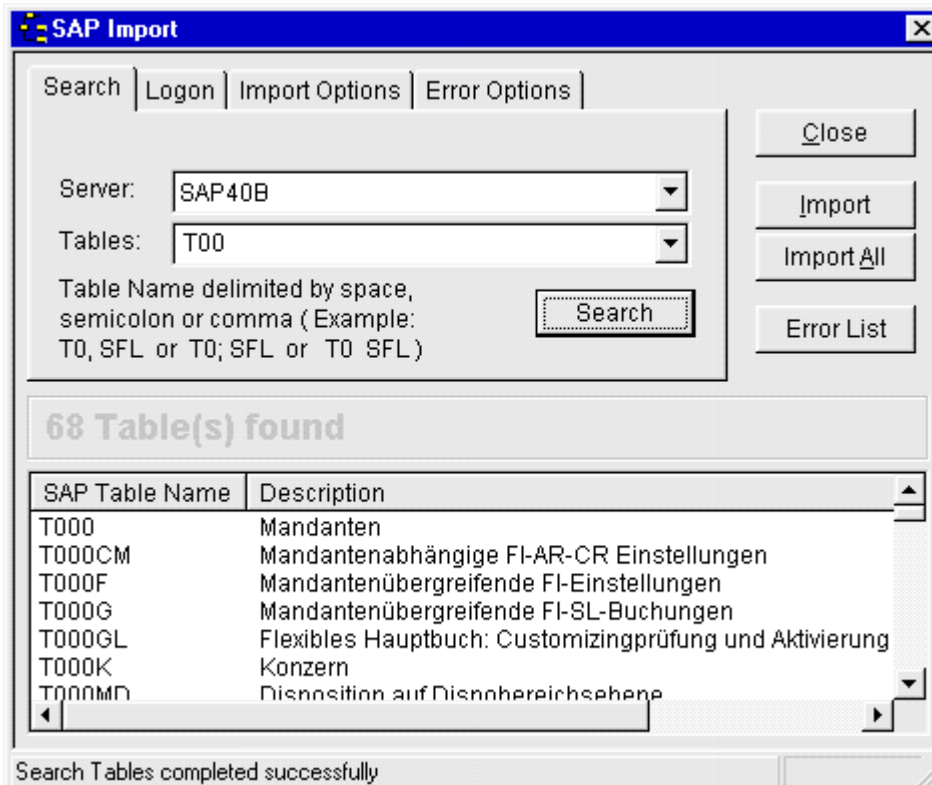
Text colors or fonts may be chosen freely for each mapped entity.

The text color may be set through the context menu in the Outliner or via the tab 'Colors'.

Text fonts on the other hand are only changeable through the context menu. The maximum font height is 16.

All colors and fonts are instantaneously displayed at design time and may still be altered to suite at runtime.

Please note that the Outliner saves the text color/font for the mapped entities, only (Entities with the bitmap



). If you set the colors/fonts for other entities they do no longer exist after closing the window 'Outliner Detail'.

### See also:

[Design Time of the Outliner Control](#)

[Mapping Entities](#)

[Outliner Control Properties](#)

[Bitmap Property](#)

[Default Values for New Outliner Controls](#)

[Outliner Functions](#)

## Bitmap Property

On condition that the property ,No Bitmap' is not active you may define two bitmaps for each mapped entity.

The ,Default Bitmap' is always displayed at runtime if the corresponding Outliner item is not selected. The ,Selected Bitmap' will be active upon selecting an item.

As default Zeidon offers the closed folder as ,Default Bitmap' and the opened folder as ,Selected Bitmap'. To change the bitmap please utilize the tab ,Bitmaps' or the Outliner's context menu.

For an optimal display we recommend to redefine the height and width of very large or small bitmaps via the tab ,Bitmaps' (default = 16). Please note that this value is valid for all bitmaps and may not be changed at runtime any more.

Analogous to the text colors/fonts Zeidon loses all bitmap-settings after closing the window "Outliner Detail" which have been set for unmapped entities.

