User's Reference Guide

COPYRIGHT
1989 - 2000 by **ZUCCHETTI TOOLS** S.r.l.
All rights reserved.
This publication contains information protected by copyright. This publication may not be reproduced, stored, or transmitted, in any form or by any means without the prior permission of the publisher.

TRADEMARKS
All produced trademarks are ownership of the holder and are acknowledged by this publication.

ZUCCHETTI TOOLS S.r.l. SOFTWARE TECHNOLOGY
PADOVA - BELLARIA - RIMINI
E-mail: clabrn@codelab.it
Web Address:
http:\\www.zucchettitools.com
http:\\www.codepainter.com
http:\\www.codelab.it

Summary

Chapter 1

Introduction

1.1 Introduction

The User's Reference Guide is your reference to the product.

The following chapters have been structured to give you an exhaustive and easy introduction to the product.

The beginning of each chapter explains the logic of single tools. The chapters then go into more detail explaining menus and options.

Chapter 2

Front End

2.1 Introduction

CODE**PAINTER REVOLUTION** is not a monolithic program that realizes all tasks described in this User Reference Guide using one executable function only. It is rather made of various integrated tools that help you creating and maintaining Business/Commercial Applications.

Each task is performed by a dedicated tool that is written in the language that best achieves the result.

CodePainter's Front-End is a user friendly interface that brings together all the tools making up the development environment.

This chapter analyzes the functions of CodePainter's FrontEnd.

2.2 The Front End Working Logic

The Front End dialog window contains all options required to create and manage projects.

Picture 1 - CodePainter Front End

Let us now briefly describe the drop down menus.

The 'File' menu allows creating and/or selecting the project directory.

The 'Generation' menu allows managing the generation of prototypes, code and documentation.

The 'Project' menu allows defining and organizing both, the project language and the directory.

The 'View' menu allows enabling and disenabling the 'Utilities' toolbar, which makes the interaction with the tool easy to use.

The 'Help' menu gives you detailed information that help you exploiting this powerful tool.

2.3 The Front End Toolbar

The Front End toolbars allow interacting directly with the various Painters.

You can move these toolbars using the mouse from the window's top to the side. You can also drag and drop it anywhere within the Front End window simply clicking the toolbar and pressing **<Ctrl>**. Further details are explained in the following sections.

2.3.1 Painter Tools Toolbar

This toolbar allows opening CodePainter's Visual Tools. The 'Painter Tools' functionality will be explained in the following chapters.

Picture 2 - The Painter Tools Toolbar

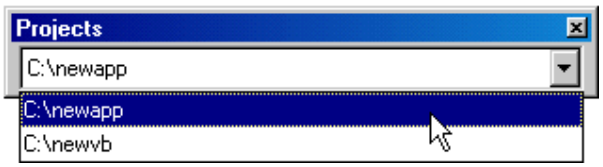
Here to follow you will find a brief explanation of the toolbar buttons.

Note
The 'Painter Tools' toolbar access the same functions as the 'Painters' menu.

Button	Meaning
	Opens the Design Painter.
	Opens the Master Painter.
	Opens the Detail Painter.
	Opens the Master/Detail Painter.
	Opens the Dialog Window Painter.
	Opens the Routine Painter.
	Opens the Documentation Painter.
	Runs the Design Generation.
	Runs the Code Generation.
	Runs the Documentation Generation.

2.3.2 The Projects Toolbar

This toolbar contains the list of the last opened projects.

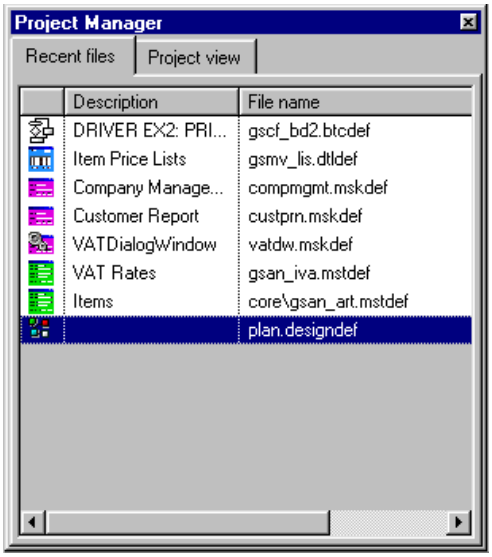


Picture 3 - The Projects Toolbar And The List Of

CodePainter stores information on the last opened project in the file C__<ApplicationDirectoryName>.Recent under the 'CPR' directory in the root of the user's disk. This allows you changing directory with a mouse click and access an existing project quickly and easily. Last modified files can be retrieved in 'Recent Files'.

2.3.3 The Project Manager Toolbar

This toolbar contains information on the current project.



Picture 4 - The Project Manager Toolbar And The List Of Modified Definition Files

CodePainter stores information on the current project in the 'CPR.MEM' file, under the application directory.

You can always check information on the application organization and the working progress.

Let's now analyze the 'Project Manager' window and the toolbar options.

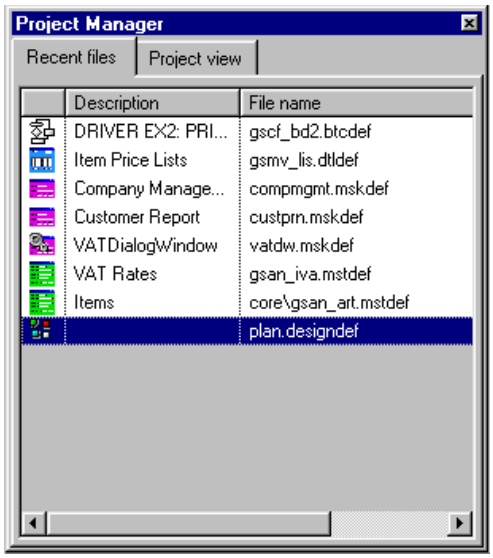
Recent Files

This sections details the latest modified definition files. If files are not changed after the creation of the project design, the list is empty.

The first column on the left displays icons that define the entity classes. The same icons are in the 'Painters' menu. The '*gear*' on an icon means that the entity must be re-generated.

The second column displays the entity description. The third column displays file names.

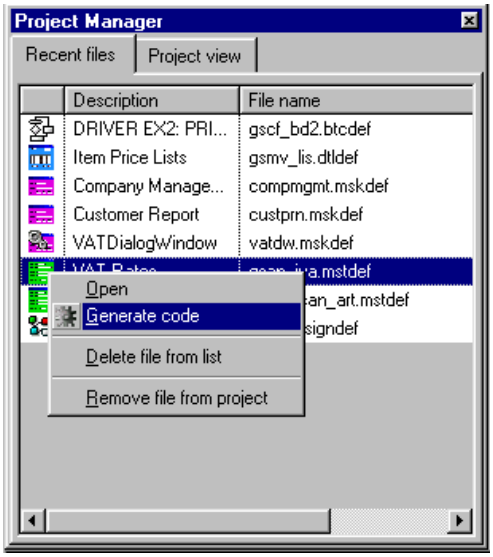
Clicking column headings you can change the list's order. Entities can be sorted by type, description or file name.



Picture 5 - The Project Manager Toolbar And The List Of Modified Definition Files Ordered By Description.

Right clicking items on the list the correspomding menu is opened. It allows executing Codify related tasks. The 'Open' option opens the 'Codify' phase for the selected entity. The 'Generate Code' option runs the code generation for the selected entity. The 'Delete File From List' option deletes the selected entity from the list whereas the 'Remove File From Project' option deletes the entity from the project.

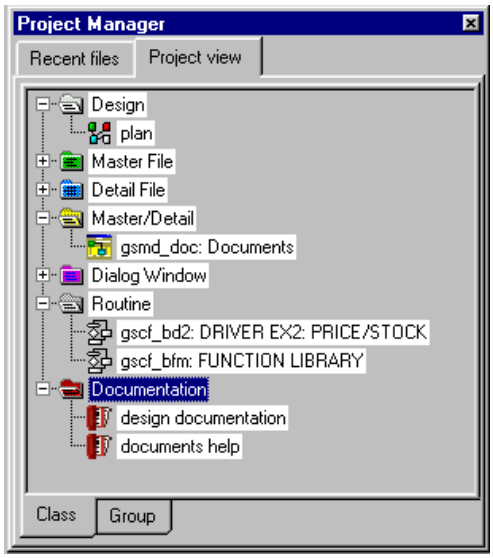
Picture 6 - Recent File Section And The Right Clicked Menu



When files are deleted from the list the window is updated. When files are deleted the 'Project View' tab-strip is updated as well. Both options do not physically delete files.

Project View

The 'Project View' tab-strip has two folders: 'Class' and 'Group', in which project files are displayed.



Picture 7 - The Project View Tab-strip

Class

In this folder project files are treeviewed by class.

Clicking a class the 'Add Files To Project' option is displayed. Using this option you can add files to the project. The 'Open File' window is opened and you can select one or more files to be added to the project. Differentiated selection can be made either pressing 'CTRL' or 'SHIFT'.

Once a class is expanded you can right click on one of the items in the list. The menu that allows executing Codify related tasks is opened. The 'Open' option opens the 'Codify' phase for the selected entity. The 'Generate Code' option runs the code generation for the selected entity. The 'Remove File From Project' option deletes the entity from the project.

When files are deleted from the project, files are not physically deleted.

Group

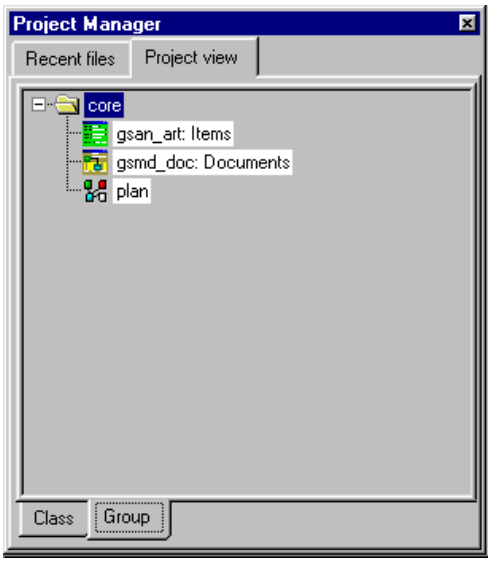
This folder treeviews project files grouped together using the 'Group' option in the 'Project' menu.

Example

As you go on developing your application the project complexity grows as well as the number of files defined. To maintain a clear overview of the project you need to create subdirectories in your working directory so as to group files logically.

To add files to the logical group 'Files' open the 'Project' menu, select the 'Group' option, create the subdirectory 'Files' and define which files belong to that group.

Picture 8 - The 'Project View' Dialog Window And The 'Group'



The new subdirectory is integrated in the Treeview displayed in the 'Group' folder.

Right clicking a group you can add/ modify the group using the 'Add/Modify Group' option or remove it. For more information please refer to the 'Project' menu, 'Group' option.

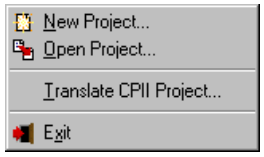
Removing a group from the project the treeview is updated, but files are not physically deleted.

Right clicking entities in the group a menu is opened. It allows executing Codify related tasks. The 'Open' option opens the 'Codify' phase for the selected entity. The 'Generate Code' option runs the code generation for the selected entity. The 'Delete File From List' option deletes the selected entity from the list whereas the 'Remove File From Project' option deletes the entity from the project.

2.4 Main Menu

2.4.1 File

The 'File' menu has a set of options to maintain projects.



Picture 9 - File Menu

From this menu you can:

- Create new projects.
- Load existing projects.
- Port old projects.
- Exit CODEPAINTER REVOLUTION Front End.

New Project

In the 'New Project' option you can define the directory in which the new project is created as well as the source language.

For more detail please refer to section **'Creating A New Project'**.

Open Project

In the 'Open Project' option you can select the directory from which the project can be opened.

Translate CPII Project

The 'Translate CPII Project' option ports projects developed using CodePainter II in CODEPAINTER REVOLUTION, opening a new project.

This menu item is enabled only when an old project directory is opened.

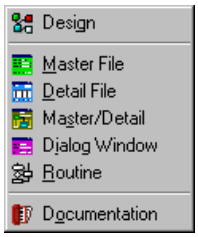
For more detail please refer to **'Porting Old Projects'**.

Exit

The 'Exit' option exits the frontend.

2.4.2 Painters

Using the Painters menu you can access any available tool. The 'Painters' menu has the same options as the 'Painter Tools' Toolbar.



Picture 10 - Painters Menu

Here to follow you will find a brief description of the tools. For more information please refer to the dedicated sections (Design Painter, Master File Painter, etc.)'.

Design

Opens the Design Painter to create and manage design files.

Master File

Opens the Master/File Painter to create and manage Master File entities.

Detail File

Opens the Detail File Painter to create and manage Detail File entities.

Master/Detail File

Opens the Master/Detail Painter to create and manage Master/Detail File entities.

Dialog Window

This option opens the Dialog Window Painter, which creates and manages Dialog Window entities.

Routine

Opens the Routine Painter to create and manage Routine procedures and functions.

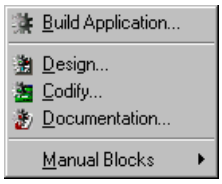
Documentation

Opens the Documentation Painter to create and manage documentation.

2.4.3 Generation

Using the 'Generation' menu you can generate any project's element and add and/or extract manual areas in the generated source code.

Picture 11 - Generation Menu



For more information please refer to '**Generating Project Files**'.

Build Application...

The 'Build Application' option executes the three generation options in sequence: Design Generation ('Design...'), Code Generation ('Codify...') and Documentation Generation ('Documentation...'). The three generation options can also be executed separately.

Design...

The 'Design' option generates the design.

Codify...

The 'Codify' option generates the source code for the all entities in the project.

Documentation...

The 'Documentation' option generates the project's documentation.

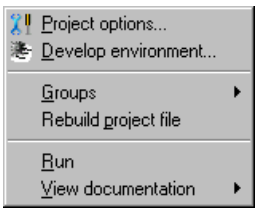
Manual Blocks

The 'Manual Blocks' option allows extracting manual areas, which have been typed directly in the source code.

For more information please refer to '**Extracting Manual Areas**'.

2.4.4 Project

The 'Project' menu allows maintaining the working project.



Picture 12 - Project Menu

Here to follow there is a brief description of the main functions. For more information please refer to 'Project Maintenance.

Project Options...

Defins project options for large applications and/or for multi-users.

Develop Environment...

Defins the project's target language, the development environment directory and the template directory.

Groups

Creates and modifies project groups.

picture 13 - Groups

Run

Executes the project application.

View Documentation

Displays a first draft of the design documentation containing the description of the project's main module, i.e. the section defined in the 'Main Design File' field in the 'Project Options...' of the 'Project' menu.

2.4.5 View

Using the View menu you can enable or disenable the tabstrips in 'Project View' and the various FrontEnd toolbars.

Picture 14 - View Menu

Project Tabstrips

Enables or disenables tab-strips in the 'Project View' window of the 'Project Manager' toolbar.

Toolbars

Enables or disenables FrontEnd toolbars.

2.4.6 Help

The 'Help' menu provides an on-line help or gives you information on the product.

picture 15 - Help Menu

Contents

The 'Contents' option displays the On-line Help structure.

Index

The 'Index' option displays the index.

Search

The 'Search' option allows you searching for topics in the On-line help.

About

The 'About' option displays information about the product.

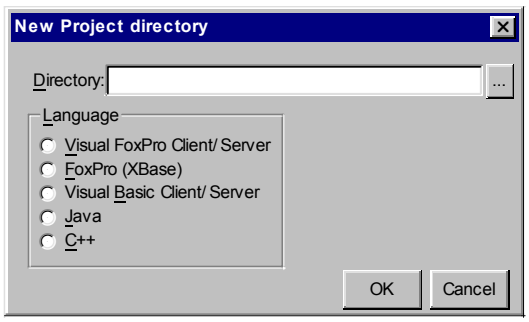
For more information please refer to '**Product Information**'.

2.5 New Project Creation

When you want to start a new project you need to specify the working directory and the language used for code generation in the 'New Project Directory' window.

2.5.1 New Project Directory

Picture 16 - New Project Directory: Page 1



NewApplic

Directory in which the new project will be created. When the defined directory does not exists it is created and all CodePainter files required to start developing are copied under this directory.

...

language

Source language in which the code for the new project will be generated.

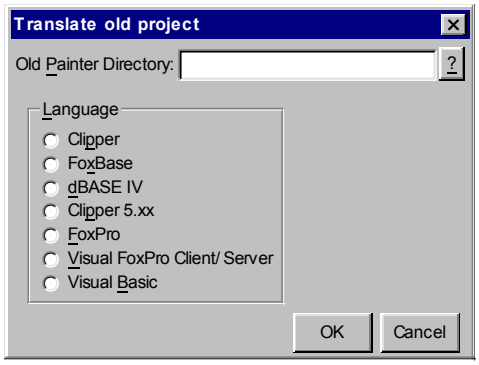
2.6 Porting Old Projects

With CODE**PAINTER REVOLUTION** you can port old projects created in CODEPAINTER II. The following section explains the main insrtructions required for project porting.

2.6.1 Translate Old Project

Allows to translate projects created with CodePainterII. Selecting a project created with CodePainter II the translation process is started from the repository. When the translation process is interrupted or it has not been started by the user, you can re-execute it. CPR recognizes the old format reading CP2.MEM. If this latter file is included in the directory the translation process can be re-executed. When the process is ended old definition files are saved under the directory 'OldDefs' including CP2.MEM. To restore the old project you simply need to copy these old files under the project directory.

Picture 17 - Translate Old Project: Page 1



paintdir

CodePainter II directory. To translate old projects you need both CodePainter II (build 30 up-wards) and CodePainter Revolution.

language

Source language used by the old project.

2.7 Generating Project Files

CodePainter supports project development in all life cycle stages. A project in CodePainter is started with the *Design* phase in which you create a project plan. The plan leads to the prototype creation that uses design generators.

Prototypes can be considered bridges between the Design and the Codify phase.

Prototypes on one hand allow implementing project designs very quickly and take make changes in an early stage of the project. On the other hand prototypes are the first step towards the Codify phase. Indeed you do not need to start anew but you simply need to improve what you already did using standard methods.

Prototypes help you refining the Design phase. Prototypes are made so quickly that you can discuss requirements with your customer while constructing new options and thus being able to show changes immediately.

CodePainter offers a set of specialized tools to define predefined classes. Using these tools you can improve the dialog window's layout and add new features to elements in the window. You can do that in a comfortable WYSIWYG (What You See Is What You Get) environment driven by the mouse.

New introduced features are automatically translated in executable code basing on specialized templates and using Codify generators.

CodePainter gives you the project overview at different detail levels. Using the Documentation tool you can create different types of documentation. This tool simply assembles definitions input at the various project levels.

Definitions input in the Design Phase form the technical documentation, which contains an overview on the software project, the database description, and the list of procedures that must be created.

The technical documentation is developed in an early stage of the project and can therefore be used to make cost estimates and to define user requirements in detail.

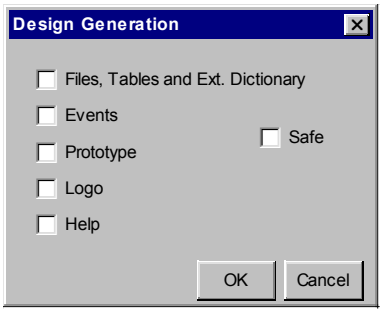
CodePainter creates Reference Guides and the On-line Help combining designed definitions with developed dialog windows. Manuals are therefore always up to date with the latest SW application release.

Let us now analyze the various options that are supported by the generators of the different phases.

2.7.1 Design Generation

Runs generations related to the design plan.

Picture 18 - Design Generation: Page 1



Files, Tables and Ext. Dictionary

Creates the extended dictionary .XDC. The dictionary is used by the system when the application is run in order to maintain the database.

Events

For future use.

Prototype

Generates prototypes. Creates the prototype for each entity that has the Prototype flag set. If the prototype already exists the system asks you if you want to overwrite it. The new prototype cancels all changes made with the Codify tools. In order to keep changes to the prototype you need to deselect the Prototype flag in the design plan. To take over the new changes made in the design plan in the Codify dialog window you need to use the 'Design Compare' option in the 'Edit' menu.

Logo

Creates the application logo.

Help

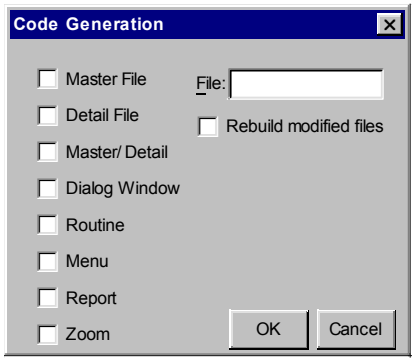
Creates the help application.

Safe

Defines whether to ask to confirm the prototype overwriting for existing definition files.

2.7.2 Code Generation

Source Code generation. This dialog window runs the generators that produce the source code basing on the contents of the repository and the project definition.



Picture 19 - Code Generation: Page 1

Master File

Generates the source code for Master File entities.

Detail File

Generates the source code for Detail File entities.

Master/Detail

Generates the source code for Master/Detail entities.

Dialog Window

Generates the source code for Dialog Window entities.

Routine

Generates the source code for Routine entities.

Menu

Generates the application's menu.

Report

Generates the source code for Report entities.

Zoom

Generates the source code for Zoom entities.

Files

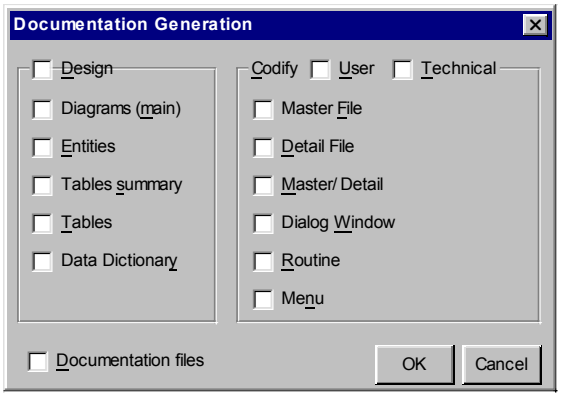
Filter on the files that must be re-generated. Allows to define a file name in order to generate specific files only. The character '*' lists the contents of the directories. The generated files are those contained in the project file, no matter in which directory they are.

Rebuild Modified Files

Re-generates all files. When this flag is set the system generates only those files having a previous program date than the corresponding definition file. When this flag is not set all files are regenerated.

2.7.3 Documentation Generation

Generates the documentation.



Picture 20 - Documentation Generation: Page 1

Design

Generates the documentation starting from the design plan. When the flag is set the documentation is generated including the notes defined in the design plan.

Diagrams (main)

Generates the main diagram.

Entities

Generates information on all entities.

Tables summary

Generates summary documentation on the various records layout.

Tables

Generates extended documentation on the various records layout.

Data Dictionary

Generates the data dictionary.

User

Generates User documentation starting from Codify tools.

Technical

Generates Technical documentation starting from Codify definitions.

Master File

Generates documentation on 'Master Files'.

Detail File

Generates documentation on 'Detail Files'.

Master/Detail

Generates documentation on 'Master/Details'.

Dialog Window

Generates documentation on 'Dialog Windows'.

Routine

Generates documentation on 'Routines'

Menu

Generates documentation on the menu.

Documentation files

Generates documentation on publications.

2.8 Extracting Manual Areas

CodePainter automatically generates the required code, but sometimes you may need to manually integrate it. You can manage manual areas inputting and extracting them.

Manual areas can be added in the Codify Phase, opening the Globals menu and selecting the Manual Blocks option. Using this option you can add code in single entities.

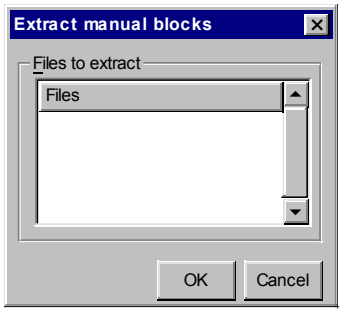
When the application is regenerated manual areas are maintained. The source code of generated procedures is a regular ASCII file and therefore you can always work directly on the code. Simple generators would write the code anew, thus loosing the manual area. This would also have repercussions on the project life cycle as applications could not be easily maintained.

CodePainter allows to extract manual areas and to save the manually written code in the entity definition file. Changes are therefore maintained.

In the 'Manual Blocks' option you can run the extraction defining the list of files that must be analyzed in the 'Extract Manual Blocks' dialog window.

2.8.1 Extract Manual Blocks

List of files that must be analyzed.



Picture 21 - Extract Manual Blocks: Page 1

File Filters

List of files that must be analyzed.

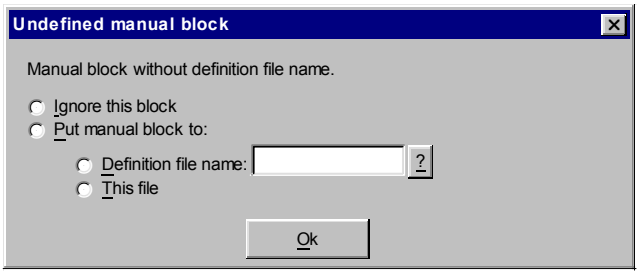
This list contains specifications of the files that must be analyzed during the extraction of Manual areas. Elements in the list follow wildcards rules for directories: e.g."*.prg" analyzes all files having the extenction prg.

When manual areas contained in the source procedure are not defined correctly CodePainter displays the 'Undefined Manual Block' dialog window that allows making required changes to extraction parameters.

2.8.2 Undefined Manual Block

When you write Manual areas you need to define a first comment containing the name of the area and a second comment detailing in which definition file it must be re-generated.

In case you forget to define the second comment in the source code this dialog window allows you to define it afterwards.



Picture 22 - Undefined Manual Block: Page 1

manual_option

Ignore this block

The Manual area is skipped. It is not added to the definition file.

Put manual block to:

Saves the Manual area in the definition files defined below.

file_option

Definition file name:

Saves the Manual area in the file defined in the textbox next to it.

This file

Save the Manual area in the definition file having the same name as the analyzed program.

OK

Button to confirm the dialog window.

Filename

Name of the file in which the Manual area must be saved.

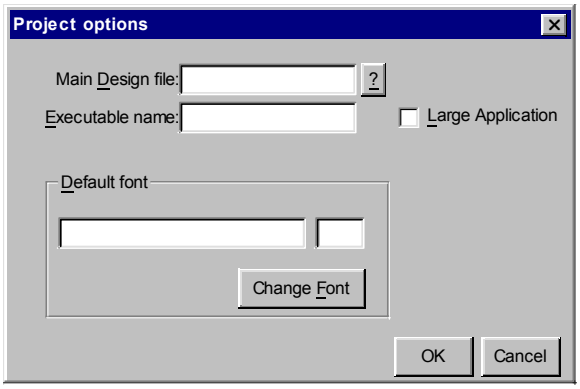
2.9 Project Maintenance

You can maintain your project using the options defined in the 'Project' menu.

2.9.1 Project Options

Project definition.

Picture 23 - Project Options: Page 1



planfile

Name of the main design file. Projects can have more than one design plan. This textbox defines which file must be considered the main one. Usually the first design plan created within the project is called 'PLAN' and is considered the main one.

nameexe

Name of the executable file. If the selected source language requires an executable file, then it will be automatically created using this name.

Multiuser

Defines whether the application is multiuser.

Large Application

Defines whether the application is made of many files. All files created by CodePainter are in the project's directory. Large applications have many files. Setting this flag you can automatically spread files on more project's subdirectories. The files contained are linked to each other:

EXE

Contains information to execute the project, i.e. all files that must be delivered to the customer.

SRC

Source code files. The effective name of this directory changes depending on the source language selected. This enables you to create different source files basing on the same project.

OBJ

When the selected source language requires object files, i.e. the compilation result, files are saved under this directory. Again, you can have different directories for the different source languages.

DOC

Documentation files. This directory is created even when the flag is not set. The main project's directory contains all definition files that make up CodePainter's repository. You can divide these files in sub-directories creating project groups using the dedicated Front-End tab-strip.

Change Font

Clicking the button the Global Font Definition window is opened.

txtFont

Default font. Entity prototypes are created using this font.

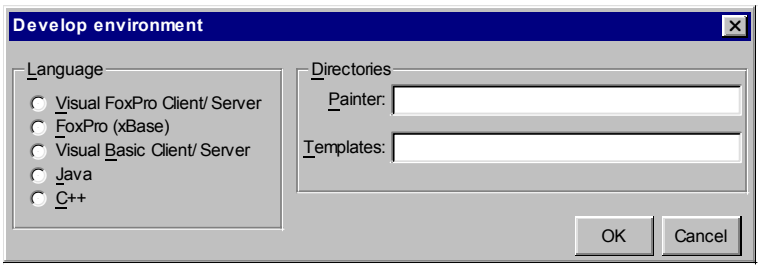
txtPoints

Points of the default font. Entity prototypes are created using these font points.

2.9.2 Develop Environment

Definition of the development environment.

Picture 24 - Develop Environment: Page 1



language

Defines the programming language in which the source application is created.

paintdir

The painter directory is defined automatically.

tpldir

Directory from which templates are taken. The directory is automatically defined basing on the selected source language.

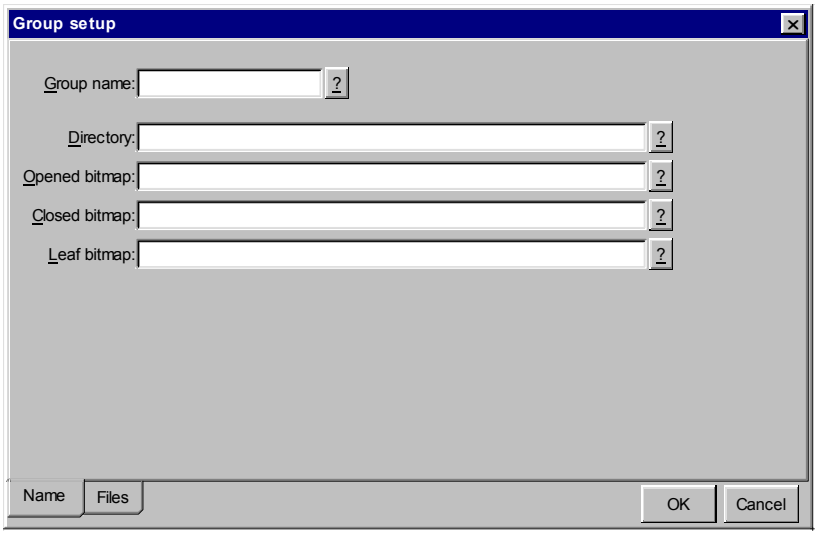
The 'Groups' option opens a submenu that allows adding/changing and deleting groups in your project.

2.9.3 Group Setup

Displays group properties and allows changing them. The 'Files' page is enabled when at least the group's 'Name' property has been defined. Using groups you can organize files in the current project. You can group entities according to the entity type ('Master Files', 'Routine' group, etc.), or according to functional criteria ('Accounting', 'Warehouse' group, etc.). Using groups you make large projects more understandable.

Main Page

Group definition property.



Picture 25 - Group Setup: Page 1

Group name

Name of the group. Brief description that identifies the group of files uniquely. This property can have a different name from the directory name. You can create groups without creating project's subdirectories.

Directory

Location where selected files are moved to.

Opened bitmap

Name of the bitmap file ('BMP') displayed when the group branch is expanded. Groups can be treeviewed in the 'Group' tabstrip or in the 'Project View' toolbar on the Front End.

Closed bitmap

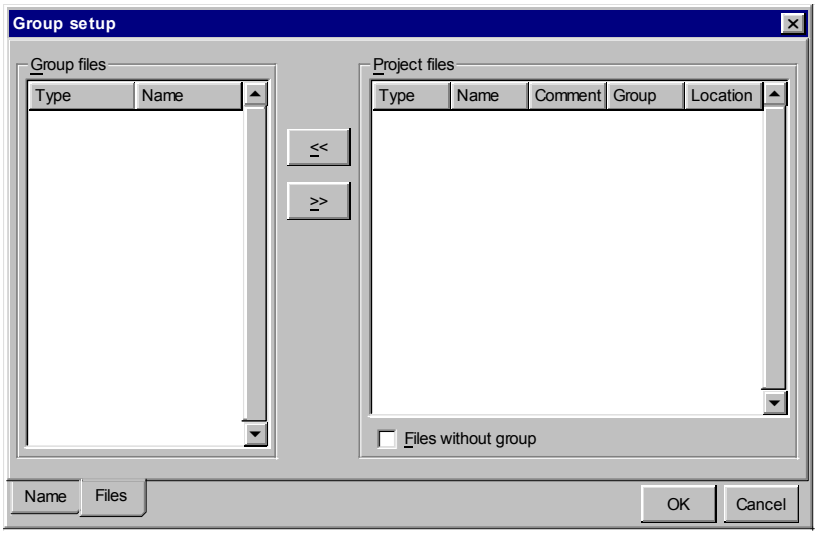
Name of the bitmap file ('BMP') displayed when the group branch is collapsed. Groups can be treeviewed in the 'Group' tabstrip or in the 'Project View' toolbar on the Front End.

Leaf bitmap

Name of the bitmap file ('BMP') displayed in the leafs of the current group, i.e. in the files that belong to the group. Groups can be treeviewed in the 'Group' tabstrip or in the 'Project View' toolbar on the Front End.

Files Page

List of definition files belonging to the group.



Picture 26 - Group Setup: Page 2

Group files

List of files belonging to the current group. You can change the list using the '<<' and '>> buttons.

<<

Moves files selected from the list of projects files to the list of files belonging to the current group.

>>

Moves selected files from the list of files belonging to the current group to the list of project files.

Project files

List of files belonging to the current project.

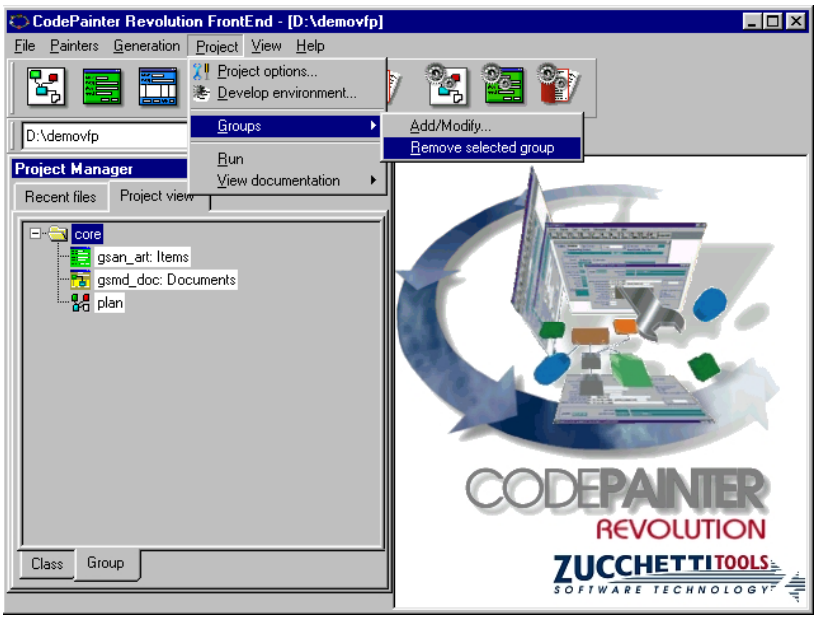
Files without group

Displays project files that do not belong to any group.

2.9.4 Remove...

The 'Remove' option allows deleting selected groups from the 'Project View' tab-strip in the 'Project Manager' toolbar.

Picture 27 - To delete selected groups from



Deleting groups from the project updates the 'Project View' organization and layout, but does not physically cancel selected files.

2.10 Product Information

The option 'About' in the 'Help' menu displays a set of information on the product.

2.10.1 About

The About dialog window contains information on the procedure in use.

The logo CODEPAINTER REVOLUTION is displayed and information on the version, series number and build number are given.

The build number helps you identifying CodePainter's distribution, no matter if the product is released totally or partially or through patches.

Picture 28 - About: Page 1

Chapter 3

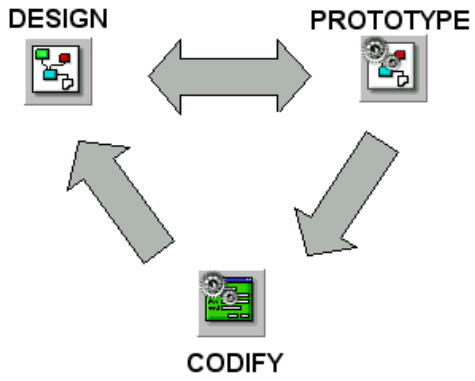
Design Painter

3.1 Introduction

To guarantee the success of a project you require to analyze the issue into detail and design the solution properly. The Project Manager's task is to provide overall solutions starting from the HW and SW that must be used. Even little mistakes or misunderstandings during the Design phase will have repercussions throughout the project. The later mistakes are identified the more costly it becomes to fix them.

The Design Painter combined with the Prototype phase allow you to get a project overview quickly that you can share with the customer. Corrective action, if required, can be taken in early stages of the project thus guaranteeing high quality. It further builds the basis for the Codify phase.

Picture 29 - The Design Cycle



The Design Painter also allows you to comment each defined element. This means that you can put notes against programs, procedures, files, fields, etc. These notes will be used to write technical and/or user reference documentation.

The Design Painter supports SW applications throughout their life cycles. You can mix existing entities with new ones, add new parts in the application or delete obsolete parts. The prototype can be performed for certain parts only or for the whole design, depending on your requirements. Documentation, databases, global definitions and menus are always automatically realigned with the project design.

3.2 The Working Logic

Selecting the Design Painter a plain worksheet and a set of toolbars are opened. The worksheet is a blank plan on which you will start painting your design.

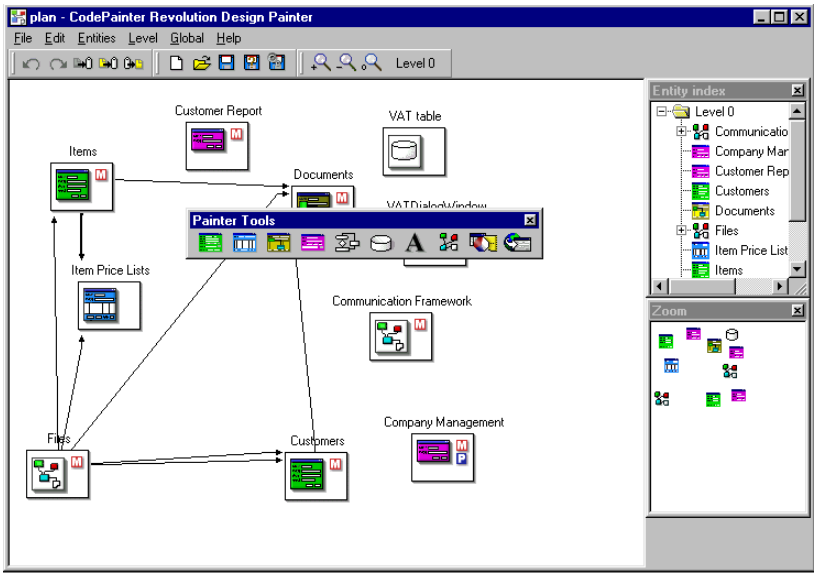


figura 1 - Design Painter

Here to follow you will find the list of icons representing the various entities in different statuses.

Symbol	Meaning
	Master File Entity
	Detail File Entity
	Master/Detail Entity
	Dialog Window Entity
	Routine Entity
	Database Table Entity
	String Entity
	Group Entity
	Output Entity
	External Entity
	Incomplete Entity
	Menu Flag: flagged entities are added to the application's menu
	Prototype Flag: for flagged entities a prototype must be created

Entities can be added to the plan either using the 'Painter Tools' toolbar or using the 'Entities' menu.

Added entities can be selected, moved and re-sized so as to obtain a logically organized plan. Entities are selected by clicking them. You can drag and drop entities or re-size them working on the anchors, i.e. on the '*little squares*' displayed once entities are selected.

To access certain design functionalities you can also use keyboard commands.

Using **<Del>** selected entities are deleted. Using **<Enter>** you can open the definition dialog window of the selected entities or groups.

Using **Arrows Keys** you can move selected entities.

Using the **<Shift>** + **<Arrows Keys>** you can re-size selected entities or groups. The top left corner stays fix.

Using **<Ctrl>** + **<Shift>** + **<Arrows Keys>**'the selected entity or group can be re-sized. The bottom left corner stays fix.

You can scroll entities on the screen selecting the 'starting' entity and pressing a combination of the keys **<Tab>**, **<Shift>** and **<Ctrl>**.

The **<Tab>** key scrolls forwards. The keys **<Shift>**and **<Tab>** scroll backwards.

You can scroll groups forwards pressing **<Ctrl>** + **<Tab>** and backwards pressing **<Ctrl>** + **<Shift>** + **<Tab>**.

To scroll on dialog windows options use standard Windows functionalities.

To scroll tab-strips use **Arrows Keys**.

To edit options you can press **<Alt>** + **<underlined letter>**.

Selection lists can be sorted by column. You can further add or delete list items right clicking the mouse or using the **'Ins'** and **'Del'** keys.

Double clicking a 'Group' entity you edit the group contents.

Dragging and dropping entities (Master, Detail, Master/Detail, Dialog, Routine...) in 'Groups' all defined links are maintained.

Right clicking an entities a menu is opened that differs depending on the selected entity. These menus allow editing entity properties.

3.3 Design Painter Toolbars

Let us now analyze the Design Painter toolbars.

3.3.1 Clipboard Toolbar

The 'Clipboard' toolbar allows you to execute typical Window's commands such as Undo, Redo, Cut, Copy and Paste.

Picture 30 - Clipboard Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).
	The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).
	The 'Copy' option copies one or more selected elements.
	The 'Cut' option deletes selected elements.
	The 'Paste' option pastes copied or cutted elements in the selected position.

3.3.2 The Painter Tools toolbar

The Painter toolbar allows you to add entities to the current Design Plan. It has the same functionalities as the 'Entities' menu.

figura 2 - Painter Tools Toolbar

The following table shows you the toolbar buttons and their meanings.

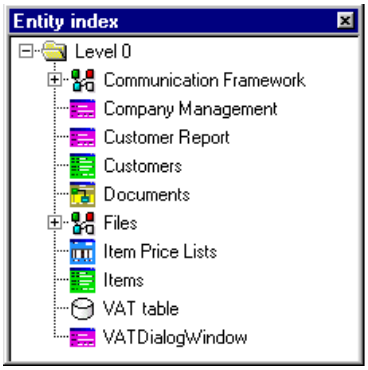
Button	Meaning
	Adds Master File entities. Used to manage e.g. customer files, item files etc.
	Adds Detail File entities. They manage records one to many. Used to manage e.g invoices.
	Adds Master/Detail entities. They manage records one to many. Used to manage e.g. invoices.
	Inputs Dialog Window entities. Used to manage selection or confirmation dialog windows. These entities are not linked to tables but can read any table available in the project.
	Inputs Routine entities. Used to write complete procedures or functions for specific data processing.
	Inputs Database Table entities. Used to support tables that do not have associated managing procedures.
	Inputs String entities. Used to add descriptive strings to the plan.
	Inputs Group entities. Used to organize the design in logical blocks. Within groups you can add other groups or entities that usually have a logical link.
	Inputs Output entities. Used to execute reports, queries and other data processing outputs.
	Inputs External Entities. Used to build references to entities which are external to the current project and belong to design plans of other modules or of other projects.

3.3.3 Entity Index Toolbar

The 'Entity Index' toolbar displays a tree-view of the Design plan elements. It displays elements' names and positions in the design hierarchy.

Hierarchical levels and the treeview order are determined by Groups, i.e. their sequential numbers and position in the plan. Each group is displayed as a directory. Using '+' and '-' you can open or close levels.

Picture 31 - The Entity Index Toolbar

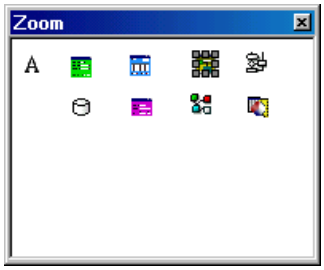


Entities can be opened double clicking the desired entity directly from the 'Entity Index' window.

3.3.4 Zoom Toolbar

The 'Zoom' toolbar gives you an overview of entities in the selected group. It displays their position in the plan but does not display links.

figura 3 - Zoom Toolbar



Using this toolbar you can work directly on the entities. You can select, move and re-size entities. Changes are made also in the design plan. You can also copy and paste elements in the 'Zoom' window. Pasted elements must be positioned in the design plan. You can open entities double clicking them in the 'Zoom' window.

3.3.5 File Toolbar

The 'File' toolbar has a set of buttons to help you interacting with the tool. The 'File' toolbar has the same functionalities as the 'File' menu.

figura 4 - File Toolbar

The following table shows you the toolbar icons and their meanings:

Button	Meaning
	Creates new Design Plans.
	Opens existing Design Plans.
	Saves definition files of the current element.
	Saves definition files with different names.
	Saves definition files and runs their generation ('Design Generation').

3.3.6 Pages Toolbar

The 'Pages' toolbar helps you browsing between the different plan levels. The 'Pages' toolbar has the same functionalities as the 'Level' menu.

Picture 32 - The Pages Toolbar

The following table displays the tollbar buttons and their meanings.

Button	Meaning
	Goes down a level when a Group is selected.
	Goes up a level taking you out of the Group.
	Brings you back on the design plan.

3.4 Main Menu

3.4.1 File

Picture 33 - File Menu

The 'File' menu has a set of options to:

- Create new design plans.
- Load definition files for existing design plans.
- Save changes to the current design plan.
- Save design plans with different names.
- Save changes of the current design plan and generate the design.
- Read information on CODE**PAINTER REVOLUTION**.
- Exit the tool.

N.B.
The design definition file is stored in two ASCII files ***'<DesignName>.designed'*** *and* ***'<DesignName>.designdef.shelve'****. When the design plan is opened or saved you open and save the two files contemporary.*

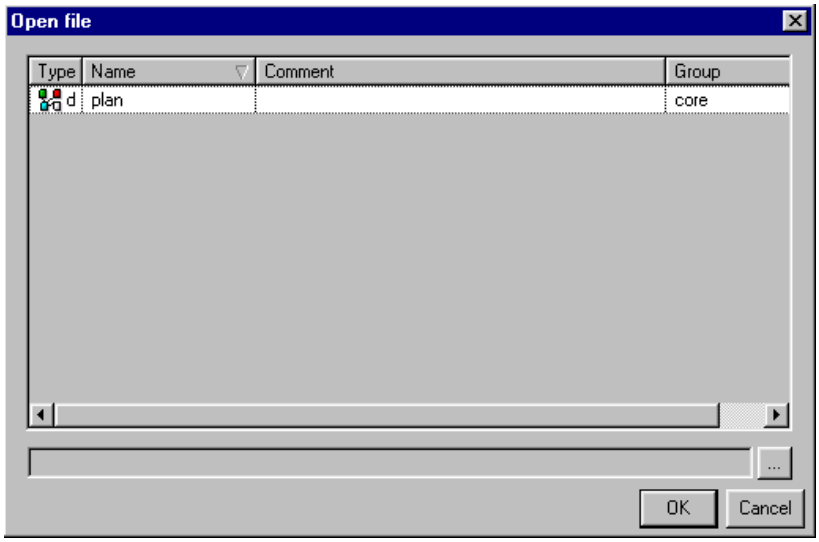
New

The 'New' option closes the current design plan asking whether you want to save the changes or not and opens a new design plan.