### **See also:**

[Design Time of the Outliner Control](#)

[Mapping Entities](#)

[Outliner Control Properties](#)

[Outliner Control Properties](#)

[Colors and Fonts](#)

[Default Values for New Outliner Controls](#)

[Outliner Functions](#)

## Default Values for New Outliner Controls

Zeidon sets the following default values at the creation of a new Outliner:

- • Has Buttons
- • Has Line
- • Lines at Root
- • Disable as Read Only
- • Bitmap Height = 16
- • Bitmap Width = 16
- • Zeidon Default Bitmap for each entity
- • Zeidon Selected Bitmap for each entity

### See also:

[Design Time of the Outliner Control](#)

[Mapping Entities](#)

[Outliner Control Properties](#)

[Outliner Control Properties](#)

[Colors and Fonts](#)

[Bitmap Property](#)

[Outliner Functions](#)

## Outliner Functions

All Outliner functions always start with the prefix "OL\_" and are only applicable for this control.

Please refer to the 'Operations Reference' for a detailed description of the following functions:

### Survey:

Properties	OL_GetCtrlState
	OL_SetCtrlState
Outliner Items	OL_GetCurrentEntityName
	OL_SelectItem
	OL_EditLabel
Colors and Fonts	OL_GetBackgroundColor
	OL_GetTextColorForEntity
	OL_GetTextFontForEntity
	OL_SetBackgroundColor
	OL_SetTextColorForEntity
	OL_SetTextFontForEntity
Bitmaps	OL_GetDefaultBitmap
	OL_GetSelectedBitmap
	OL_SetDefaultBitmap
	OL_SetSelectedBitmap
Expand Status	OL_GetExpandState
	OL_GetExpandingEntityName
	OL_GetCollapsingEntityName
	OL_GetLastExpandedEntityName
	OL_GetLastCollapsedEntityName
	OL_ExpandEntity

### See also:

[Design Time of the Outliner Control](#)

[Mapping Entities](#)

[Outliner Control Properties](#)

[Colors and Fonts](#)

[Bitmap Property](#)

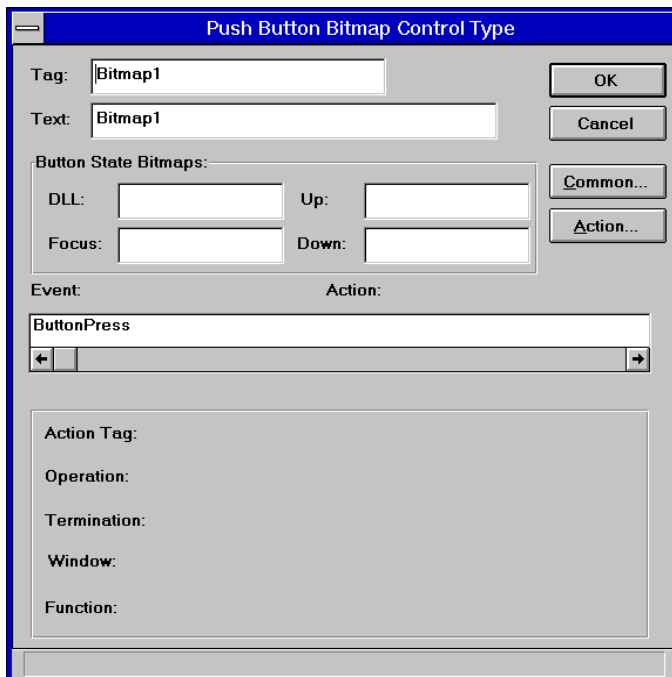
[Default Values for New Outliner Controls](#)



## Bitmap Push Buttons

Bitmap push buttons operate the same way regular push buttons do. The difference is they contain bitmap pictures instead of text. You can have three different bitmaps for the same button. Which one is presented depends upon the state of the button.

To define the bitmap push button characteristics, double-click on the bitmap push button with focus in the window image. The Push Button Bitmap Control Type dialog box is then presented.



The following information defines a bitmap push button.