Open...

The 'Open' option loads definition files belonging to the current project.

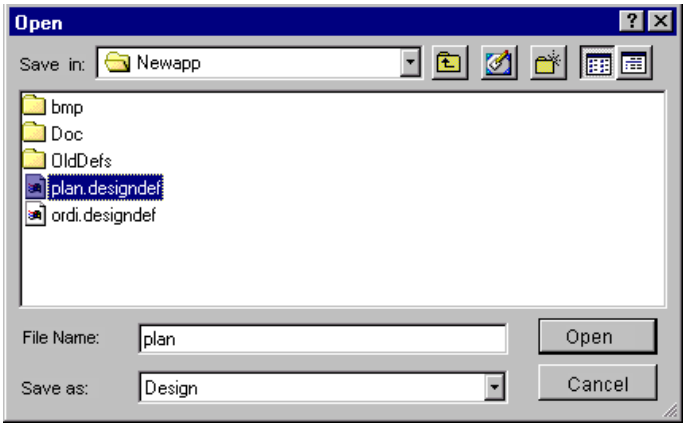
Picture 34 - Open File



The dialog window lists all entities in the project. You can order files by Type, Name, Comment or Group clicking the corresponding column. You can identify the sorting column by the arrow next to the column description.

Picture 35 - Sort Columns

Clicking the '...' button you can browse to search design plans in other project modules.



Picture 36 - 'Open' Dialog Window To Select Design Plans

Save

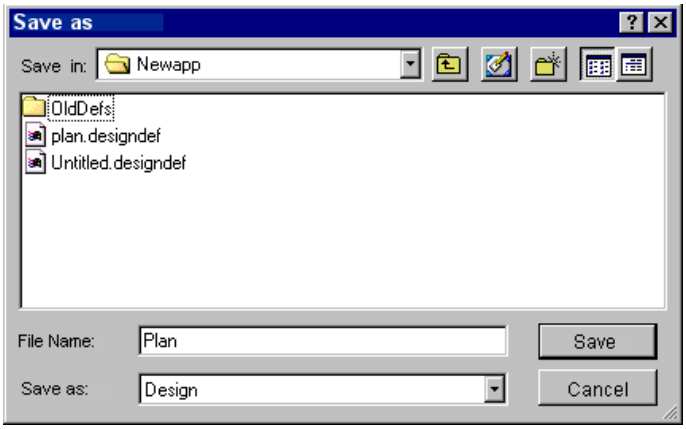
The 'Save' option saves definition files in use.

When you save the design plan back-up files (.BAK) are created. These store the plan without including the latest changes.

Save As...

The 'Save As' option saves the design plan with a different name.

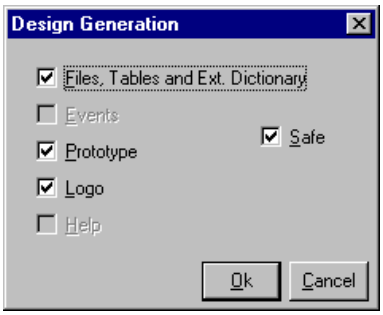
Picture 37 - Save As Dialog Window



Save And Generate

The 'Save And Generate' option saves the current design plan and runs the generation. The same is done opening the 'Generation' menu and selecting the 'Design'.

Picture 38 - Design Generation



About

The 'About' option displays information on the product.

For more information please refer to section 'Product Information'.

Exit

The 'Exit' option makes you exit the tool asking you whether you want to save the changes or not.

3.4.2 Edit

Picture 39 - Edit Menu

The 'Edit' menu has a set of options to:

Undo/repeat commands.

Delete, copy and add elements.

Search, display, and replace elements.

Hide or show toolbars.

Undo

The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).

Redo

The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).

Copy

The 'Copy' option copies elements. More elements can be either selected or grouped in a selection frame that you can draw with the mouse.

Cut

The 'Cut' option deletes selected elements from the plan.

Paste

The 'Paste' option pastes copied or cutted elements in the selected position.

Find...

The 'Find' option opens the Find and Replace dialog window to find and/or replace elements.

For more information please refer to 'Edit Menu Advanced Options'.

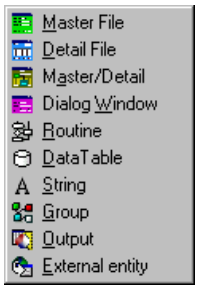
Toolbars

The Toolbars option displays a submenu to enable or disenable toolbars on the Design Painter.

Picture 40 - Toolbar Menu

3.4.3 Entities Menu

The 'Entities' menu has a set of options to add entities to the Design plan.



Picture 41 - Entities Menu

For more information on entities please refer to the 'Entities Definition' section.

Master

Master File entities are made of a database table and a procedure allowing to input, modify and cancel data stored in the database. 'Master Files' are files without repeated fields. They are used to manage files in which records are computed one by one. For instance you can use Master File entities to manage customer files. Indeed in this case information related to individual customers is recorded in single records. When you change details on a given customer the relevant record is updated.

Detail

In business/commercial applications you are often required to manage more complex data structures, where fields can be repeated. This may be the case when you are required to manage invoices. One invoice can be saved as a single record but the invoice *Body* needs to contain many rows in which the same fields are repeated for the number of itemsthat the order contains.

Detail entities manage single tables. Records having the same primary key are processed as single documents.

This kind of structure allows to manage a single instance of an entity (e.g invoice no.1) that is physically joined with 'N' table records. 'N' records are the detail of the items listed in invoice no.1.

Detail entities typically have a '*Header*' containing the primary key and some general data; a '*Body*' with its unlimited repeated fields that make up the rows containing the document's detail; and a '*Footer*' containing other general data and/or document's totals. Consequently when you define a Detail entity you need to determine which fields belong to the header, which to the body and which to the footer. Repeated fields in the body must be flagged as '*Repeated*'.

In client/server environments Detail entities must have a composite primary key to clearly identify and distinguish row values. The primary key must include one or more header fields and one or more body fields. Often you may not be able to identify appropriate candidates within the Body. This is why CodePainter automatically defines the numeric field '**CPROWNUM**' to number the body rows. For example in an invoice header you may select the fields 'document date' and 'number' as part of the primary key. To make the body rows unique you have two options: either you define the 'Item Number' as part of the primary key or you select the field CPROWNUM. If you select the 'Item Number' each item number cannot be repeated more than once on the same document/ invoice. If you select the repeated field CPROWNUM as part of the primary key the field will automatically take on the next number for each row that is entered.

The generated procedures that manage these entities deal with the 'N' table records in one document only. Data contained in Header and Footer is displayed in the main document. The Body displays the repeated fields in a grid. Rows can be scrolled up or down.

Master/Detail

The Master/Detail entity is similar to the Detail entity in terms of display, definition and use. The main difference is that the Master/Detail entity is made of two entities, a **Master File** and a **Detail File** entity.

Master/Detail entities manage two physical tables. The first (Master) contains Header and Footer data. The second (Detail) contains the Master Key and repeated rows of information. The two tables are linked through the primary key.

Fields are defined the same way as in 'Detail' entities (Header, Body and Footer fields). Again, repeated fields in the body must be flagged as '*Repeated*'. In the Database tab-strip you need to define the 'Data Name' and the 'Physical Name' for both, the detail and the master table.

Tables for this entity are created as follows: the master table contains header and footer fields. The detail table contains the primary key of the master table and all repeated fields.

In client/server environments the Detail entity must have a composite primary key to clearly identify and distinguish row values. The primary key must include one or more header fields and one or more body fields. Often you may not be able to identify appropriate candidates within the Body. This is why CodePainter automatically defines the numeric field '**CPROWNUM**' to number the body rows. For example in an invoice header you may select the fields 'document date' and 'number' as part of the primary key. To make the body rows unique you have two options: either you define the 'Item Number' as part of the primary key or you select the field CPROWNUM. If you select the 'Item Number' each item number cannot be repeated more than once on the same document/ invoice. If you select the repeated field CPROWNUM as part of the primary key the field will automatically take on the next number for each row that is entered.

Despite the fact that the entity is made of two linked tables, the generated dialog window is displayed as a single document. Data contained in Header and Footer is displayed in the main document. The Body displays the repeated fields in a grid. Rows can be scrolled up or down.

Choosing Detail File entities rather than Master/Detail entities is bound to the kind of problem you are required to manage.

Working with Detail Files there is the advantage that you use a single table only. On the other hand if you have many 'general data' fields in the header and footer the Detail table will contain redundancy data. Indeed general data would be repeated as often as the number of rows in the document. Master/Detail entities store 'general data' only once, thus avoiding unnecessary redundancies.

When the header must contain a few fields only creating a Master Table would be an unnecessary waste of memory space. In this latter case you should use Detail File entities.

Dialog Window

Dialog Window entities typically contain variables for selection parameters, buttons, integrated zooms, etc.

Dialog Windows do not manage tables. Indeed when you define a Dialog Window entity you will notice that there is no 'Database' tab-strip. In the 'Main' tab-strip you need to define the 'Entity' and 'Program' names and select a template. You can also set the flags to generate the prototype and to include the dialog window in the application's menu.

Dialog Window prototypes contain only few information. Using the Dialog Window Painter you will be able to add variables, fields, buttons, etc.

Routine

Routine entities are used to run routine procedures or to process specific functions on files.

Using Routine entities you can update a due register at the end of each month basing on payment files; or to run a procedure that sets to zero the end of year balances.

You should think of routines as black boxes in which data is input, processed and output.

In the Routine 'Main' tab-strip you need to define the 'Entity' and 'Program' names and select a template. You can also set the flags to generate the prototype and to include the routine in the application's menu.

Routine prototypes contain only few information. Using the 'Routine' Painter you will be able to define the data flow and the commands that must be executed.

Data Table Entity

Often different business/commercial applications have the same problems that cannot be managed through input/output dialog windows, but require support tables.

For instance you may need to import the Customers file from an old procedure using a support file.

CODE**PAINTER REVOLUTION** uses the **Data Table Entity** class, that allows you to input data required to define tables (structure, logical and physically name, keys and indexes) without creating managing procedures. Indeed the 'Main' definition window of Data Table entities does not linked procedure and template.

In client/server applications this entity has also the **External Table** flag that allows you to specify the entity as external to the database. This means that you can link to tables, which are on different servers. External tables are not maintained by the current application.

String

Strings allow you to add comment strings to the work sheet. This may be useful to better explain the project and/or to write same general notes.

Group

Groups are logical gatherings of objects.

Groups are used to compact the design plan spreading it on more levels and make it more understandable.

As your project grows you may find it helpful to logically group entities.

Once the group is created you can edit it either double clicking it or right clicking it and selecting the 'Edit' option.

In the definition window you can change the group's **Name** and enable or disenable the **Menu** flag to include or exclude it from the application's menu. This allows you to create a number of application's submenus easily.

The **Links** page displays the list of *visibility* links affecting the group.

To move existing entities into a group, right click the desired entity, select the 'Group to' option and select the name of the group.

Output

This entity allows you to define references to objects, e.g. reports, graphs, queries, etc. Outputs are defined at run-time using the application's Visual Tools.

External Entity

This entity allows to add entities belonging to other projects in your current project.

3.4.4 Level

The 'Level' menu helps you browsing in the design plan.

Picture 42 - Level Menu

Top Level

The 'Top Level' option returns to the first level of the design plan. When you are in a Group you can come back to the design plan selecting this option.

Down

The 'Down' option allows you to go down a level. This option works only for selected groups and shows you the groups' contents.

Up

The 'Up' option exits groups and goes up a level in the Design hierarchy (tree).

3.4.5 Global

In the 'Global' menu you can define information on the application that will be displayed in the generated application.

For more information please refer to 'Global Menu Advanced Options'.

Picture 43 - Global Menu

Definitions...

In the 'Definitions' option you can define the application's title, subtitle, author and version. You can also see the creation date and the last revision date.

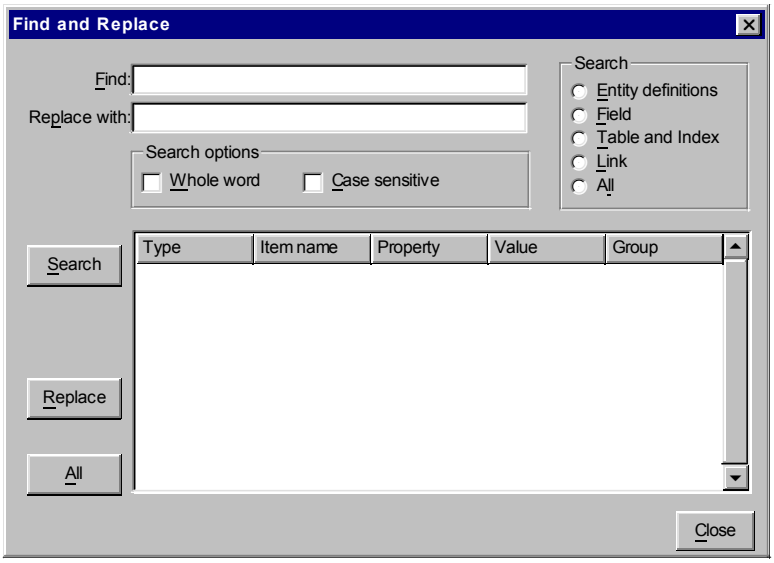
3.5 Edit Menu Advanced Options

The Design Painter has also the 'Find and Replace' functionality, which can be accessed opening the 'Edit' menu and selecting the 'Find' option.

3.5.1 Find And Replace

Find and replace dialog window.

Picture 44 - Find And Replace: Page 1



Search

String that must be searched.

Replace

String that must replace the searched string.

Options

Entity definitions

Searches only amongst entity names.

Table and Index

Searches only amongst tables and indexes.

Link

Searches amongst links.

All

Searches anywhere.

Field

Searches amongst fields.

Whole Word

When this field is flagged the search activity must be performed on whole words only. When this flag is not flagged search results may include substrings of long words.

Case sensitive

The search activity matches upper and lower Case.

Search

Starts the search activity.

Occurrences

Search result listing elements found.

Replace

Replaces strings in all selected elements (occurences), found using the string defined in the 'Replace with' textbox.

All

Replaces strings in all elements (occurences), found using the string defined in the 'Replace with' textbox no matter if elements are selected or not.

3.6 Global Menu Advanced Option

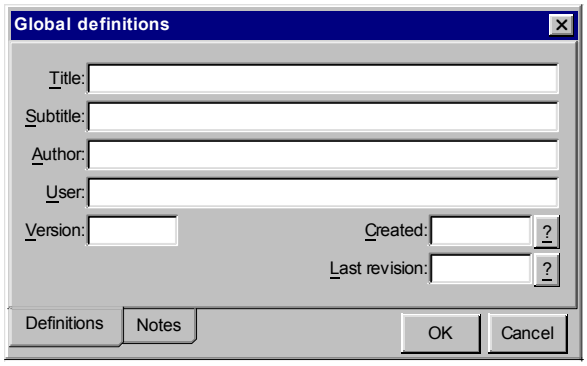
3.6.1 Global Definitions

Descriptive information documenting the design plan.

This information becomes part of the automatically generated documentation.

Definitions Page

Picture 45 - Global Definitions: Page 1



Title

Brief description of the design plan.

This information is used to create the application's logo and the documentation's heading.

Subtitle

Further application's description.

Author

Author of the design plan.

User

Customer for whom the application is developed.

Version

Application version.

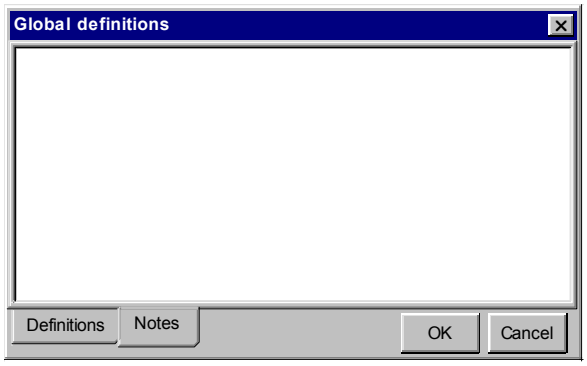
Created

Creation date of the Design plan.

Revised

Last time the design plan has been revised.

Notes Page



Picture 46 - Global Definitions: Page 2

Notes

General notes on the design plan.

These notes will be included in the technical documentation. They will be added straight after the title before the notes on the various entities.

3.7 Item Edit Menu Options

3.7.1 Item Edit Menu

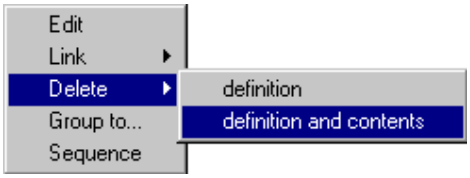
To edit defined entities right click an entity and select the 'Edit' option from the opened menu.

Picture 47 - Edit Menu:the prototyped entities

When no prototype has been generated for a given entity the 'Open Codify' option is not available in the right click menu, because no interface has been created for that entity. You can either prototype the selected entity or you can link the entity to another entity for which the prototype exists.

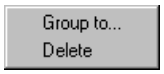
Picture 48 - Edit Menu: Not Prototyped Entities

The 'Open Codify' option is also disenabled for 'Groups'.



Picture 49 - Menu Edit: Group Of Entities

You can select multiple entities. Right clicking the selection you can either delete (Delete) the selection or include the selection into a Group (Group To).



Picture 50 - Menu Edit: Multiple Entities Selection

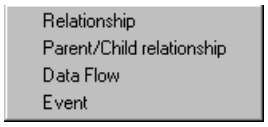
Edit

The Edit option opens the Entity Definition window for the selected entity.

Link...

The Link option opens a submenu for the link definition. Links available depend on the type of entity selected. When you select one of the available link types the mouse changes in '+LINK'. To establish the link you need to select the target entity.

Linking Entities



Picture 51 - Link Menu

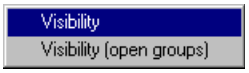
Entities can be linked through:

- Relationship Links.
- Parent/Child Relationship Links.
- Data Flow Links (this kind if link can be defined only from and to routine entities).
- Event Links.

Linking Groups

You can link two groups using the 'Visibility' option or link a group to an entity in another group using the 'Visibility Open Groups' option.

Picture 52 - Link Group Menu



Delete

The 'Delete' option cancels the selected entity. To delete groups you have two options: either you delete the group definition using the 'Definition' option or you delete the group and its contents using the 'Definition and Contents' option.

Picture 53 - Delete Group Menu

Group To...

The 'Group to' option adds entities in the selected group.

Open Codify...

The 'Open Codify' option opens the selected entity using the corresponding Painter.

Sequence...

The 'Sequence' option opens the sequential number dialog window and shows the list of sequential numbers defined in the entities.

3.8 Link Edit Menu Options

3.8.1 Link Edit Menu

Right clicking the line identifying the link between two entities you access a menu that allows editing the relationship definition dialog window.

For more information please refer to section 'Defining Links'.

Depending on the kind of entities linked the menu differs.

Select **Relationship Link**:

Picture 54 - Relationship Link Menu

Select **Parent/Child Link**:

Picture 55 - Parent/Child Link Menu

Select **Data Flow Link**:

Picture 56 - DataFlow Link Menu

Select **Event Link**:

Picture 57 - Event Link Menu

The submenu opened in the 'Link Edit' menu displays two items. One is used to edit and the other to delete links.

Picture 58 - An example of 'Link SubMenu'

3.9 Entities Definition

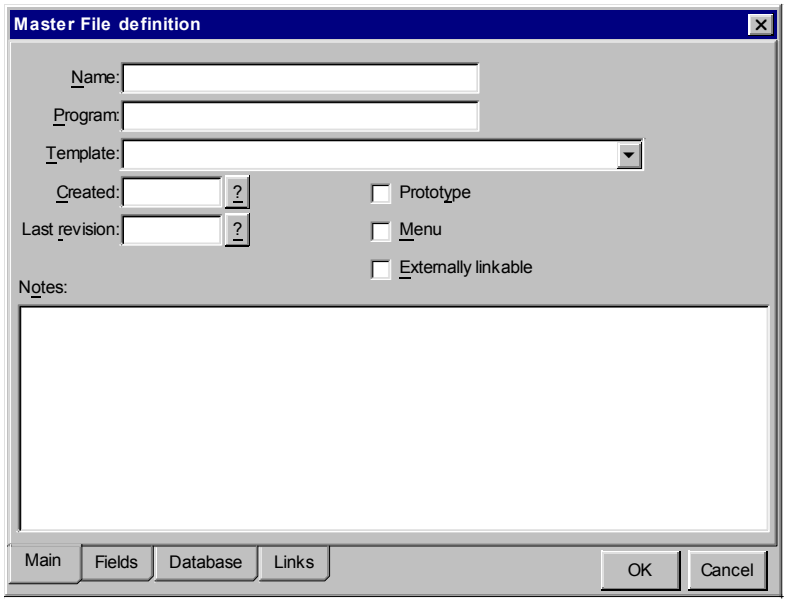
This section analyzes the design options of each entity in detail.

3.9.1 Master File Definition

This dialog window is used during the Design phase to define properties of 'Master', 'Master/Detail', or 'Detail' entities.

Main Page

Main Page. Defines general information on the entity added to the design plan.



Picture 59 - Master File Definition: Page 1

Name

Entity title.

The entity title is a brief description of the entity. Each entity has a unique name, which identifies it. The description helps identifying the entity's functionality without the need to follow naming rules.

The description is used by the menu prototype to create the toolbar from which the entity can be executed and by the prototype to name the created window.

Program

Name identifying the entity.

The program name must follow file naming rules. The description (previous field) must not follow any rule.

Each entity has a set of associated CodePainter repository files: codify, program and documentation files. The program name is the basis for all these files.

The Prototype tool creates for each entity a file describing the entity's characteristics. The file is used in the Codify phase. It has the name defined in the 'Program' field. The extension changes according to the kind of entity (MstDef for 'Master file', DtlDef for 'Detail file', etc.).

For each entity one or more files containing classes and procedures are defined. Whether one or more files are created depends on the programming language and the template selected. The main file will always have this name. The extension depends on the programming language.

The documentation creates more files: one for 'User' and one for 'Technical' documentation including various files used to display the documentation on the screen.

Template

Templates are procedure skeletons, which are combined with the definitions contained in the repository, so that the source program can be produced. This process is called automatic code generation template driven. Changing templates you can generate the documentation, or the source code, or change the programming language in which the program is re-generated.

Using this option you can define the template that must be used during the process.

Each entity class can have more templates. Different source codes will be generated starting from the same definition scheme stored in the repository. For example for each class a 'Parent' and a 'Child' table are generated.'Child'. 'Parent' entities are independent procedures that can be opened in the application's menu. 'Child' procedures must receive the primary key from the 'Parent'. 'Child' procedures are therefore integrated in the Parent either directly or in form of buttons from which the Child procedure can be launched. The generated code changes according to the selected template.

The combobox details all tables available for the defined entity class.

Created

Date in which the entity was created.

This date is defaulted by the system when the entity is created.

Revised

Date in which the last changes have been made.

This date is automatically up-dated when the entity is changed. This date combined with the 'Created' date gives you an idea of the time spent on defining the entity.

Entities having an old 'Revised' date are typically stable entities, where no new implementations are required. Entities having 'Created' and 'Revised' dates, which are close to each other are typically entities that must be simply managed. Entities with an old 'Created' date and a recent 'Revised' date are typically complex entities or key processes in the program's architecture. Entities having a recent 'Revised' date are typically entities under development.

Prototype

Defines whether a prototype must be created or not.

Once you defined the entity in the design plan you can create a prototype for that entity. Prototypes are fully functional programs but are incomplete in terms of calculations and validations. Further the layout usually requires improvements.

Prototypes are typically created in order to check together with customer/ user whether specifications are detailed enough to go on to the Codify phase (JAD: joined application development).

When this flag is set and the prototype generation executed, a new prototype is created that replaces the exiting one. When the flag is deselected no prototype is created for that entity.This flag is usually set until you need to work on the design plan. As soon as you start working with the Codify tools you should deselect the files. In order to integrate changes made in the design plan with a dialog window that must no longer be prototyped you can use the 'Design compare' option in the 'Edit' menu in the Codify tools. You will so identify missing fields, links, etc.

Menu

Defines whether the entity must be added to the application's menu.

For each entity a dialog window is created that allows inputting and changing data. When this flag is set the application's menu will contain the entity's name from where the associated procedure can be launched.

Externally linkable

Defines whether the entity can be seen from a different design plan.

For large applications you can create more design levels. Designs on other levels belonging to the same project can read entities in other design plans, which have this flag set.

Once the entity is exported and before you start making big changes you should verify that these changes don't have major impacts on the definitions contained in the other design levels. Within the current design plan CodePainter can automatically check (and realign if required) if a field has been used in a link and the linked field is not compatible.

Keep this simple rule in mind: in exported entities you should never delete fields or change field types.

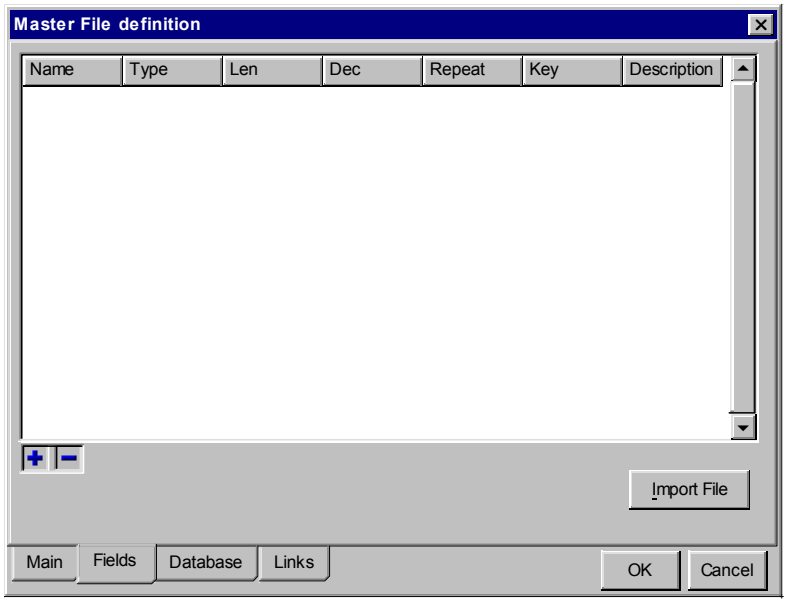
Notes

Notes on the entity.

Notes should include project requirements for each entity. These notes will be integrated in the technical documentation. You can refer to these notes at any point in time during the Codify phase.

Fields Page

Lists all fields in the entity.



Picture 60 - Master File Definition: Page 2

Fields

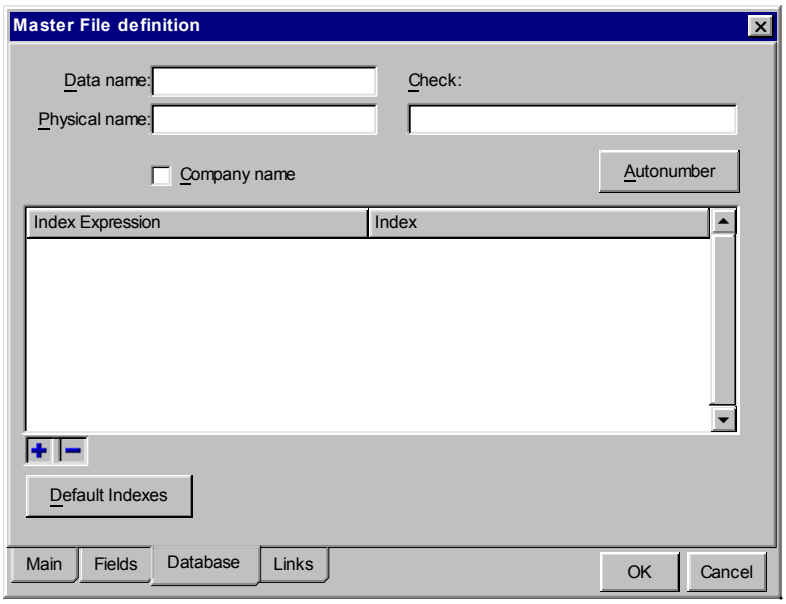
List of fields contained in the entity.

Import File

Allows importing existing xBASE or ODBC database structures.

Database Page

Picture 61 - Master File Definition: Page 3



Data name

Symbolic name of the table associated to the entity.

This name is the reference alias for the relational database table associated to the entity. The physical name of the table can be different. When the multicompany system is activated for this table there can be more than one physical table associated to the symbolic name.

Physical name

Physical name of the table associated to the entity.

In relational databases tables are associated to entities. When the multicompany system is activated a table for each defined company is associated to one entity. The table used is the one of the current company.

The physical name allows giving the table a name which is different from the symbolic one. In multicompany environments the company name/ code becomes part of the physical name. The physical name must contain the string 'xxx', which will be replaced by the current company name/ code. This applies to any sentence that accesses the database.

Check

Check sentences on the table.

Here you can define a SQL sentence, which will be applied before any data is input in the database. When the sentence gives FALSE the input is refused.

mastername

Master Table symbolic name.

This name is defaulted adding the suffix "_m" to the detail table's symbolic name.

masterphname

Master table physical name.

This name is defaulted adding the suffix "_m" to the detail table's physical name. The same multicompany options explained for the 'detail' table apply.

mastercheck

Check sentence on the master table.

Here you can define a SQL sentence, which will be applied before any data is input in the database. When the sentence gives FALSE the input is refused.

Company name

Shows that the table is mutlicompany.

When the flag is set the mutlicompany system for the table associated to this entity is activated. To one symbolic name more physical tables are associated. The table used is the one linked to the current company. The physical name must contain the string 'xxx'. The string will be replaced by the name of the current company (i_CodAzi).

Autonumber

Accesses the table that defines progressive numbers.

Index expression

List of indexes.

This list details the indexes that are created for the table. The first index describes the primary key. The other indexes describe the other keys created in order to increase the speed of search and sort activities.

In relational databases each SQL sentence is analyzed and the quickest access plan iscreated (optimization). When a table has indexes these are used in order to optimize response times. This is why you should always create a set of indexes to easesearch path, e.g defining the code as well as the description as keys.

Creating indexes made of too many fields does not necessarily help the optimizer to find the quickest path. Indexes made of many elements are used only when the SQL sentence includes all fields (making the index) in the selection criteria. A good rule is to create indexes made of maximum three fields. For example you have an index made of NUMDOC (document number) and by DATDOC (document date). The SQL sentence in which the WHERE clause includes both fields can use the entire index. When the clause includes NUMDOC only it used half of the index. When the clause includes DATDOC no index is used, because DATDOC comes only after NUMDOC.

When queries mainly start from the document date your index should be made of DATDOC first and followed by NUMDOC.

When queries are mainly made on DATDOC as well as on NUMDOC you should create two indexes so that queries are always optimized.

When you want to search data by date and order it by document number you should create two indexes. The first including the fields DATDOC and NUMDOC. The second including NUMDOC only. The first index will be used when date and number, or date only are given, or when want to order data by date and number. The second index is used when the only the number is given.

Good indexes must be selective, i.e. basing on the value input for the query they must lead to results that have few records. Queries can otherwise not be optimized. For example you have the character field CUSTSEX length 1 containing M for men, W for woman and a blank for companies. You have 9000 records. Following statistics no matter which value you input you will always get one third of the records, i.e. 3000 records. You also have the 'Company' field length 30. Typically each record will contain a different value. An index on this field would probably always give you the result of 1. Generally speaking we can say that the first index does not help you much, whereas the second gives you the exact record you are searching for.

Indexes are a list of fields divided by commas.

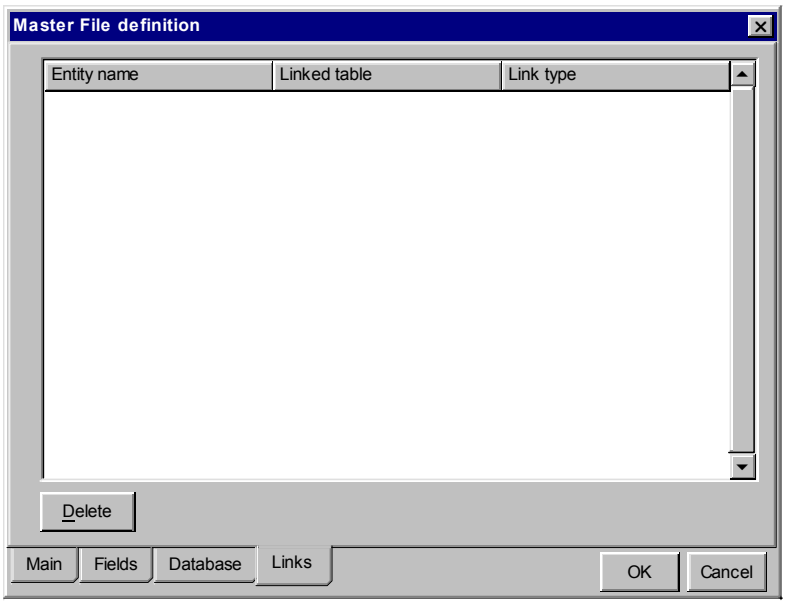
Default Indexes

Default indexes.

This button creates default expressions for indexes basing on keys/indexes defined for fields.

Links Page

Picture 62 - Master File Definition: Page 4



Links

List the links of the current entity.

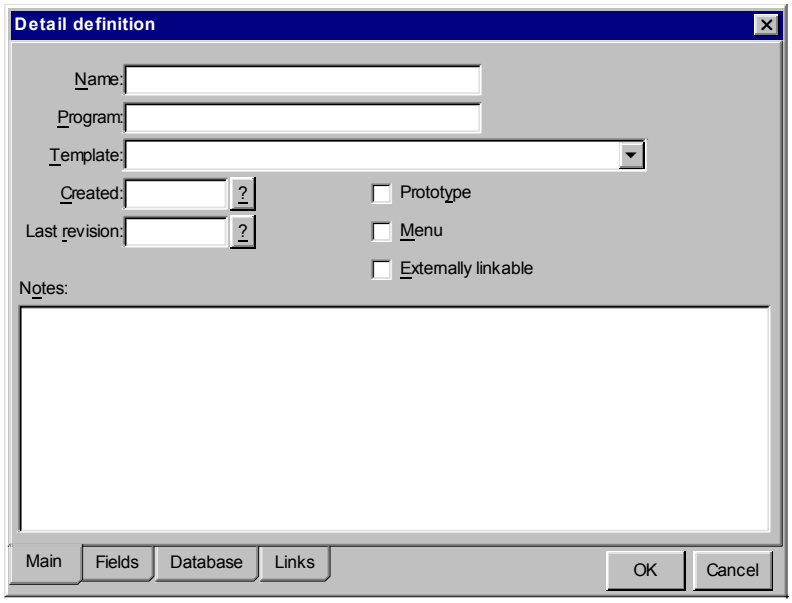
Delete

Deletes selected links.

3.9.2 Detail definition

Main Page

Main Page. Defines general information on the entity added to the design plan.



Picture 63 - Detail definition: Page 1

Name

Entity title.

The entity title is a brief description of the entity. Each entity has a unique name, which identifies it. The description helps identifying the entity's functionality without the need to follow naming rules.

The description is used by the menu prototype to create the toolbar from which the entity can be executed and by the prototype to name the created window.

Program

Name identifying the entity.

The program name must follow file naming rules. The description (previous field) must not follow any rule.

Each entity has a set of associated CodePainter repository files: codify, program and documentation files. The program name is the basis for all these files.

The Prototype tool creates for each entity a file describing the entity's characteristics. The file is used in the Codify phase. It has the name defined in the 'Program' field. The extension changes according to the kind of entity (MstDef for 'Master file', DtlDef for 'Detail file', etc.).

For each entity one or more files containing classes and procedures are defined. Whether one or more files are created depends on the programming language and the template selected. The main file will always have this name. The extension depends on the programming language.

The documentation creates more files: one for 'User' and one for "Technical' documentation including various files used to display the documentation on the screen.

Template

Templates are procedure skeletons, which are combined with the definitions contained in the repository, so that the source program can be produced. This process is called automatic code generation template driven. Changing templates you can generate the documentation, or the source code, or change the programming language in which the program is re-generated.

Using this option you can define the template that must be used during the process.

Each entity class can have more templates. Different source codes will be generated starting from the same definition scheme stored in the repository. For example for each class a 'Parent' and a 'Child' table are generated.'Child'. 'Parent' entities are independent procedures that can be opened in the application's menu. 'Child' procedures must receive the primary key from the 'Parent'. 'Child' procedures are therefore integrated in the Parent either directly or in form of buttons from which the Child procedure can be launched. The generated code changes according to the selected template.

The combobox details all tables available for the defined entity class.

Created

Date in which the entity was created.

This date is defaulted by the system when the entity is created.

Revised

Date in which the last changes have been made.

This date is automatically up-dated when the entity is changed. This date combined with the 'Created' date gives you an idea of the time spent on defining the entity.

Entities having an old 'Revised' date are typically stable entities, where no new implementations are required. Entities having 'Created' and 'Revised' dates, which are close to each other are typically entities that must be simply managed. Entities with an old 'Created' date and a recent 'Revised' date are typically complex entities or key processes in the program's architecture. Entities having a recent 'Revised' date are typically entities under development.

Prototype

Defines whether a prototype must be created or not.Defines whether a prototype must be created or not.

Once you defined the entity in the design plan you can create a prototype for that entity. Prototypes are fully functional programs but are incomplete in terms of calculations and validations. Further the layout usually requires improvements.

Prototypes are typically created in order to check together with customer/ user whether specifications are detailed enough to go on to the Codify phase (JAD: joined application development).

When this flag is set and the prototype generation executed, a new prototype is created that replaces the exiting one. When the flag is deselected no prototype is created for that entity.This flag is usually set until you need to work on the design plan. As soon as you start working with the Codify tools you should deselect the files. In order to integrate changes made in the design plan with a dialog window that must no longer be prototyped you can use the 'Design compare' option in the 'Edit' menu in the Codify tools. You will so identify missing fields, links, etc.

Menu

Defines whether the entity must be added to the application's menu.

For each entity a dialog window is created that allows inputting and changing data. When this flag is set the application's menu will contain the entity's name from where the associated procedure can be launched.

Externally linkable

Defines whether the entity can be seen from a different design plan.

For large applications you can create more design levels. Designs on other levels belonging to the same project can read entities in other design plans, which have this flag set.

Once the entity is exported and before you start making big changes you should verify that these changes don't have major impacts on the definitions contained in the other design levels. Within the current design plan CodePainter can automatically check (and realign if required) if a field has been used in a link and the linked field is not compatible.

Keep this simple rule in mind: in exported entities you should never delete fields or change field types.

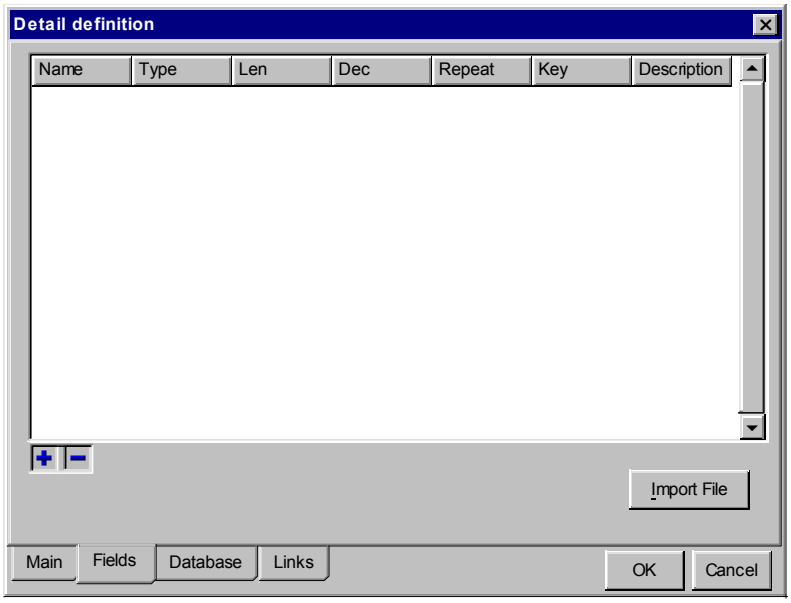
Notes

Notes on the entity.

Notes should include project requirements for each entity. These notes will be integrated in the technical documentation. You can refer to these notes at any point in time during the Codify phase.

Fields Page

Lists all fields in the entity.



Picture 64 - Detail definition: Page 2

Fields

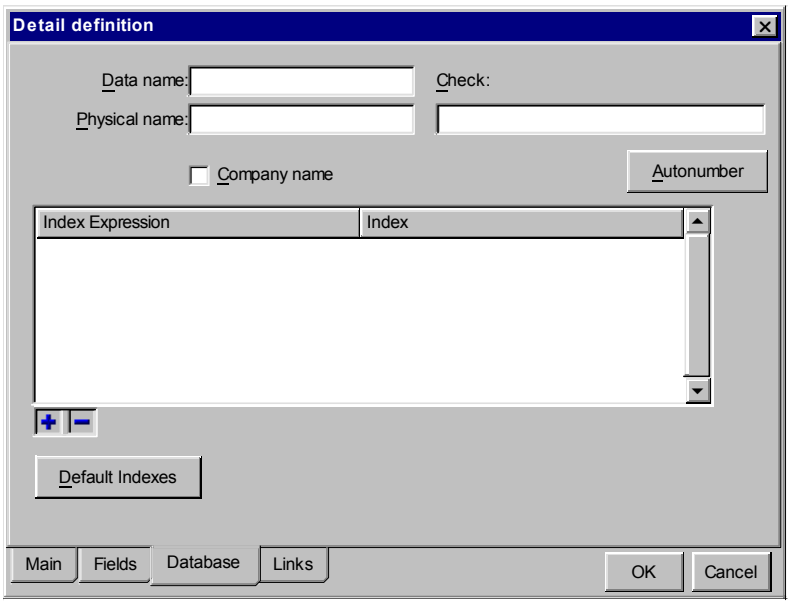
List of fields contained in the entity.

Import File

Allows importing existing xBASE or ODBC database structures.

Database Page

Picture 65 - Detail definition: Page 3



Data name

Symbolic name of the table associated to the entity.

This name is the reference alias for the relational database table associated to the entity. The physical name of the table can be different. When the multicompany system is activated for this table there can be more than one physical table associated to the symbolic name.

Physical name

Physical name of the table associated to the entity.

Check

Check sentences on the table.

Here you can define a SQL sentence, which will be applied before any data is input in the database. When the sentence gives FALSE the input is refused.

mastername

Master Table symbolic name.

This name is defaulted adding the suffix "_m" to the detail table's symbolic name.

masterphname

Master table physical name.

This name is defaulted adding the suffix "_m" to the detail table's physical name. The same multicompany options explained for the 'detail' table apply.

mastercheck

Check sentence on the master table.

Here you can define a SQL sentence, which will be applied before any data is input in the database. When the sentence gives FALSE the input is refused.

Company name

Shows that the table is mutlicompany.

When the flag is set the mutlicompany system for the table associated to this entity is activated. To one symbolic name more physical tables are associated. The table used is the one linked to the current company. The physical name must contain the string 'xxx'. The string will be replaced by the name of the current company (i_CodAzi).

Autonumber

Accesses the table that defines progressive numbers.

Index expression

List of indexes.

This list details the indexes that are created for the table. The first index describes the primary key. The other indexes describe the other keys created in order to increase the speed of search and sort activities.

In relational databases each SQL sentence is analyzed and the quickest access plan iscreated (optimization). When a table has indexes these are used in order to optimize response times. This is why you should always create a set of indexes to easesearch path, e.g defining the code as well as the description as keys.

Creating indexes made of too many fields does not necessarily help the optimizer to find the quickest path. Indexes made of many elements are used only when the SQL sentence includes all fields (making the index) in the selection criteria. A good rule is to create indexes made of maximum three fields. For example you have an index made of NUMDOC (document number) and by DATDOC (document date). The SQL sentence in which the WHERE clause includes both fields can use the entire index. When the clause includes NUMDOC only it used half of the index. When the clause includes DATDOC no index is used, because DATDOC comes only after NUMDOC.

When queries mainly start from the document date your index should be made of DATDOC first and followed by NUMDOC.

When queries are mainly made on DATDOC as well as on NUMDOC you should create two indexes so that queries are always optimized.

When you want to search data by date and order it by document number you should create two indexes. The first including the fields DATDOC and NUMDOC. The second including NUMDOC only. The first index will be used when date and number, or date only are given, or when want to order data by date and number. The second index is used when the only the number is given.

Good indexes must be selective, i.e. basing on the value input for the query they must lead to results that have few records. Queries can otherwise not be optimized. For example you have the character field CUSTSEX length 1 containing M for men, W for woman and a blank for companies. You have 9000 records. Following statistics no matter which value you input you will always get one third of the records, i.e. 3000 records. You also have the 'Company' field length 30. Typically each record will contain a different value. An index on this field would probably always give you the result of 1. Generally speaking we can say that the first index does not help you much, whereas the second gives you the exact record you are searching for.

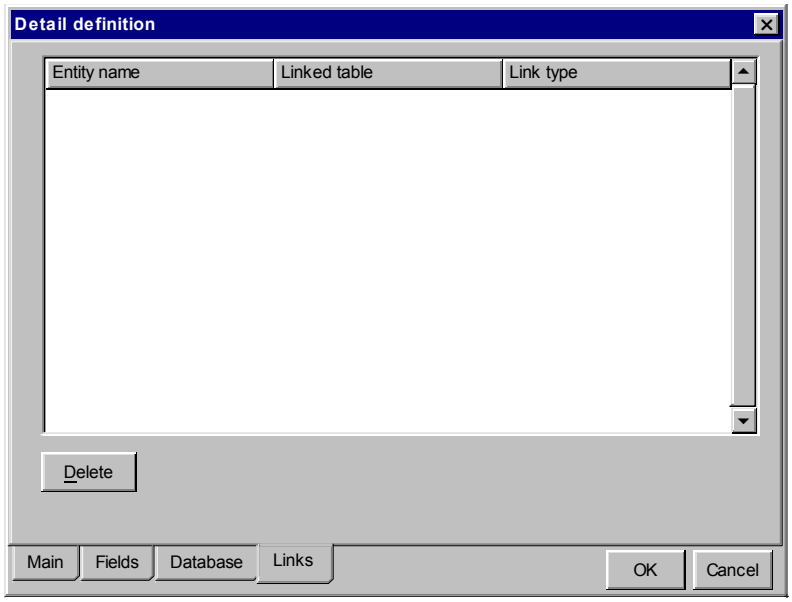
Indexes are a list of fields divided by commas.

Default Indexes

Default indexes.

This button creates default expressions for indexes basing on keys/indexes defined for fields.

Links Page



Picture 66 - Detail definition: Page 4

Links

List the links of the current entity.