**Tag** - Enter a name to uniquely identify the bitmap push button within this window.

**Text** - Enter any text which is to appear under the bitmap in the push button.

**DLL** - Enter the name of the DLL containing the resource compiled bitmap(s).

**Focus** - Enter the name of the bitmap you want displayed in the push button when the push button has focus.

**Up** - Enter the name of the bitmap you want displayed in the push button when the push button is in the up position.

**Down** - Enter the name of the bitmap you want displayed in the push button when the push button is selected.

**Event Action** - Double-click the ButtonPress event line to select an action to be invoked for this event. The Select Action dialog box is presented. Double-click on the desired action from the list or single-click on the action to highlight it and press the **OK** button. If you need to specify a new action, refer to [Creating Window Actions](#) for details on how to add actions to a window. Press the **Remove** button to remove the action from this control event.

**Action** - Press this button if you wish to update the specified Action for the ButtonPress Event. This will present the Action Detail dialog box. Refer to [Creating Window Actions](#) for details on updating actions.

**NOTE:** if a push button has an action assigned to it and if in the process of execution, that action is disabled by the application, the button will automatically be grayed. See "Disabling Window Actions" for more details.

**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

Once the fields in the Push Button Bitmap Control Type dialog box have been updated, press **OK**.

## Bitmaps

Zeidon allows for the inclusion of picture or bitmap controls. The file formats supported include: BMP and ICO.

To update a bitmap control, double-click on the control image in the window image. This will present a Bitmap Control Type dialog box.

Bitmap Control Maintenance

Tag: BmpCtl1

Text: BmpCtl1

DLL Name: account

File Name: C:\WINDOWS\PAT\_TAG.BMP

☒ Fit Bitmap to Size ☒ Use Device Dependent Bitmap

Event: Left Click, Left DblClick, Right Click, Right DblClick

Action:

Attribute Mapping

View:

Entity:

Attribute:

Context:

OK, Cancel, Browse..., Common..., Mapping...

The following fields may be updated:

**Tag** - Enter a name to uniquely identify the bitmap control within this window.

**Text** - Enter the text to be displayed if the specified bitmap is not found.

**DLL Name** - Enter the DLL name. Enter the DLL name only. Do not include the .dll extension.

**File Name** - Enter the name of the bitmap file to be displayed. Alternatively, you can press the **Browse** button to display the Open dialog box from which you can select the desired file.

**Fit Bitmap to Size** - If this check-box is selected, the bitmap will be resized to fit into the size of the bitmap control. NOTE: The aspect ratio of the bitmap will not be maintained if the bitmap is resized so, circles may appear as ovals and squares as rectangles. If this check-box is not selected the bitmap will displayed with the upper left corner of the bitmap in the upper left corner of the bitmap control. If the bitmap is larger than the control in either the horizontal or vertical direction, it will be cropped. If it is smaller, it will not fill the control.

**Use Device Dependent Bitmap** - If this check-box is selected, the bitmap will be displayed with device dependent colors.

**Event Actions** - The events for a bitmap control are listed in this list box. Actions defined for that window are listed in the Window Action list box. To assign an action to a control event highlight the event, highlight the action to be assigned from the Window Actions list box, and press the **Select** button. To remove an action

from an event highlight the event and press the **Remove** button. If you need to specify a new action, refer to [Creating Window Actions](#) for details on how to add actions to a window.

**Mapping** - Press the **Mapping** button to present the Select Attribute Mapping dialog box and to specify data mapping for the control. Refer to [Data Mapping for Controls](#) for details on attribute selection. The value of the attribute selected should be the file name of the bitmap control. If mapping is selected, you can use the Context combo-box to select the proper context for the mapping attribute.

**Common** - Press this button to specify common information such as color, message bar text, proportional sizing specification and presentation behavior. This information is described in detail in [Common Details for Controls](#).

Once the fields in the Bitmap Control Type dialog box have been completed, press **OK**.