Delete

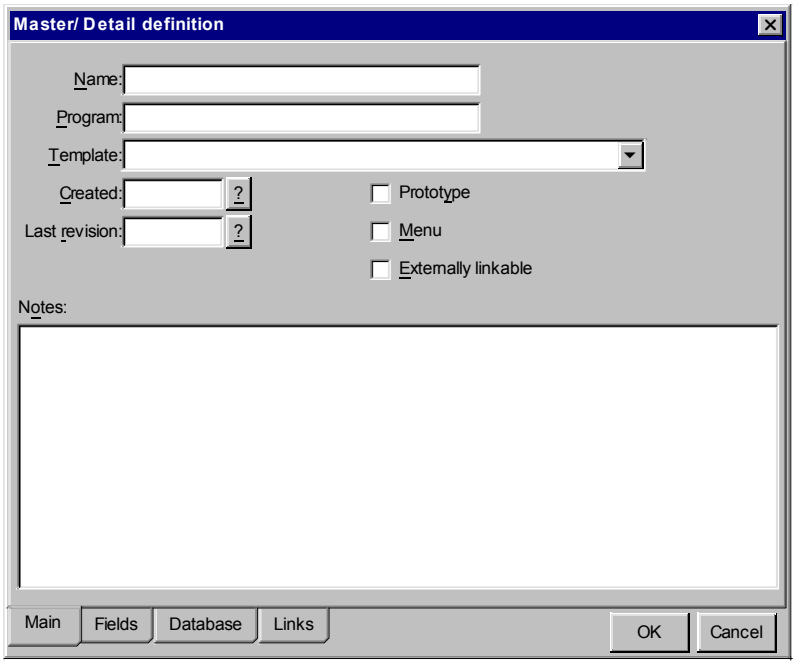
Deletes selected links.

3.9.3 Master/Detail Definition

Main Page

Main Page. Defines general information on the entity added to the design plan.

Picture 67 - Master/Detail Definition: Page 1



Name

Entity title.

The entity title is a brief description of the entity. Each entity has a unique name, which identifies it. The description helps identifying the entity's functionality without the need to follow naming rules.

The description is used by the menu prototype to create the toolbar from which the entity can be executed and by the prototype to name the created window.

Program

Name identifying the entity.

The program name must follow file naming rules. The description (previous field) must not follow any rule.

Each entity has a set of associated CodePainter repository files: codify, program and documentation files. The program name is the basis for all these files.

The Prototype tool creates for each entity a file describing the entity's characteristics. The file is used in the Codify phase. It has the name defined in the 'Program' field. The extension changes according to the kind of entity (MstDef for 'Master file', DtlDef for 'Detail file', etc.).

For each entity one or more files containing classes and procedures are defined. Whether one or more files are created depends on the programming language and the template selected. The main file will always have this name. The extension depends on the programming language.

The documentation creates more files: one for 'User' and one for 'Technical' documentation including various files used to display the documentation on the screen.

Template

Templates are procedure skeletons, which are combined with the definitions contained in the repository, so that the source program can be produced. This process is called automatic code generation template driven. Changing templates you can generate the documentation, or the source code, or change the programming language in which the program is re-generated.

Using this option you can define the template that must be used during the process.

Each entity class can have more templates. Different source codes will be generated starting from the same definition scheme stored in the repository. For example for each class a 'Parent' and a 'Child' table are generated.'Child'. 'Parent' entities are independent procedures that can be opened in the application's menu. 'Child' procedures must receive the primary key from the 'Parent'. 'Child' procedures are therefore integrated in the Parent either directly or in form of buttons from which the Child procedure can be launched. The generated code changes according to the selected template.

The combobox details all tables available for the defined entity class.

Created

Date in which the entity was created.

This date is defaulted by the system when the entity is created.

Revised

Date in which the last changes have been made.

This date is automatically up-dated when the entity is changed. This date combined with the 'Created' date gives you an idea of the time spent on defining the entity.

Entities having an old 'Revised' date are typically stable entities, where no new implementations are required. Entities having 'Created' and 'Revised' dates, which are close to each other are typically entities that must be simply managed. Entities with an old 'Created' date and a recent 'Revised' date are typically complex entities or key processes in the program's architecture. Entities having a recent 'Revised' date are typically entities under development.

Prototype

Defines whether a prototype must be created or not.

Once you defined the entity in the design plan you can create a prototype for that entity. Prototypes are fully functional programs but are incomplete in terms of calculations and validations. Further the layout usually requires improvements.

Prototypes are typically created in order to check together with customer/ user whether specifications are detailed enough to go on to the Codify phase (JAD: joined application development).

When this flag is set and the prototype generation executed, a new prototype is created that replaces the exiting one. When the flag is deselected no prototype is created for that entity.This flag is usually set until you need to work on the design plan. As soon as you start working with the Codify tools you should deselect the files. In order to integrate changes made in the design plan with a dialog window that must no longer be prototyped you can use the 'Design compare' option in the 'Edit' menu in the Codify tools. You will so identify missing fields, links, etc.

Menu

Defines whether the entity must be added to the application's menu.

For each entity a dialog window is created that allows inputting and changing data. When this flag is set the application's menu will contain the entity's name from where the associated procedure can be launched.

Externally linkable

Defines whether the entity can be seen from a different design plan.

For large applications you can create more design levels. Designs on other levels belonging to the same project can read entities in other design plans, which have this flag set.

Once the entity is exported and before you start making big changes you should verify that these changes don't have major impacts on the definitions contained in the other design levels. Within the current design plan CodePainter can automatically check (and realign if required) if a field has been used in a link and the linked field is not compatible.

Keep this simple rule in mind: in exported entities you should never delete fields or change field types.

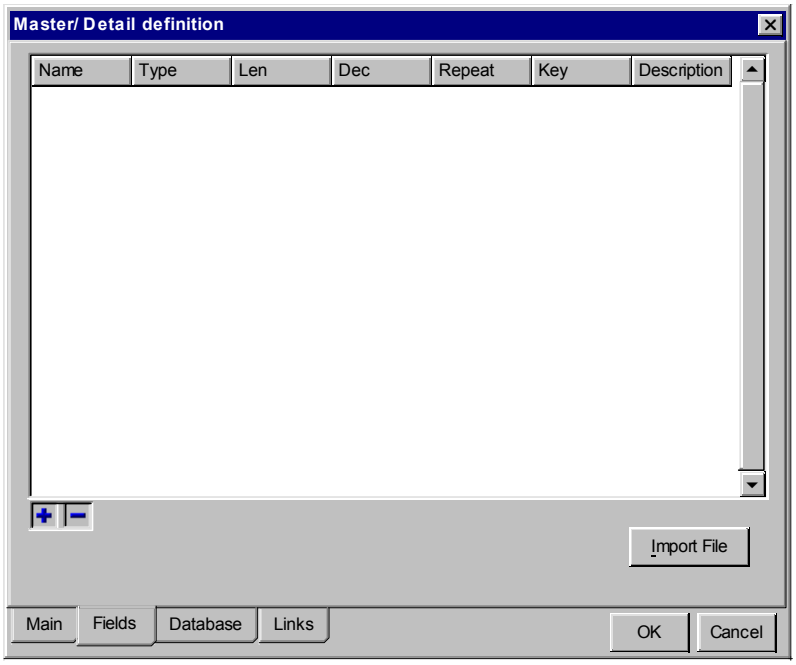
Notes

Notes on the entity.

Notes should include project requirements for each entity. These notes will be integrated in the technical documentation. You can refer to these notes at any point in time during the Codify phase.

Fields Page

Lists all fields in the entity.



Picture 68 - Master/Detail Definition: Page 2

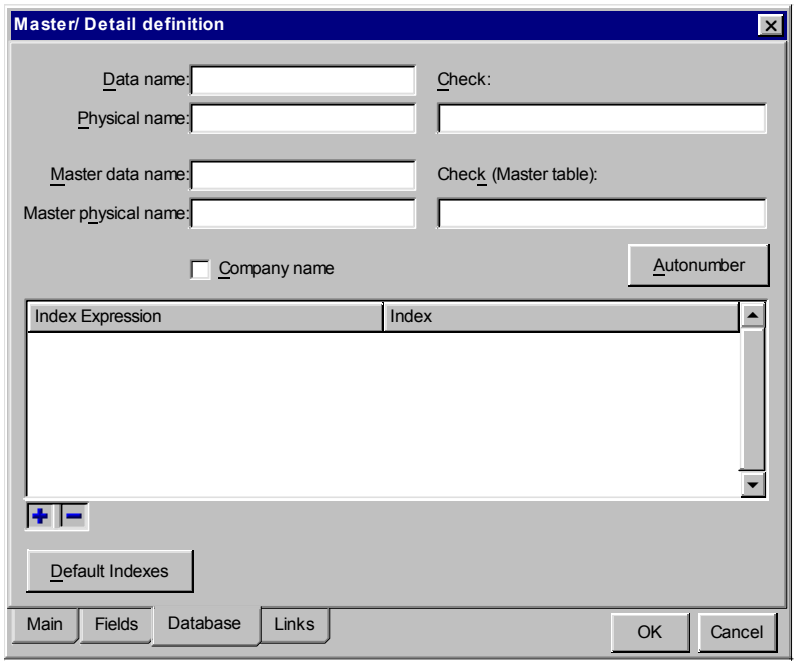
Fields

List of fields contained in the entity.

Import File

Allows importing existing xBASE or ODBC database structures.

Database Page



Picture 69 - Master/Detail Definition: Page 3

Data name

Symbolic name of the table associated to the entity.

This name is the reference alias for the relational database table associated to the entity. The physical name of the table can be different. When the multicompany system is activated for this table there can be more than one physical table associated to the symbolic name.

Physical name

Physical name of the table associated to the entity.

In relational databases tables are associated to entities. When the multicompany system is activated a table for each defined company is associated to one entity. The table used is the one of the current company.

The physical name allows giving the table a name which is different from the symbolic one. In multicompany environments the company name/ code becomes part of the physical name. The physical name must contain the string 'xxx', which will be replaced by the current company name/ code. This applies to any sentence that accesses the database.

Check

Check sentences on the table.

Here you can define a SQL sentence, which will be applied before any data is input in the database. When the sentence gives FALSE the input is refused.

mastername

Master Table symbolic name.

This name is defaulted adding the suffix "_m" to the detail table's symbolic name.

masterphname

Master table physical name.

This name is defaulted adding the suffix "_m" to the detail table's physical name. The same multicompany options explained for the 'detail' table apply.

mastercheck

Check sentence on the master table.

Here you can define a SQL sentence, which will be applied before any data is input in the database. When the sentence gives FALSE the input is refused.

Company name

Shows that the table is mutlicompany.

When the flag is set the mutlicompany system for the table associated to this entity is activated. To one symbolic name more physical tables are associated. The table used is the one linked to the current company. The physical name must contain the string 'xxx'. The string will be replaced by the name of the current company (i_CodAzi).

Autonumber

Accesses the table that defines progressive numbers.

Index expression

List of indexes.

This list details the indexes that are created for the table. The first index describes the primary key. The other indexes describe the other keys created in order to increase the speed of search and sort activities.

In relational databases each SQL sentence is analyzed and the quickest access plan iscreated (optimization). When a table has indexes these are used in order to optimize response times. This is why you should always create a set of indexes to easesearch path, e.g defining the code as well as the description as keys.

Creating indexes made of too many fields does not necessarily help the optimizer to find the quickest path. Indexes made of many elements are used only when the SQL sentence includes all fields (making the index) in the selection criteria. A good rule is to create indexes made of maximum three fields. For example you have an index made of NUMDOC (document number) and by DATDOC (document date). The SQL sentence in which the WHERE clause includes both fields can use the entire index. When the clause includes NUMDOC only it used half of the index. When the clause includes DATDOC no index is used, because DATDOC comes only after NUMDOC.

When queries mainly start from the document date your index should be made of DATDOC first and followed by NUMDOC.

When queries are mainly made on DATDOC as well as on NUMDOC you should create two indexes so that queries are always optimized.

When you want to search data by date and order it by document number you should create two indexes. The first including the fields DATDOC and NUMDOC. The second including NUMDOC only. The first index will be used when date and number, or date only are given, or when want to order data by date and number. The second index is used when the only the number is given.

Good indexes must be selective, i.e. basing on the value input for the query they must lead to results that have few records. Queries can otherwise not be optimized. For example you have the character field CUSTSEX length 1 containing M for men, W for woman and a blank for companies. You have 9000 records. Following statistics no matter which value you input you will always get one third of the records, i.e. 3000 records. You also have the 'Company' field length 30. Typically each record will contain a different value. An index on this field would probably always give you the result of 1. Generally speaking we can say that the first index does not help you much, whereas the second gives you the exact record you are searching for.

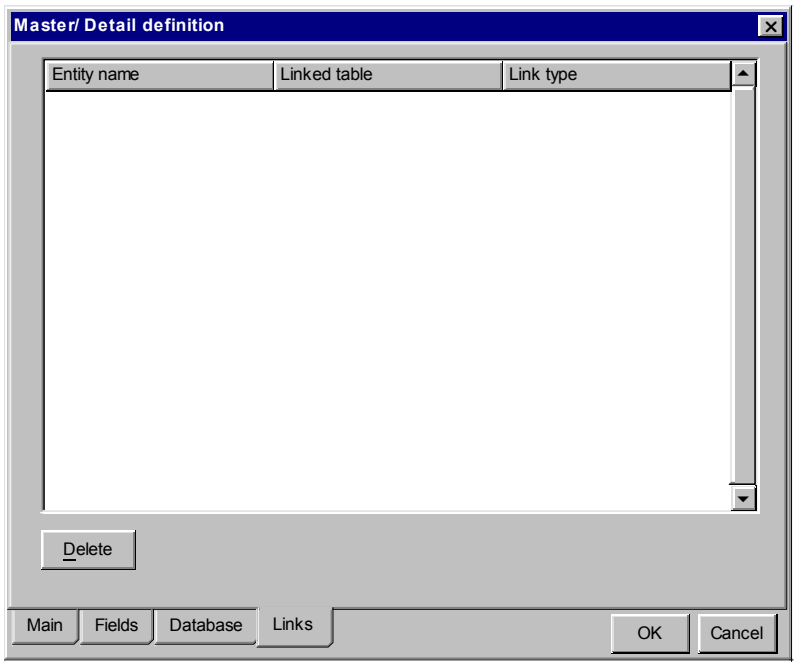
Indexes are a list of fields divided by commas.

Default Indexes

Default indexes.

This button creates default expressions for indexes basing on keys/indexes defined for fields.

Links Page



Picture 70 - Master/Detail Definition: Page 4

Links

ist the links of the current entity.

Delete

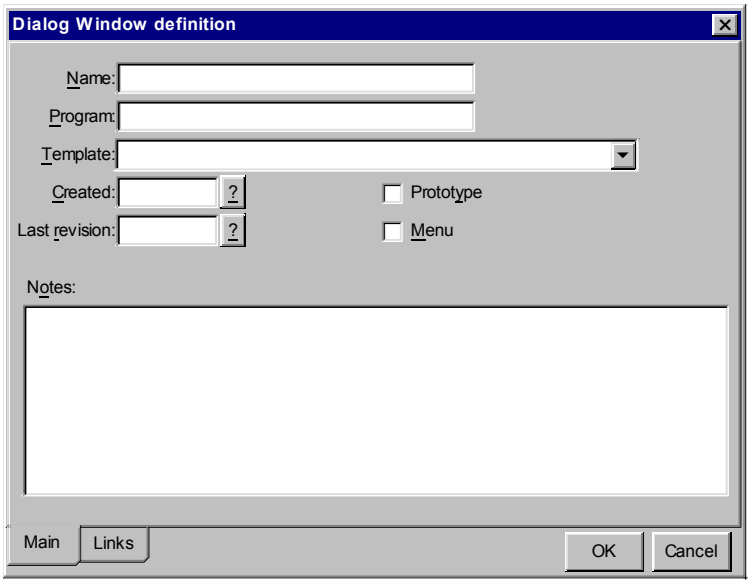
Deletes selected links.

3.9.4 Dialog Window definition

Main Page

Main Page. Defines general information on the entity added to the design plan.

Picture 71 - Dialog Window definition: Page 1



name

Entity title.

The entity title is a brief description of the entity. Each entity has a unique name, which identifies it. The description helps identifying the entity's functionality without the need to follow naming rules.

The description is used by the menu prototype to create the toolbar from which the entity can be executed and by the prototype to name the created window.

Program

Name identifying the entity.

The program name must follow file naming rules. The description (previous field) must not follow any rule.

Each entity has a set of associated CodePainter repository files: codify, program and documentation files. The program name is the basis for all these files.

The Prototype tool creates for each entity a file describing the entity's characteristics. The file is used in the Codify phase. It has the name defined in the 'Program' field. The extension changes according to the kind of entity (MstDef for 'Master file', DtlDef for 'Detail file', etc.).

For each entity one or more files containing classes and procedures are defined. Whether one or more files are created depends on the programming language and the template selected. The main file will always have this name. The extension depends on the programming language.

The documentation creates more files: one for 'User' and one for 'Technical' documentation including various files used to display the documentation on the screen.

Template

Templates are procedure skeletons, which are combined with the definitions contained in the repository, so that the source program can be produced. This process is called automatic code generation template driven. Changing templates you can generate the documentation, or the source code, or change the programming language in which the program is re-generated.

Using this option you can define the template that must be used during the process.

Each entity class can have more templates. Different source codes will be generated starting from the same definition scheme stored in the repository. For example for each class a 'Parent' and a 'Child' table are generated.'Child'. 'Parent' entities are independent procedures that can be opened in the application's menu. 'Child' procedures must receive the primary key from the 'Parent'. 'Child' procedures are therefore integrated in the Parent either directly or in form of buttons from which the Child procedure can be launched. The generated code changes according to the selected template.

The combobox details all tables available for the defined entity class.

Created

Date in which the entity was created.

This date is defaulted by the system when the entity is created.

Revised

Date in which the last changes have been made.

This date is automatically up-dated when the entity is changed. This date combined with the 'Created' date gives you an idea of the time spent on defining the entity.

Entities having an old 'Revised' date are typically stable entities, where no new implementations are required. Entities having 'Created' and 'Revised' dates, which are close to each other are typically entities that must be simply managed. Entities with an old 'Created' date and a recent 'Revised' date are typically complex entities or key processes in the program's architecture. Entities having a recent 'Revised' date are typically entities under development.

Prototype

Defines whether a prototype must be created or not. For Dialog Windows the prototype tool creates the dialog window only, because these kind of entities do not include lists of fields.

Menu

Defines whether the entity must be added to the application's menu.

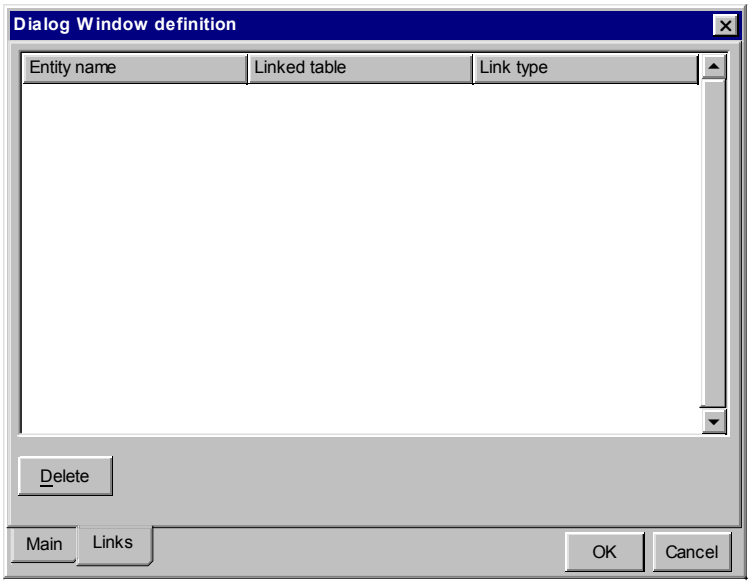
For each entity a dialog window is created that allows inputting and changing data. When this flag is set the application's menu will contain the entity's name from where the associated procedure can be launched.

Note

Notes on the entity.

Notes should include project requirements for each entity. These notes will be integrated in the technical documentation. You can refer to these notes at any point in time during the Codify phase.

Links Page



Picture 72 - Dialog Window definition: Page 2

links

List the links of the current entity.

Delete

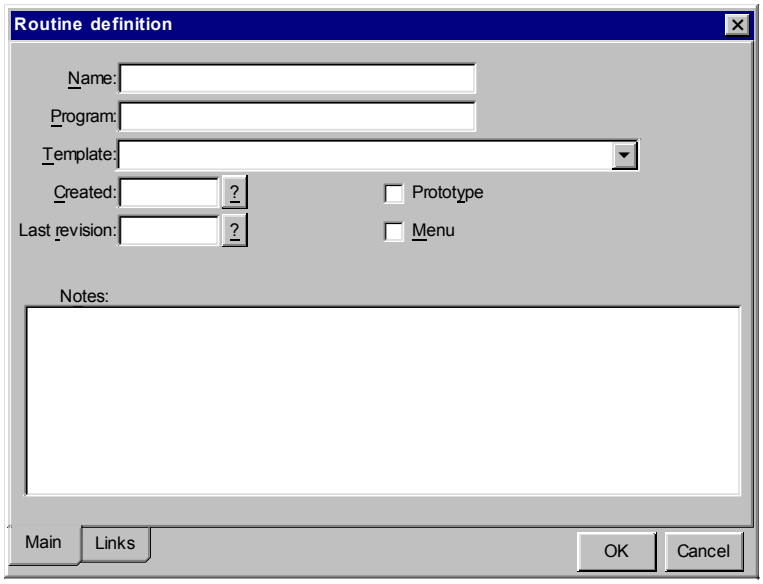
Deletes selected links.

3.9.5 Routine Definition

Main Page

Main Page. Defines general information on the entity added to the design plan.

Picture 73 - Routine Definition: Page 1



name

Entity title.

The entity title is a brief description of the entity. Each entity has a unique name, which identifies it. The description helps identifying the entity's functionality without the need to follow naming rules.

The description is used by the menu prototype to create the toolbar from which the entity can be executed and by the prototype to name the created window.

Program

Name identifying the entity.

The program name must follow file naming rules. The description (previous field) must not follow any rule.

Each entity has a set of associated CodePainter repository files: codify, program and documentation files. The program name is the basis for all these files.

The Prototype tool creates for each entity a file describing the entity's characteristics. The file is used in the Codify phase. It has the name defined in the 'Program' field. The extension changes according to the kind of entity (MstDef for 'Master file', DtlDef for 'Detail file', etc.).

For each entity one or more files containing classes and procedures are defined. Whether one or more files are created depends on the programming language and the template selected. The main file will always have this name. The extension depends on the programming language.

The documentation creates more files: one for 'User' and one for 'Technical' documentation including various files used to display the documentation on the screen.

Template

Templates are procedure skeletons, which are combined with the definitions contained in the repository, so that the source program can be produced. This process is called automatic code generation template driven. Changing templates you can generate the documentation, or the source code, or change the programming language in which the program is re-generated.

Using this option you can define the template that must be used during the process.

Each entity class can have more templates. Different source codes will be generated starting from the same definition scheme stored in the repository. For example for each class a 'Parent' and a 'Child' table are generated.'Child'. 'Parent' entities are independent procedures that can be opened in the application's menu. 'Child' procedures must receive the primary key from the 'Parent'. 'Child' procedures are therefore integrated in the Parent either directly or in form of buttons from which the Child procedure can be launched. The generated code changes according to the selected template.

The combobox details all tables available for the defined entity class.

Created

Date in which the entity was created.

This date is defaulted by the system when the entity is created.

Revised

Date in which the last changes have been made.

This date is automatically up-dated when the entity is changed. This date combined with the 'Created' date gives you an idea of the time spent on defining the entity.

Entities having an old 'Revised' date are typically stable entities, where no new implementations are required. Entities having 'Created' and 'Revised' dates, which are close to each other are typically entities that must be simply managed. Entities with an old 'Created' date and a recent 'Revised' date are typically complex entities or key processes in the program's architecture. Entities having a recent 'Revised' date are typically entities under development.

Prototype

Defines whether a prototype must be created or not.For Routines the prototype tool creates the dialog window only, because these kind of entities do not include lists of fields

Menu

Defines whether the entity must be added to the application's menu.

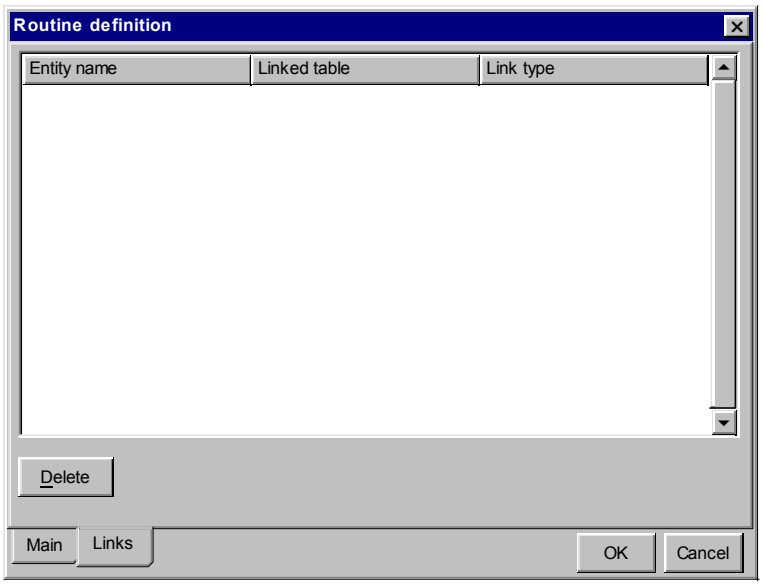
For each entity a dialog window is created that allows inputting and changing data. When this flag is set the application's menu will contain the entity's name from where the associated procedure can be launched.

Notes

Notes on entities.

Notes should include project requirements for each entity. These notes will be integrated in the technical documentation. You can refer to these notes at any point in time during the Codify phase.

Links Page



Picture 74 - Routine Definition: Page 2

links

List the links of the current entity.

Delete

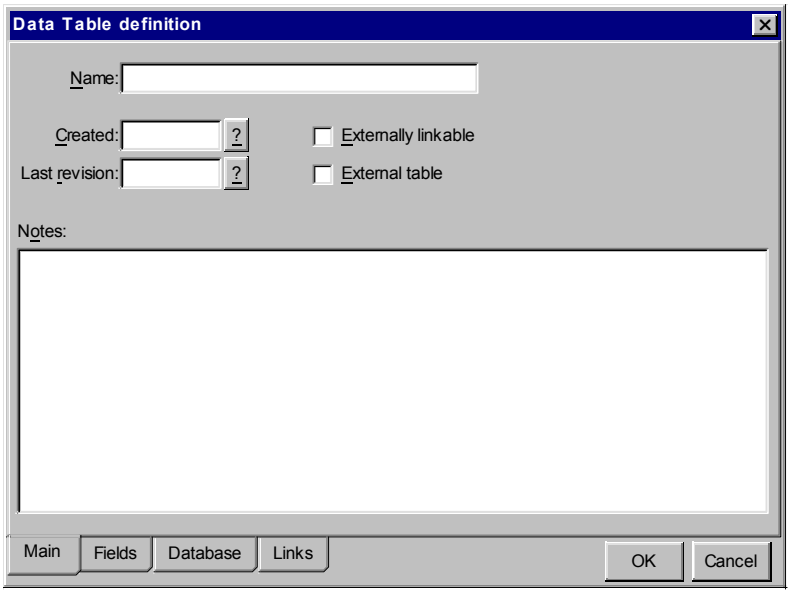
Deletes selected links.

3.9.6 Data Table Definition

Main Page

Main Page. Defines general information on the entity added to the design plan.

Picture 75 - Data Table Definition: Page 1



Name

Entity title.

The entity title is a brief description of the entity. Each entity has a unique name, which identifies it. The description helps identifying the entity's functionality without the need to follow naming rules.

The description is used by the menu prototype to create the toolbar from which the entity can be executed and by the prototype to name the created window.

Program

Name identifying the entity.

The program name must follow file naming rules. The description (previous field) must not follow any rule.

Each entity has a set of associated CodePainter repository files: codify, program and documentation files. The program name is the basis for all these files.

The Prototype tool creates for each entity a file describing the entity's characteristics. The file is used in the Codify phase. It has the name defined in the 'Program' field. The extension changes according to the kind of entity (MstDef for 'Master file', DtlDef for 'Detail file', etc.).

For each entity one or more files containing classes and procedures are defined. Whether one or more files are created depends on the programming language and the template selected. The main file will always have this name. The extension depends on the programming language.

The documentation creates more files: one for 'User' and one for 'Technical' documentation including various files used to display the documentation on the screen.

Template

Templates are procedure skeletons, which are combined with the definitions contained in the repository, so that the source program can be produced. This process is called automatic code generation template driven. Changing templates you can generate the documentation, or the source code, or change the programming language in which the program is re-generated.

Using this option you can define the template that must be used during the process.

Each entity class can have more templates. Different source codes will be generated starting from the same definition scheme stored in the repository. For example for each class a 'Parent' and a 'Child' table are generated.'Child'. 'Parent' entities are independent procedures that can be opened in the application's menu. 'Child' procedures must receive the primary key from the 'Parent'. 'Child' procedures are therefore integrated in the Parent either directly or in form of buttons from which the Child procedure can be launched. The generated code changes according to the selected template.

The combobox details all tables available for the defined entity class.

Created

Date in which the entity was created.

This date is defaulted by the system when the entity is created.

Revised

Date in which the last changes have been made.

This date is automatically up-dated when the entity is changed. This date combined with the 'Created' date gives you an idea of the time spent on defining the entity.

Entities having an old 'Revised' date are typically stable entities, where no new implementations are required. Entities having 'Created' and 'Revised' dates, which are close to each other are typically entities that must be simply managed. Entities with an old 'Created' date and a recent 'Revised' date are typically complex entities or key processes in the program's architecture. Entities having a recent 'Revised' date are typically entities under development.

Prototype

Defines whether a prototype must be created or not.

Once you defined the entity in the design plan you can create a prototype for that entity. Prototypes are fully functional programs but are incomplete in terms of calculations and validations. Further the layout usually requires improvements.

Prototypes are typically created in order to check together with customer/ user whether specifications are detailed enough to go on to the Codify phase (JAD: joined application development).

When this flag is set and the prototype generation executed, a new prototype is created that replaces the exiting one. When the flag is deselected no prototype is created for that entity.This flag is usually set until you need to work on the design plan. As soon as you start working with the Codify tools you should deselect the files. In order to integrate changes made in the design plan with a dialog window that must no longer be prototyped you can use the 'Design compare' option in the 'Edit' menu in the Codify tools. You will so identify missing fields, links, etc.

Menu

Defines whether the entity must be added to the application's menu.

For each entity a dialog window is created that allows inputting and changing data. When this flag is set the application's menu will contain the entity's name from where the associated procedure can be launched.

Externally linkable

Defines whether the entity can be seen from a different design plan.

For large applications you can create more design levels. Designs on other levels belonging to the same project can read entities in other design plans, which have this flag set.

Once the entity is exported and before you start making big changes you should verify that these changes don't have major impacts on the definitions contained in the other design levels. Within the current design plan CodePainter can automatically check (and realign if required) if a field has been used in a link and the linked field is not compatible.

Keep this simple rule in mind: in exported entities you should never delete fields or change field types.

External table

This flag is used to define database tables, which are external to the current database. This flag allows you to define tables stored on different servers, that are therefore not maintained by the current application.

Notes

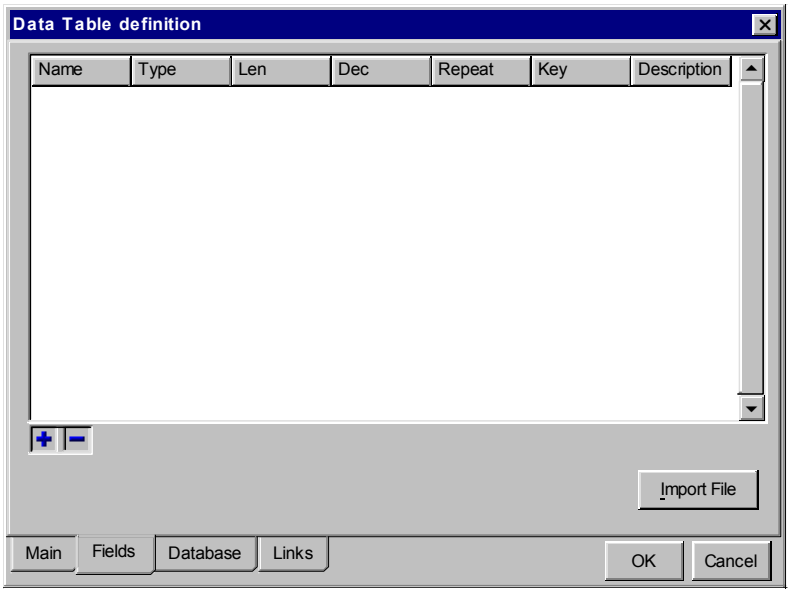
Notes on the entity.

Notes should include project requirements for each entity. These notes will be integrated in the technical documentation. You can refer to these notes at any point in time during the Codify phase.

Fields Page

Lists all fields in the entity.

Picture 76 - Data Table Definition: Page 2



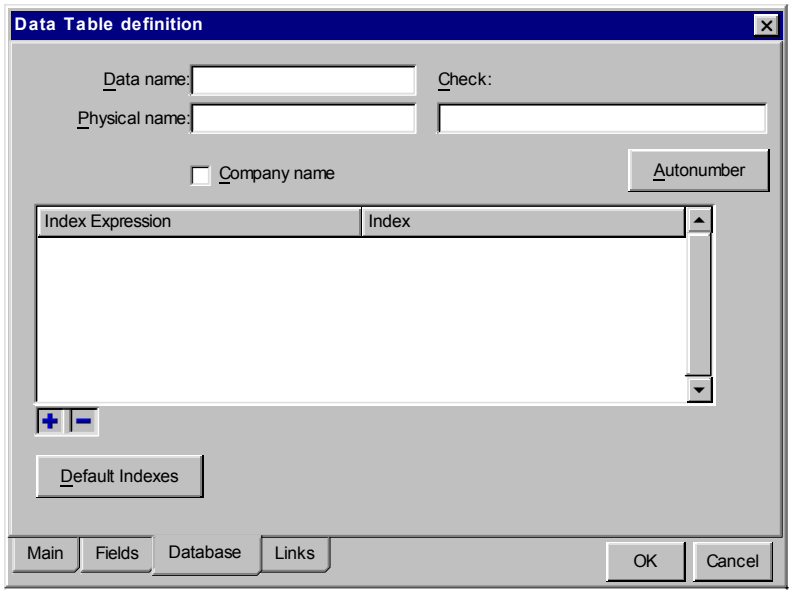
Fields

List of fields contained in the entity.

Import File

Allows importing existing xBASE or ODBC database structures.

Database Page



Picture 77 - Data Table Definition: Page 3

Data name

Symbolic name of the table associated to the entity.

This name is the reference alias for the relational database table associated to the entity. The physical name of the table can be different. When the multicompany system is activated for this table there can be more than one physical table associated to the symbolic name.

Physical name

Physical name of the table associated to the entity.

In relational databases tables are associated to entities. When the multicompany system is activated a table for each defined company is associated to one entity. The table used is the one of the current company.

The physical name allows giving the table a name which is different from the symbolic one. In multicompany environments the company name/ code becomes part of the physical name. The physical name must contain the string 'xxx', which will be replaced by the current company name/ code. This applies to any sentence that accesses the database.

Check

Check sentences on the table.

Here you can define a SQL sentence, which will be applied before any data is input in the database. When the sentence gives FALSE the input is refused.

mastername

Master Table symbolic name.

This name is defaulted adding the suffix "_m" to the detail table's symbolic name.

masterphname

Master table physical name.

This name is defaulted adding the suffix "_m" to the detail table's physical name. The same multicompany options explained for the 'detail' table apply.

mastercheck

Check sentence on the master table.

Here you can define a SQL sentence, which will be applied before any data is input in the database. When the sentence gives FALSE the input is refused.

Company name

Shows that the table is mutlicompany.

When the flag is set the mutlicompany system for the table associated to this entity is activated. To one symbolic name more physical tables are associated. The table used is the one linked to the current company. The physical name must contain the string 'xxx'. The string will be replaced by the name of the current company (i_CodAzi).

Autonumber

Accesses the table that defines progressive numbers.

Index expression

List of indexes.

This list details the indexes that are created for the table. The first index describes the primary key. The other indexes describe the other keys created in order to increase the speed of search and sort activities.

In relational databases each SQL sentence is analyzed and the quickest access plan iscreated (optimization). When a table has indexes these are used in order to optimize response times. This is why you should always create a set of indexes to easesearch path, e.g defining the code as well as the description as keys.

Creating indexes made of too many fields does not necessarily help the optimizer to find the quickest path. Indexes made of many elements are used only when the SQL sentence includes all fields (making the index) in the selection criteria. A good rule is to create indexes made of maximum three fields. For example you have an index made of NUMDOC (document number) and by DATDOC (document date). The SQL sentence in which the WHERE clause includes both fields can use the entire index. When the clause includes NUMDOC only it used half of the index. When the clause includes DATDOC no index is used, because DATDOC comes only after NUMDOC.

When queries mainly start from the document date your index should be made of DATDOC first and followed by NUMDOC.

When queries are mainly made on DATDOC as well as on NUMDOC you should create two indexes so that queries are always optimized.

When you want to search data by date and order it by document number you should create two indexes. The first including the fields DATDOC and NUMDOC. The second including NUMDOC only. The first index will be used when date and number, or date only are given, or when want to order data by date and number. The second index is used when the only the number is given.

Good indexes must be selective, i.e. basing on the value input for the query they must lead to results that have few records. Queries can otherwise not be optimized. For example you have the character field CUSTSEX length 1 containing M for men, W for woman and a blank for companies. You have 9000 records. Following statistics no matter which value you input you will always get one third of the records, i.e. 3000 records. You also have the 'Company' field length 30. Typically each record will contain a different value. An index on this field would probably always give you the result of 1. Generally speaking we can say that the first index does not help you much, whereas the second gives you the exact record you are searching for.

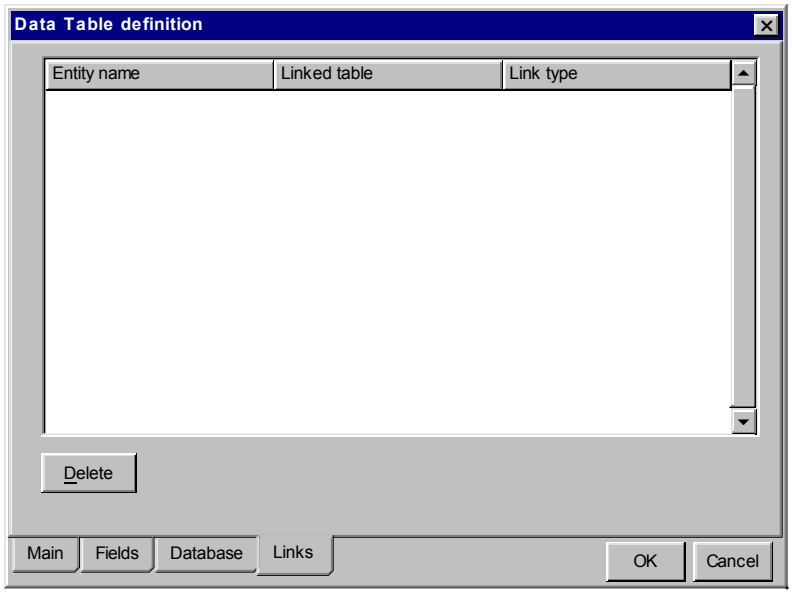
Indexes are a list of fields divided by commas.

Default Indexes

Default indexes.

This button creates default expressions for indexes basing on keys/indexes defined for fields.

Links Page



Picture 78 - Data Table Definition: Page 4

Links

List the links of the current entity.

Delete

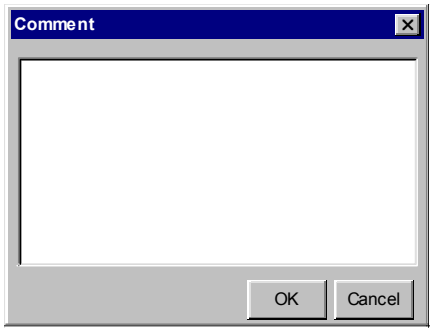
Deletes selected links.

3.9.7 Comment

Dialog window that defines the contents of comment strings on the design plan.

Strings are notes that comment important aspects of the design plan. They complement the notes defined in each entity.

Picture 79 - Comment: Page 1



Name

Comment displayed within the design plan string.

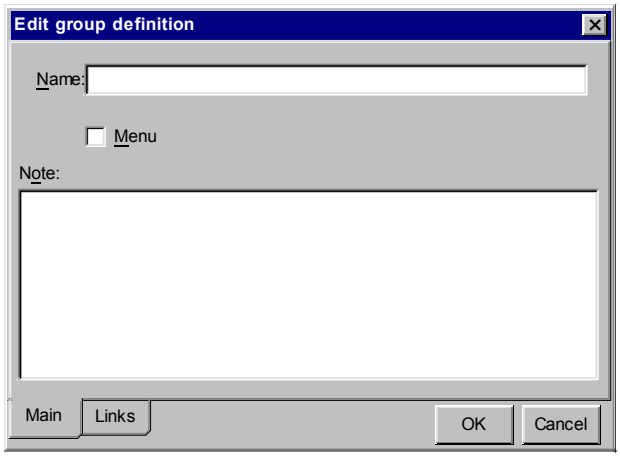
3.9.8 Edit Group Definition

Properties of design groups.

Defines properties of groups added to the design plan. Groups are entities containing other entities that logically belong together. This allows dividing the design plan on more levels.

Main Page

Main Page. Defines general information on the entity added to the design plan.



Picture 80 - Edit Group Definition: Page 1

Name

Name of the group.

Menu

Defines whether the group must be added to the application's menu.

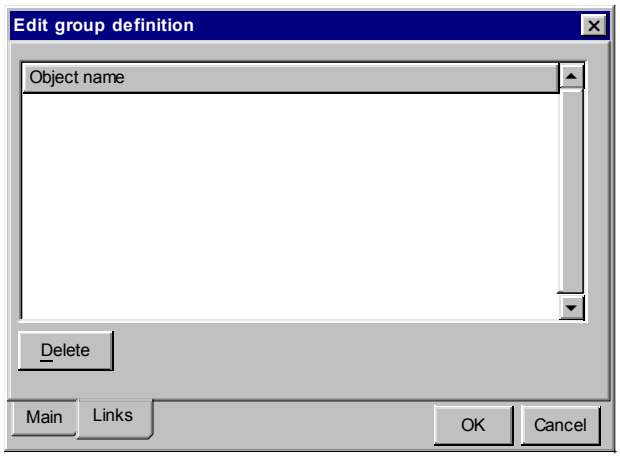
When the group becomes part of the application's menu a submenu will show all entities belonging to the group. When the group is not part of the menu, entities belonging to the group will be in the application's main menu.

Note

Notes describing the group. Here you can define why you have grouped entities, i.e. the reasoning behind the grouping. Notes be included in Technical documentation.

Links Page

Picture 81 - Edit Group Definition: Page 2



Links

Links defined for the group.

Links between a group and other entities display links between entities in the group and entities outside the group. Each group can be considered as a design sub-level.

In order to ease link management, linked entities in the group are smaller. This is to remind you that the links have been defined elsewhere.

Delete

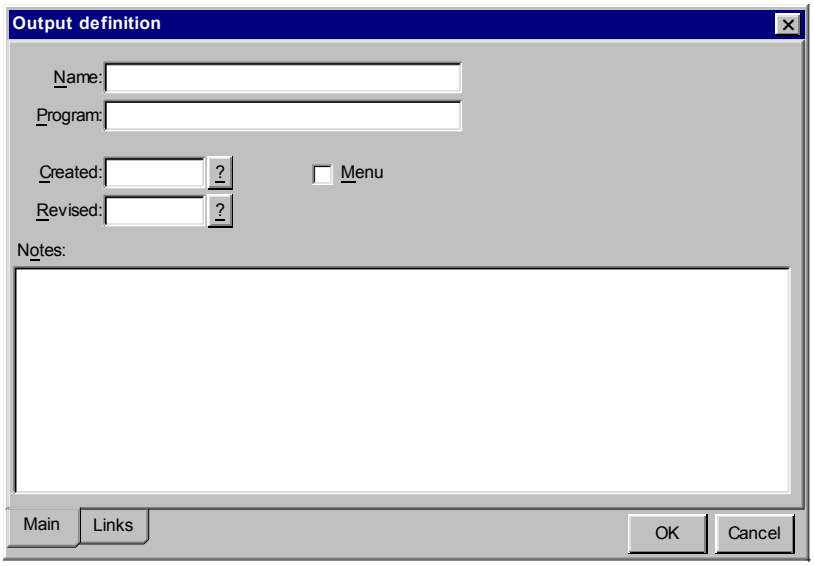
Deletes selected links from the link listbox.

You can also delete links right clicking the line representing the desired link.

3.9.9 Output Definition

Main Page

Main Page. Defines general information on the entity added to the design plan.



Picture 82 - Output Definition: Page 1

Name

Name of the Output entity.

Procedure

Name of the configuration that must be executed.

Output entities hook VisualUTK, the system that manages queries and reports. VisualUTK is available in any application generated with CodePainter. Output entities are typically used to add a print option to the prototype's menu. This is made defining the name of a query (compulsory) and the entity layout (optional).

In the design plan you should add only those reports included in the application specifications. All other reports should be created directly in the application.

Created

Date in which the entity was created.

This date is defaulted by the system when the entity is created.

Revised

Date in which the last changes have been made.

This date is automatically up-dated when the entity is changed. This date combined with the 'Created' date gives you an idea of the time spent on defining the entity.

Entities having an old 'Revised' date are typically stable entities, where no new implementations are required. Entities having 'Created' and 'Revised' dates, which are close to each other are typically entities that must be simply managed. Entities with an old 'Created' date and a recent 'Revised' date are typically complex entities or key processes in the program's architecture. Entities having a recent 'Revised' date are typically entities under development.

Menu

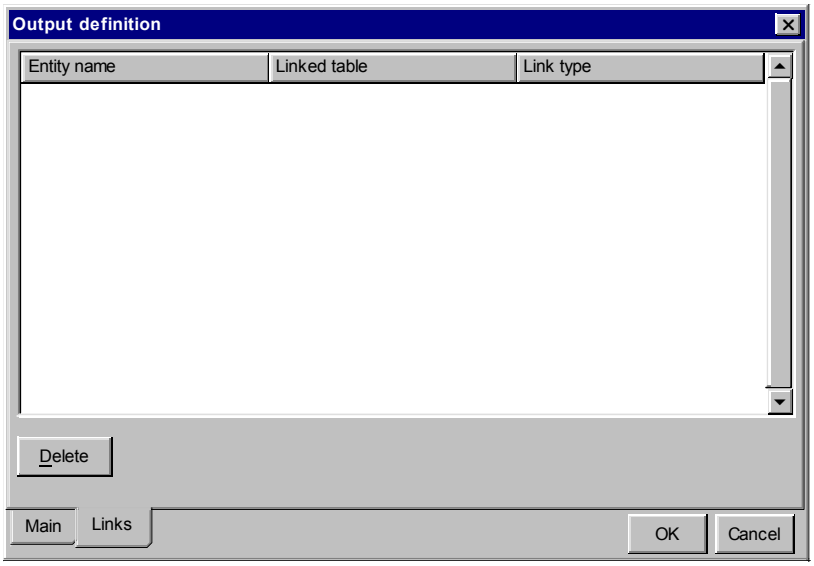
Defines whether the entity must be added to the application's menu.

For each entity a dialog window is created that allows inputting and changing data. When this flag is set the application's menu will contain the entity's name from where the associated procedure can be launched.

Notes

Notes on the entity. Notes should include project requirements for each entity. These notes will be integrated in the technical documentation. You can refer to these notes at any point in time during the Codify phase.