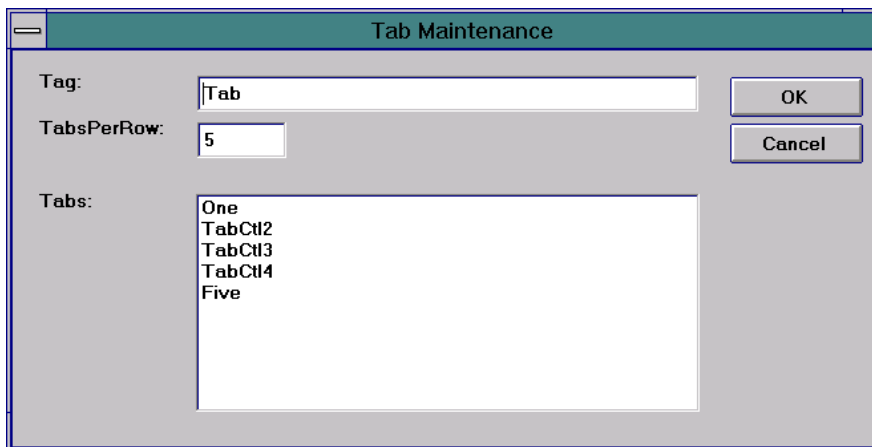
## Tabs

Tab controls allow multiple "pages" to be stacked on top of each other on a single window. By selecting the visible tabs attached to each page, the user can bring the appropriate page to the top where it is visible.

In Zeidon, a tab control is painted first and then the individual tab pages are painted next. The same toolbar icon is used to paint both. Paint a tab control first and size it appropriately. Next, paint the individual tab pages within the tab control by clicking and dragging. The size of the individual tab pages will automatically be adjusted to fill the tab control regardless of how big or small you paint them. For each tab page painted, you will see an associated tab displayed along the top of the tab control.

To paint other Zeidon controls on an individual tab page, click on the tab associated with that page to bring the page to the top. You can then paint Zeidon controls on the tab page as usual.

To define a tab control's characteristics, double-click on the tab control with focus in the window image. The Tab Maintenance dialog box is then presented.

The image shows a dialog box titled "Tab Maintenance". It has a light gray background and a dark green title bar. Inside the dialog, there are three main sections. The first section is labeled "Tag:" and has a text input field containing the word "Tab". To the right of this field are two buttons: "OK" and "Cancel". The second section is labeled "TabsPerRow:" and has a small text input field containing the number "5". The third section is labeled "Tabs:" and contains a list box with five items: "One", "TabCtl2", "TabCtl3", "TabCtl4", and "Five".

The following fields may be updated:

**Tag** - Enter a name to uniquely identify the bitmap control within this window.

**Tabs Per Row** - Enter the maximum number of tabs that will be displayed in a single row across the top of the control. If the number of tabs in the control exceeds the maximum Tabs Per Row, additional rows of tabs will be displayed.

**Tabs** - To update the characteristics of a tab, double click on the tab in the Tabs list box. This will open the Tab Update window from which you can update the Text to be displayed on the tab and the unique Tag to be associated with the tab page. Once the Text and Tab fields have been updated, press **OK** to return to the Tab Maintenance dialog box.

Once the fields in the Tab Maintenance dialog box have been completed, press **OK**.

## Autodesign

The Zeidon autodesign feature automatically creates application windowing dialogs based on the structure of a Logical Object Definition (LOD) and the content of the data model. Zeidon can autodesign an entire windowing dialog or autodesign attributes into an existing window. You can autodesign dialogs using mostly default settings that require little design effort, or you can specify detailed information, controlling the order of attributes on each window, literal content and function, creating a Dialog that more precisely matches your objectives. Once the dialog has been autodesigned, you can also fine tune the autodesigned windows using the Zeidon dialog painter.

The autodesign process uses three different Zeidon application components: a Logical Object Definition (LOD), a user interface specification (UIS), and a windowing dialog.

- A LOD describes the structure and relationships of the application data. Zeidon uses the LOD to determine the structure for the autodesigned dialog. Therefore, before you autodesign a dialog you must first create a LOD using the Zeidon Objects tool. Note that the LOD inherits the required information from the data model, such as relationship cardinality and entity type.
- From the LOD, you create a user interface specification (UIS). The UIS allows you the flexibility to determine how your application will look and run when it is completed. Although the structure of the UIS and the resulting autodesigned dialog are partly determined from the characteristics of the data model and LOD, you can control attribute content, prompt and button text and some button actions using the UIS.

**Note:** If you are autodesigning an individual window, you do not have to create a UIS, but you must have an existing LOD.

---

- When you initiate autodesign from the Dialogs tool, Zeidon automatically creates an application windowing dialog using the information in the UIS or adds attribute content to an existing window. Autodesigning a complete dialog (Autodesign Dialog) requires a UIS, while adding attribute content (Autodesign Window) requires only a LOD.