Links Page



Picture 83 - Output Definition: Page 2

Links

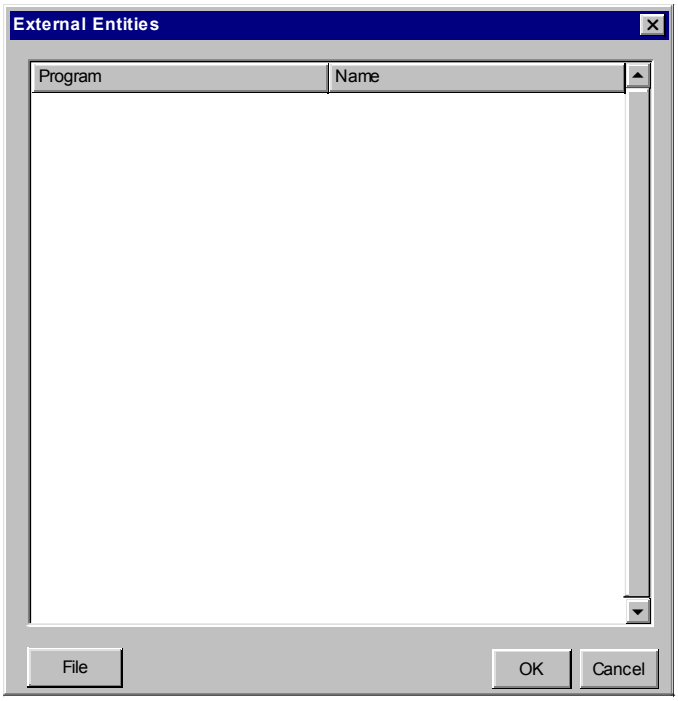
List the links of the current entity.

Delete

Deletes selected links.

3.9.10 External Entities

Picture 84 - External Entities: Page 1



Entities

List of entities exported from the selected design plan.

Any design plan can export entities that can be read by other design plans: either in the same project or in other projects.

When entities are imported a link between the current design plan and the entity is established. In the current design plan you cannot change the entity's definitions. When you edit the entity data is read from the exporting design plan.

File

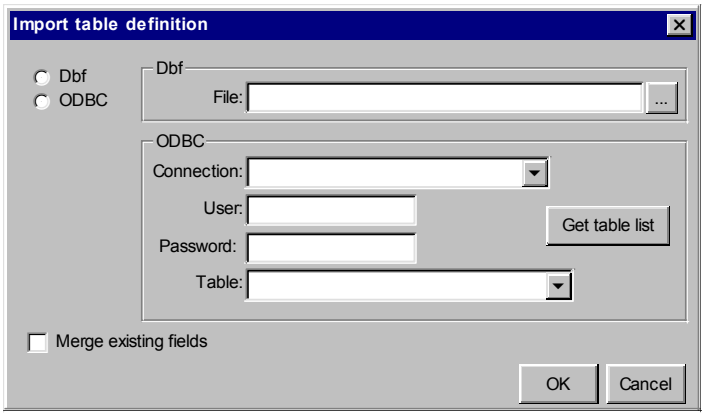
Button that allows to select the design plan from which you want to import the entity's definitions.

3.10 Importing Existing Structures

You can import existing structures from DBF files or from tables belonging to databases defining the entity's 'Field' folder.

3.10.1 Import Table Definition

This window guides you through the process of reading existing files or tables in order to import them in CodePainter.



Picture 85 - Import Table Definition: Page 1

Connection Mode

Dbf

Defines that you want to read the definition of a DBF file on the disk.

ODBC

The ODBC connection is used to read the definition of an existing database table.

Connection

ODBC connection. This combobox allows to select the ODBC connection in which the imported table must be searched. The combobox defaults the latest defined connection.

File To Import

Name of the file that must be imported. This textbox shows the name of the file that must be imported. Using the '...' button next to it you can browse to find the file name.

...

Display the disk contents to facilitate the selection of the file that must be imported.

uid

User ID used to connect to the ODBC database.

pwd

Connection password.

table

Table that must be imported.

Get table list

Button that allows you to download the list of ODBC database tables. In order to see the list you need to define connection, user and password.Afterwards, clicking this button all available database tables are displayed.

Merge Existing Fields

Defines whether to replace existing fields or merge imported fields with the ones you have already defined. When the flag is deselected existing fields are deleted and the table will be identical to the imported one.

When the flag is set exiting fields are kept and fields of the imported database are added. If a field already exists, it takes on type, length and decimals of the imported field. Other data remains unchanged. This is helpful when you need to update an existing table basing on a new external table. Importing the new database customizations the existing table remains unchanged. New fields are added and realigned.

3.11 Field Definition

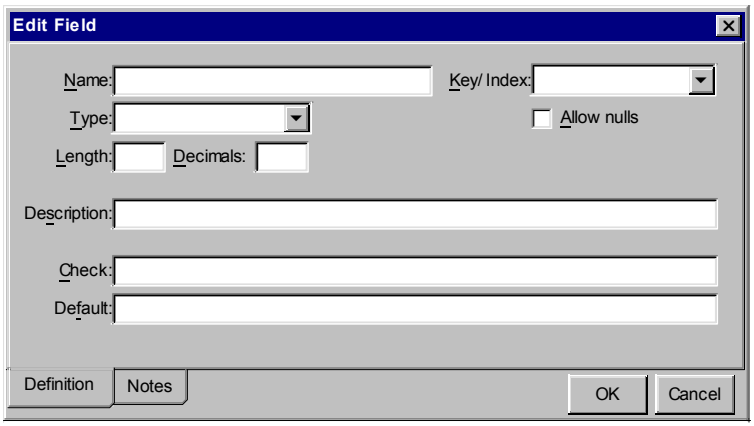
This section analyzes the 'Edit Field' dialog window for Master File and Database Table entities.

3.11.1 Edit Field

Allows to add database field parameters.

Definition Page

Picture 86 - Edit Field: Page 1



Name

Name of the field.

A field having this name is created in the database. The defined string must therefore follow the rules for field identifier of target systems on which the application is run. Usually the name can be as long as required, must begin with a letter and cannot include point, columns, semi columns or slashes.

Key/Index

Defines whether the field belongs to a primary or secondary field.

It is extremely important to define which fields belong to the primary key. Primary keys are used to check links and to create referential integrity.

Secondary indexes are less critical. Once defined the system can automatically create alternative search criteria, either using the required indexes or activating the field during search or editing phases.

Type

Field type.

The field type defined in this combobox is automatically 'translated' the target database. For example a numeric field length 4 with no decimals is translated in as INTEGER, because it is more efficient than numeric. The translation will always include the required parameters and can include others as well if these are more efficient.

Many databases have restrictions in changing column types. CodePainter therefore minimizes the need for these changes, changing the field structure only when it is strictly necessary.

Repeated

Defines repeated fields.

Repeated fields belong to the body of Detail or Master/Detail entities.

For Detail entities this means a different input method during the Codify phase. For Master/Detail entities this means that the field belongs to the detail table and not to the master table. Unrepeated fields of the primary key are the exception. They belong to both the master and detail table.

Length

Field length.

The field length can be either variable, e.g. for character and numeric, or fixed, e.g. for date and memo.

The length defines the field size. Many databases have restrictions to these changes in order to avoid that working applications are affected by new applications working on the same database.

CodePainter minimizes changes changing the fields in the database only if the new size is longer than the old one.

Decimals

Decimal number for numeric fields.

Description

Brief description of the field.

Descriptions are displayed in the lists next to the field name. They are used by the prototype tool to create the dialog window. Descriptions will be displayed next to fields as labels. In the body of Master/Detail or Detail entities descriptions are used as columns' headings.

The prototype tool creates labels for all fields in order to best support multilingual applications. Labels are aligned to the right and usually have empty spaces on the left. When the labels are translated the new word can therefore be longer without changing the dialog window's layout.

Check

Check expression for the field.

During the database programming phase this expression is added as CHECK clause on the field. The expression must be written in SQL syntax. The database does not accept values that do not comply to the check.

Default

Field default value.

When a field is not included in the list of fields initializated by the program INSERT this expression (in SQL syntax) defines which value must be given to the field. CodePainter usually initializates all fields contained in the table. User defined routine procedures may include INSERT sentence having incomplete field lists. In this latter case the defined value will be input in the fields. Otherwise they would be defaulted with the value NULL.

In order that 'total' sentences can go together with relations, all numeric fields are defined with the default value 0.

Allow nulls

Defines whether the field allows null values.

This checkbox defines whether the field can be created without a starting value, i.e. null.

Fields of primary keys cannot be initializated. Primary keys are used to find records. If the primary key contains zeros this can no longer happen. This is why the flag is deactivated.

All other fields may take on nulls depending on requirements. When the flag is deselected the database does not accept NULLs.

Notes Page

Picture 87 - Edit Field: Page 2

Notes

Field notes.

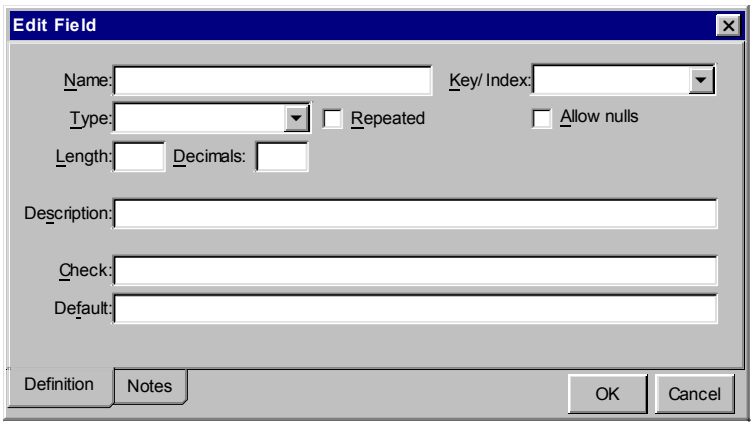
These notes are included in the Technical documentation. These notes usually contain specific requirements of the field.

The next section analyzes the 'Edit Field' dialog window for Detail and Master/Detail entities.

3.11.2 Edit Field

Definition Page

Picture 88 - Edit Field: Page 1



Name

Name of the field.

A field having this name is created in the database. The defined string must therefore follow the rules for field identifier of target systems on which the application is run. Usually the name can be as long as required, must begin with a letter and cannot include point, columns, semi columns or slashes.

Key/Index

Defines whether the field belongs to a primary or secondary field.

It is extremely important to define which fields belong to the primary key. Primary keys are used to check links and to create referential integrity.

Secondary indexes are less critical. Once defined the system can automatically create alternative search criteria, either using the required indexes or activating the field during search or editing phases.

Type

Field type.

The field type defined in this combobox is automatically 'translated' the target database. For example a numeric field length 4 with no decimals is translated in as INTEGER, because it is more efficient than numeric. The translation will always include the required parameters and can include others as well if these are more efficient.

Many databases have restrictions in changing column types. CodePainter therefore minimizes the need for these changes, changing the field structure only when it is strictly necessary.

Repeated

Defines repeated fields.

Repeated fields belong to the body of Detail or Master/Detail entities.

For Detail entities this means a different input method during the Codify phase. For Master/Detail entities this means that the field belongs to the detail table and not to the master table. Unrepeated fields of the primary key are the exception. They belong to both the master and detail table.

Length

Field length.

The field length can be either variable, e.g. for character and numeric, or fixed, e.g. for date and memo.

The length defines the field size. Many databases have restrictions to these changes in order to avoid that working applications are affected by new applications working on the same database.

CodePainter minimizes changes changing the fields in the database only if the new size is longer than the old one.

Decimals

Decimal number for numeric fields.

Description

Brief description of the field.

Descriptions are displayed in the lists next to the field name. They are used by the prototype tool to create the dialog window. Descriptions will be displayed next to fields as labels. In the body of Master/Detail or Detail entities descriptions are used as columns' headings.

The prototype tool creates labels for all fields in order to best support multilingual applications. Labels are aligned to the right and usually have empty spaces on the left. When the labels are translated the new word can therefore be longer without changing the dialog window's layout.

Check

Check expression for the field.

During the database programming phase this expression is added as CHECK clause on the field. The expression must be written in SQL syntax. The database does not accept values that do not comply to the check.

Default

Field default value.

When a field is not included in the list of fields initializated by the program INSERT this expression (in SQL syntax) defines which value must be given to the field. CodePainter usually initializates all fields contained in the table. User defined routine procedures may include INSERT sentence having incomplete field lists. In this latter case the defined value will be input in the fields. Otherwise they would be defaulted with the value NULL.

In order that 'total' sentences can go together with relations, all numeric fields are defined with the default value 0.

Allow nulls

Defines whether the field allows null values.

This checkbox defines whether the field can be created without a starting value, i.e. null.

Fields of primary keys cannot be initializated. Primary keys are used to find records. If the primary key contains zeros this can no longer happen. This is why the flag is deactivated.

All other fields may take on nulls depending on requirements. When the flag is deselected the database does not accept NULLs.

Notes Page

Picture 89 - Edit Field: Page 2

note

Field notes.

These notes are included in the Technical documentation. These notes usually contain specific requirements of the field.

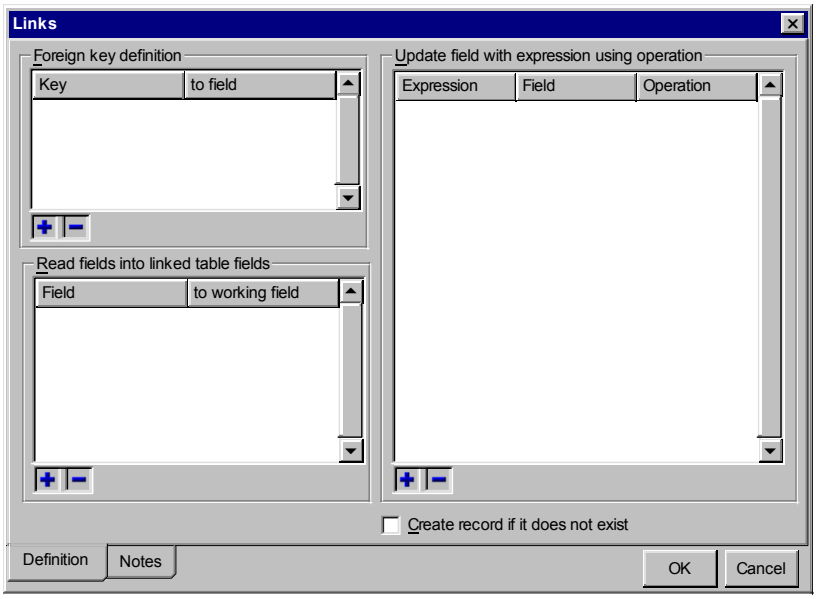
3.12 Defining Links Between Entities

This section analyzes all available kind of links between entities.

3.12.1 Links

Dialog window in which you can define the relationship between two entities. The 'Relationship' link is used to define the database referential integrity. It can sum up values stored in a file and store 'Totals' in the fields of another table.

Definition Page



Picture 90 - Links: Page 1

fkkeys

List of fields involved by the relationship and liked with the entity's primary key.

This list details which fields are linked to the primary key of the linked entity. These fields are in a dedicated list, because they are the ones the user defines before the search is started. They also check referential integrity.

During the Codify phase after the user has defined these fields a search in the linked file is started. If data is found all fields including those of the list beneath are returned. If data is not found an error message is displayed and the value of the last field in the list is set to null.

During the database management the fields of the list lead to the creation of a FOREIGN KEY sentence that programs referential integrity in the database.The generated code executes the validations but cannot guarantee that the database is not changed while it is checked and saved. Programming the database you can be sure that all validations are made directly from the system under transaction, i.e. in an isolated and protected environment. Further the database validates also the reverse integrity, i.e. it does not allow that records are deleted if there are other records referencing it.

The list is divided in two columns. The left column defines the primary key fields of the correlated database. The right column defines the fields of the current table which are linked to the other table.

For example, in the relationship 'Items' and 'Orders' the list has CODART on the left and ARTORD on the right. In the 'Orders' table you need the sentence FOREIGN KEY (ARTORD) REFERENCES ITEMS (CODART). During editing when a value is input in the field ARTORD the system searches in the 'Items' primary key. It is accepted only if there is a key having the same value.

readfrom

List of fields that must be read from the linked file.

Fields in this list define which data must be read after the system has checked that the required key in the linked field exists.The generated procedure searches the key using the fields contained in the list of key fields. If the key is found all values in the list are given.

The list is divided in two columns. The left column contains fields of the linked file that must be read. The right column contains the fields of the current table in which values must be returned.

For example consider the relationship between 'Orders' and 'Items'. This list would contain DESART and PRZART on the left and DA_ORD and PRZORD on the right. After having found an item with the searched key all other values are returned as well.

writefrom

List of fields involved in 'totalization' transactions.

To each relationship you can associate 'write' transactions. When the user saves a record you can define that certain values must be summed up and the result stored in fields of the related file. Doing so you can have current balances of orders, sales, etc.

You may want that the ordered quantity is added item by item and the result stored in a dedicated field. Here you would define QTAORD on the left, QTAART in the middle and the mathematical symbol '+' on the right.

The list has three columns. The left column contains the fields of the current table that must store the total. The column in the middle contains the fields of the related table storing the total value. The column on the right contains the mathematical symbol.

When the user confirms the data entry with 'OK' the system sums up QTAORD quantities of single ordered items by row and stored them in QTAART.Changing data involves the reversal of the old transaction and saving the new balance.

The rule is defined easily even if transactions made are very complex and require many code lines.

The column on the left contains the name of the current table that must be added. The column in the middle contains the fields of the correlated table in which totals are stored. The column on the right contains a mathematical sign: '+' to add, '-' to reverse, '=' to replace or the name of a field (length 1) of the current table.

Usually transactions are described by constants. This means that they cannot vary. You can also define transactions using a field where its value can change. In the first case the balance is given by a calculation. In the second case by a description table.

Let's analyze the description table a little closer. In 'Items' you have the three fields QTAORD, QTAESI, QTAVEN, containing ordered, stocked and sold quantities.The description table containing transaction codes has the primary key CODCAU and the fields OP_ORD, OP_ESI_OP_VEN containing the transactions that must be applied depending on the selected transaction codes. You also need a 'Transactions' table containing standard transactions for a warehouse. This table must read the transaction codes from the description table. The table contains the fields CAUMOV, ARTMOV, QTAMOV, OP_ORD_MOV, OP_ESI_MOV, OP_VEN_MOV and two relationships: one with 'Items' and one with the 'Description' table.

In the relationship with the 'Description' table the list will contain CODCAU and CAUMOV on the top left establishing relationship rules; OP_ORD and OP_ORD_MOV, OP_ESI and OP_ESI_MOV, OP_VEN and OP_VEN_MOV on the bottom left so that once the transaction code is found the transactions can be executed.

In the relationship between 'Items' and 'Transactions' the list will contain CODART and ARTMOV on the top left (definition of the foreign key). Top right you will define QTAMOV, QTAORD, OP_ORD_MOV in the first row, QTAMOV, QTAESI, OP_ESI_MOV in the second and QTAMOV, QTAVEN, OP_VEN_MOV in the third.

Moved quantities are added to or subtracted from the fields containing balances of ordered, stocked and sold quantities depending on what is written in the description table. Inputting the record with OP_ORD='+', OP_ESI=' ' and OP_VEN=' ' you have an order transaction. Inputting the record with OP_ORD='-', OP_ESI='+' and OP_VEN=' ' you have a transaction detailing that you received items in your warehouse. Inputting the record with OP_ORD=' ', OP_ESI='-' and OP_VEN='+' you have a sales transactions.

When the transaction is variable it must be stored in a field otherwise CodePainter cannot create the reversal of the transactions in case of changes or cancellations.

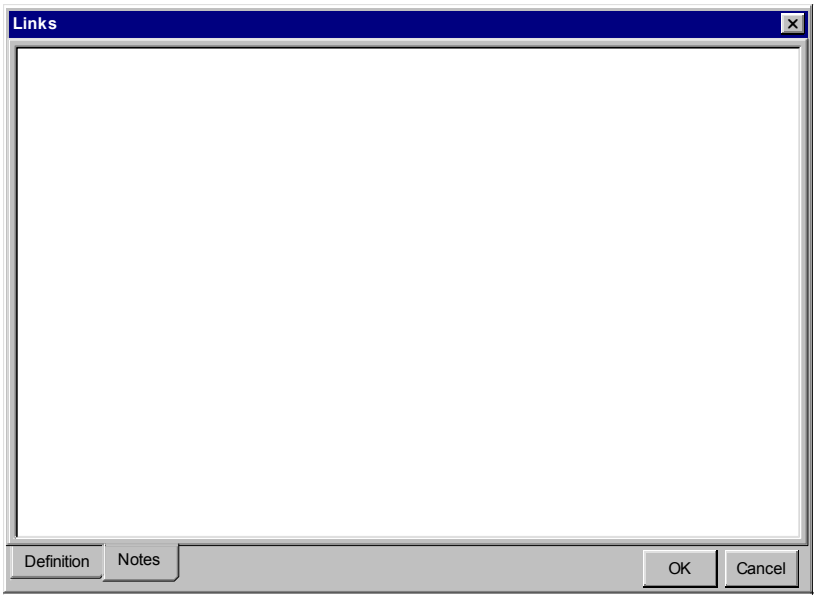
Create Record If It Does Not Exist

Flag defining that records must be created if they don't exist.

Relationships can be made of key and update information only. In the Codify phase you can load the key with a calculated value and hide the field. In this case the search is not executed and the system tries to update it. This could fail, because the corresponding field is not found. When this flag is set a new record is added to the related file. The key and update fields are entered.

This kind of relationship is used when you need to create a link to the 'Child' table checking only the existence of a key in the 'Parent'. For example consider the 'Items' entity as the 'Parent' and the 'Totals' entity as the 'Child' storing sales totals per year. For each record in 'Items' there is a 'total' record by year in which there are sales values. The relationship with 'totals' requires the 'Items' key. The key is checked and then the relationship with 'Totals' established. The 'Items' key is set by the first relationship and the 'Year' key is calculated basing on the transaction date. When the record is saved you have a complete key that may not exist in the 'Totals' table if no previous sale was made in the current year, i.e no record exists. When this flag is set the record is created within the transaction avoiding an error message saying that the key is not valid.

Notes Page



Picture 91 - Links: Page 2

Notes

Notes on the relationship.

This notes are included in the Technical documentation.

The next section analyzes a 'Parent/Child' link.

3.12.2 Parent/Children Link Definition

Defines the relationship type 'Parent/ Child'.

Definition Page

Picture 92 - Parent/Children Link Definition: Page 1

fkkeys

Fields involved in 'Parent/Child' relationships.

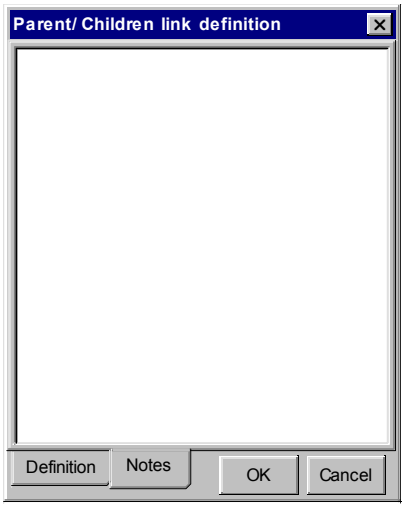
Parent/Child relationships have a 'strong' (parent) entity linked to a 'weak' (child) entity which cannot exist without the parent entity. Transactions made on the Parent are made also on the Child (cascading). This means that when the key is changed on the Parent it must be changed on the Child as well. When the Parent is deleted the Child must be deleted as well.CodePainter does not enable the user to change the primary key. This issue must be managed 'internally'. The key may change when the 'booked' number on autonumbered fields is given to a different record.

This list defines the linked fields (in Parent and Child). The column on the left defines the key fields of the Parent. The column on the right defines the key fields of the Child. Please note that the link must always include the keys of both entities.

When Child key is made by more fields of the Parent key the kind of relationship created is one to many and the Child entity must be a Detail File. When the Child key has the same number of fields as the Parent key the relationship created is one to one and the Child must be a Master File. This latter kind of relationship is used to spread files containing many fields on more tables in order to avoid possible conflicts in multiuser environments.

CodePainter defaults the two keys in the list. You can either change it or simply confirm and the relationship is established.

Notes Page



Picture 93 - Parent/Children Link Definition: Page 2

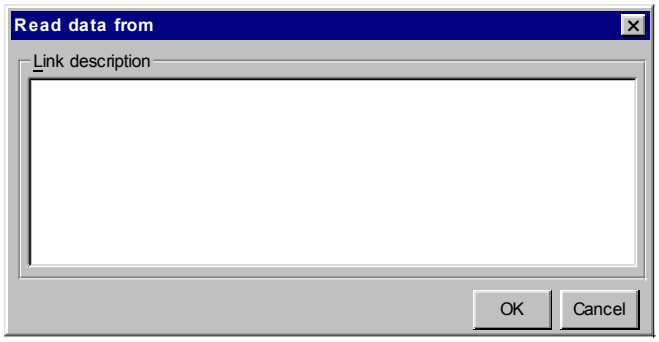
Notes

Notes on the relationship.These notes are included in the Technical documentation.

In this section we will examine in detail the definition of the links of Data Flow type.

3.12.3 Read Data From

Picture 94 - Read Data From: Page 1



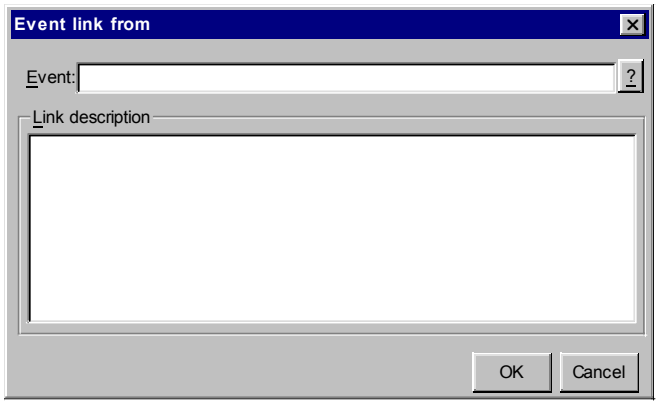
Notes

Notes on the relationship.This kind of relationships are read only and they tyoically document user requirements. These notes are included in the Technical documentation.

The next section analyzes the 'Event' link.

3.12.4 Event Link From

Event link.



Picture 95 - Event link from: Page 1

event_

Name of the event.

Defines the name of the event in the related entity that must trigger the current entity.

Notes

Notes on the relationship.These notes are included in the Technical documentation. They typically contain relationship requirements.

3.13 Managing Autonumbered Values

Master File, Detail File and Master/Detail entities defined in the design phase can manage progressive numbering automatically. To define autonumbered values edit the desired entity, go to the 'Database' folder and click the 'Autonumber' button.

3.13.1 Autonumber

Manages progressive numbering automatically.

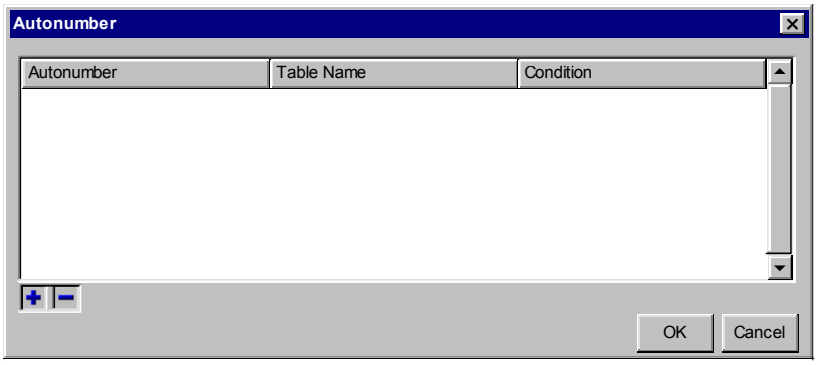
You can define fields that are numbered automatically. Once the starting number is defined the field is increased every time a new record is input.

Defining the 'autonumber' option for a field, when a new record is input the field is initialized with the next value in the numbering sequence . If the user does not change the value and the record is saved the table containing progressive numbering is read. The system checks whether any other user has taken the 'booked' number. If the 'booked' number is still available the record is saved using it. If not the next available number is identified, the record saved with this new number and a message displayed so that the user is notified of the change.

This method makes sure that in multiuser environments all available numbers are used and that no numbering gaps are created.

When the user edits the 'booked' value the field value is overridden. The 'custom' number could be greater or less than the one given by the system. When the number is greater than the one given by the system the record is saved with that number and a numbering gap is created. When the number is less than the one given by the system the record is saved taking on the 'booked' field value.

This second method allows to postpone data entry for known quantities. For example invoices are input by Head office. Unit B has issued 10 invoices but has not yet send the data. Head office can leave a gap of ten units overriding the next invoice number. When the data from Unit B arrives it can be input using the 10 left empty numbers.



Picture 96 - Autonumber: Page 1

Autonumber

List of progressive numbers.

This list contains the fields building the progressive number, the reference table, and if required the expression that defines whether the progressive system must be used or not.

The list is used by the Prototyping tool to identify progressive fields to be applied in Codify phase.

Autonumber

Name of the field that will be linked to a progressive table. You can define more fields simply dividing them with commas. In this latter case the last field is the one taking on progressive numbering, whereas the other fields are the 'variations'.

Example

```
You need to number a specific document and you also need to
differentiate in-coming from out-going documents. You need to define
two fields:TIPDOC and NUMDOC. The system will create progressive
numbers  for each TIPDOC value and inputting the progressive number in
NUMDOC. Fields receiving progressive numbers can be either numeric or
alphabetical. In order to match the alphabetical order with the sequence
character types '0' are added to in front of the letter until the field is
filled in completely.
```

Table Name

Name of the progressive value. Progressive values are stored in a dedicated table. This field contains a symbolic name required to identify which numbering must be used. You can create separate numberings, e.g. 'cli' for the customer key, 'art' for the item code) or numberings that can be used by more entities, e.g. different documents such as orders and invoices, using the same numbering.

Condition

Sometimes you may be required to use progressive numbering only under certain conditions. For example you may need to save in-coming and out-going in one entity only. You define the field NUMDOC that must be automatically numbered for out-going documents, but in which you want to manually input numbers for in-coming documents. If out-going documents have the value 'O' in the field TIPDOC you need to define the condition TIPDOC='O' so that progressive numbers are used only for these documents.

Autonumber
You want automatic progressive numbering on invoices. The field TIPODOC contains the document type ('I' for invoices, 'R' for receipts) The field NUMDOC must contain the progressive number for invoices and the manually inputted numbers for other documents.The list will contain: 'NUMDOC' in the 'Autonumber' column.'doc' in the 'Table Name' column.'TIPDOC="F"' in the 'Condition' column.

3.14 Product Information

Information on CodePainter can be accessed opening the 'Help' menu and selecting the 'About' option.

3.14.1 About

The About dialog window contains information on the procedure in use.

The logo CODEPAINTER REVOLUTION is displayed and information on the version, series number and build number are given.

The build number helps you identifying CodePainter's distribution, no matter if the product is released totally or partially or through patches.

Picture 97 - About: Page 1

Chapter 4

Master File

4.1 Introduction

Master File entities are made of a database table and a procedure, which manages the user interface where you can input, change or delete data. These entities process records one by one.

Master Files are used to manage customer files, item files, etc. Data on customers or items are defined in single records. Changes to information on one customer will affect one record only.

Master Files can be created also in the Codify phase basing on the application's data dictionary. In this case you have two different procedures that manage the same database table.

The Master File Painter is used to improve the prototype's layout and functionalities, or to create new Master File entities that will be included in the design plan.

Using the Master File Painter you can improve the window's layout adjusting elements' size and position. You can also add strings, variables, buttons etc. using the WYSIWYG (What You See Is What You Get) methodology. You can further define initialization properties, calculations, validation checks, etc.

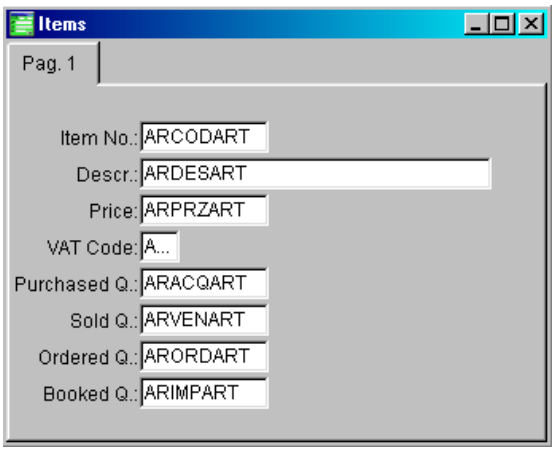
When in the Codify phase you need to change the Design plan you can maintain the changes made and generate the prototype for changed elements only. This can be achieved using comparison functions that will be explained later on.

4.2 The Working Logic

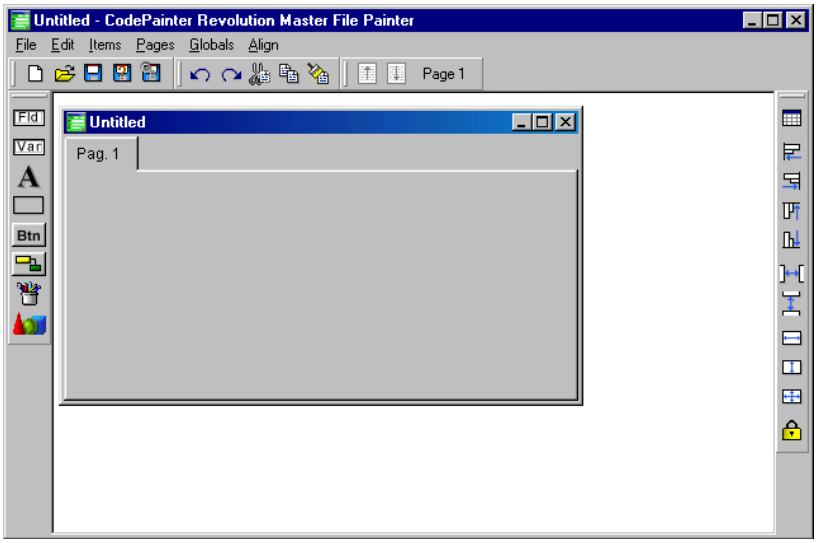
In the Master File Painter you can open prototyped entities. A set of toolbars help you interacting with the prototyped entities. You can either open entities defined in the design plan or create new ones basing on the application's data dictionary.

To open defined entities open the 'File' menu and select the 'Open' option. All existing Master File entities are listed. Fields defined in the Master File at design level have been prototyped. Fields and descriptions defined in the linked table are included in the Master File prototype and are sorted basing on the data dictionary.

Picture 98 - Master File Painter Prototype



To create new entities open the 'File' menu and select the 'New' option. You can also start working on the blank window displayed when the Master File Painter is opened.



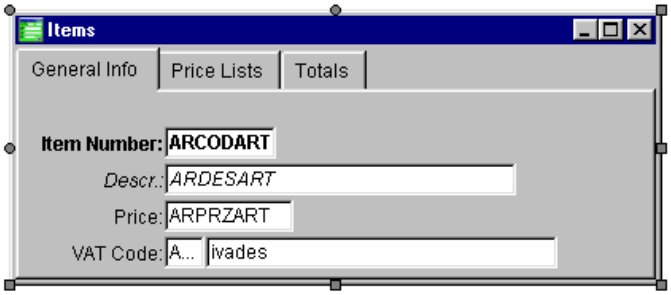
Picture 99 - Master File Painter: New Window

Using the Master File Painter you can '*design*' the interface and build in validations. Using the 'Code Generation' option you obtain a working procedure without the need to regenerate the entire design.

In the opened entity you can select, move and resize elements as you would do in any other MS Window application.

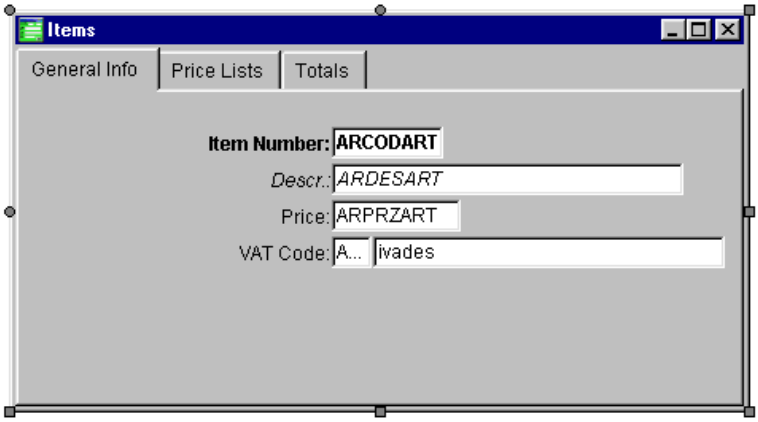
To move the entity window click the title (Caption Bar) and drag it in the desired position. You can notice that while you move the window the mouse changes into a cross.

Picture 100 - Select The Window

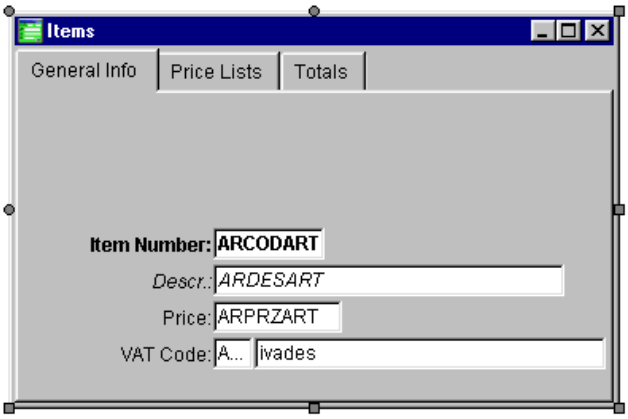


To resize the window you need to select the entity and drag the displayed anchors. The size is always changed according to the direction in which you are dragging when the mouse becomes a 'two ways' arrow. The elements in the window remain fixed compared to the left corner.

Picture 101 - Window Re-sizing By Dragging



When you right click and drag one of the circled window handles elements remain fixed compared to the bottom right corner. You can notice that while you right click and drag the circle an anchor is displayed under the 'two ways' arrow.



Picture 102 - Window Re-sizing By Right Clicking Circles

Similarly you can move and re-size any element in the entity window.

4.3 Selecting And Aligning Elements

Groups of elements can be selected in different ways:

Left clicking the mouse you can drow a rectangle around the elements you wish to select.

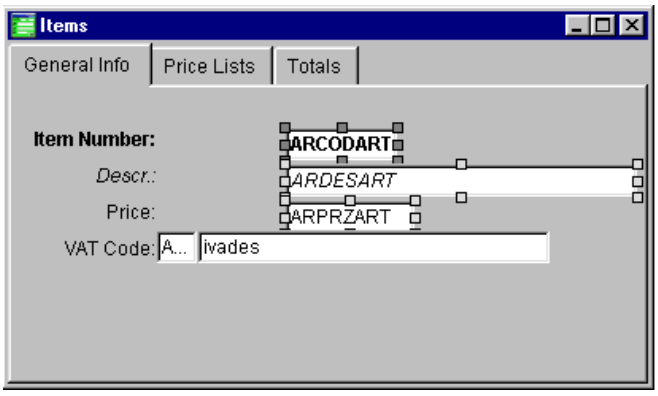
Picture 103 - Selected Group Of Elements

You can also keep the **<Shift>** key pressed and click the elements you wish to select. Single elements are deselected still keeping the **<Shift>** key pressed and clicking again the desired element. All elements are deselected clicking on the working form.

Selected elements are highlighted by handles, little *squares* around the element. Most elements with handles are light grey and one only dark grey. The latter one is the *Master Item* and is used as reference for resizing and aligning the other elements

Warning
When resizing and realigning elements the result changes depending on the Master Item selected.

Aligning the Master Item



Picture 104 - Align to the Left Of The Master Item

Picture 105 - Align to the Left Of The Master Item

The two pictures above show you the different results you will get selecting two different master items. The Master Item can be changed also once the group of elements has been selected simply clicking the desired item again.

You can align and re-size group of elements using both the 'Align' menu or toolbar.

To open the definition dialog window double click the desired element.

Right clicking elements a menu is opened that allows you to edit the element's properties, delete the element or change the editing sequence.

Pressing **<Del>** you can delete one or more selected elements.

Pressing **<Enter>** you can open the definition dialog window of the selected item or in case of multiple selection of the Master Item.

Pressing **<Cursor Keys>** you can move one or more selected elements.

Pressing **<Shift>** and **<Cursor Keys>** you can resize the selected element or in case of multiple selection the Master Item keeping the top right corner fixed.

Pressing **<Ctrl>** and **<Shift>** and **<Cursor Keys>**you can resize the selected element or in case of multiple selection the Master Item keeping the bottom left corner fixed.

Using keyboard keys you can also scroll elements.

Pressing **<Tab>** you can scroll forward.

Pressing **<Shift>** and **<Tab>** you can scroll backwards.

Pressing **<Ctrl>** and **<Tab>** you can scroll a selected group of elements forward.

Pressing **<Ctrl>** and **<Shift>** and **<Tab>** you can scroll a selected group of elements backwards.

To scroll the window elements standard Windows rules apply:

Pressing **<Tab>** you can scroll forward.

Pressing **<Shift>** and **<Tab>** you can scroll backwards.

When you are positioned on tabstrips you can press the **<Cursor Keys>** to scroll the option pages.

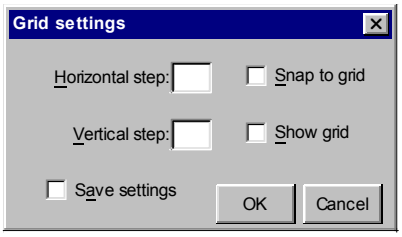
Pressing **<Alt>** and the <UnderlinedLetter> you can edit the corresponding option.

In selection lists you can sort columns. You can also add and delete elements using either the mouse or the **<Ins>** and **<Del>** keys.

When you move elements using the **<Cursor Keys>** changes are madein pixels. When you align elements you can define whether to use a positioning grid or not.

4.3.1 Grid Settings

Design grid definition.



Picture 106 - Grid Settings: Page 1

dx

Horizontal path. The grid has anchors. The distance between anchors is given by the number of pixel defined starting from the left border of the design window.

dy

Vertical path. The grid has anchors. The number of anchors is given by the number of pixel defined starting from the top border of the design window. When the dialog window has more pages the anchor is placed on the top border of the tab-strip.

Snap to grid

Defines whether the elements must be within the grid. When the flag is not selected the elements can be placed anywhere in the window. When the flag is set elements must be in fixed places.

Show grid

Defines whether the grid must be displayed or not. When elements in the dialog window are many and close to each other the grid makes the window unreadable. When this flag is not selected the grid is still active, but is not displayed.

Save settings

Defines whether grid settings must be saved or not. Grid settings can be defined and saved for each dialog window. When this flag is set settings are maintained.

4.4 Master File Painter Toolbar

Let us now analyze the Master File Painter toolbars. These toolbars help you interacting with the Master Files.

4.4.1 Painter Tools Toolbar

The Painter Tools toolbar allows you to add new elements to the current entity. The toolbar has the same functionalities as the 'Items' menu.

figura 5 - Painter Tools Toolbar

Here to follow you will find a brief description of the files and their meanings.

Button	Meaning
	Adds database table 'Field' objects.
	Adds 'Memory Variable' objects to support validations and calculations/totals.
	Adds descriptive 'String' objects.
	Adds 'Box' objects that allow you to improve the entity's layout adding lines or rectangles that help subdividing the window.
	Adds 'Button' objects. These objects run queries, procedures, system functions and/or dialog windows.
	Adds 'Parent/Child Link' objects that allow you to create buttons that are integrated in the window or that open 'Child' entities.
	Adds Bitmap objects.
	Adds 'Object' objects that allow you to integrate external objects to your project. External objects belong to predefined classes. Examples are graphs, calendars, zooms, zooms with selection, etc.

4.4.2 File Toolbar

The 'File' toolbar has a set of button that help you interacting with the tool. This toolbar has the same functionalities as the 'File' menu.

Picture 107 - File Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Creates new entities.
	Opens existing entities.
	Saves the changes made to the current entity.
	Saves the current entity with a different name.
	Saves and generates the source code for the current entity.

4.4.3 Align Toolbar

The *Align* menu allows you to position and resize groups of selected elements.

figura 6 - Align Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Opens the grid management
	Aligns elements to the left in comparison to the Master Item.
	Aligns elements to the right in comparison to the Master Item.
	Aligns elements to the top in comparison to the Master Item.
	Aligns elements to the bottom in comparison to the Master Item.
	Positions elements with the same horizontal distance in comparison to Master Item and the selected elements.
	Positions elements with the same vertical distance in comparison to Master Item and the selected elements.
	Re-sizes selected elements with the same height as the Master Item.
	Re-sizes selected elements with the same width as the Master Item.
	Re-sizes all elements with the same height and width as the Master Item.
	Blocks the possibility to move and re-size elements (the functionalities are not inhibited in the 'Align' menu.

4.4.4 Clipboard Toolbar

The 'Clipboard' toolbar allows you to execute typical Window's commands such as Undo, Redo, Cut, Copy and Paste.

Picture 108 - Clipboard Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).
	The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).
	The 'Copy' option copies one or more selected elements.
	The 'Cut' option deletes selected elements.
	The 'Paste' option pastes copied or cutted elements in the selected position.

4.4.5 Pages Toolbar

Using the 'Pages' toolbar you can browse the entity's pages. The toolbar shows two buttons and the number of the page in which you are positioned.

Picture 109 - Pages Toolbar

Here to follow you will find a brief description of the buttons and their meanings

Button	Meaning
	Moves the edit functionality from the previous page to the current page. This button does not work if you are on the first page.
	Moves the edit functionality from the following page to the current page. This button does not work if you are on the last page.

4.5 The Main Menu

4.5.1 File

Picture 110 - File Menu

The 'File' menu has a set of options to:

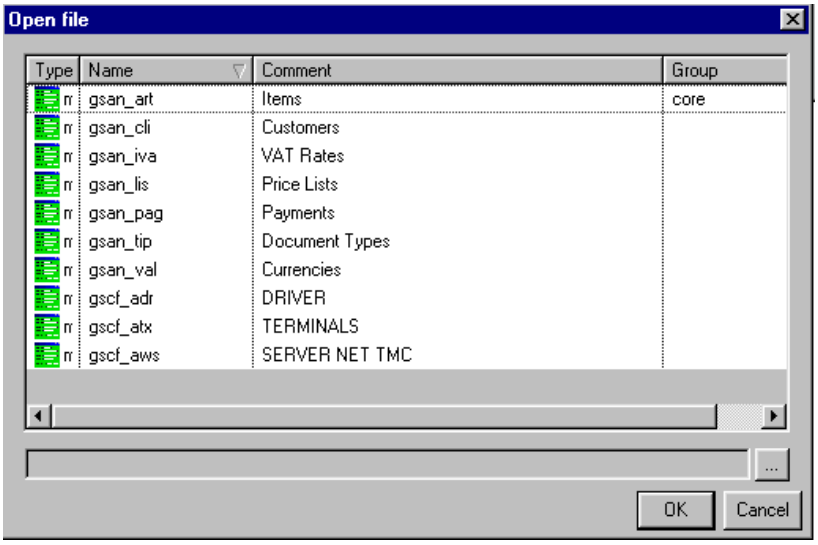
- Create new entities.
- Load existing entities.
- Save changes to the current entity.
- Save the current entity with a different name.
- Save and generate the current entity.
- Read information about CODE**PAINTER REVOLUTION**.
- Exit the tool.

New

The 'New' option closes the current entity asking whether you want to save the changes or not and opens a new entity.

Open...

The 'Open' option loads definition files belonging to the current project.



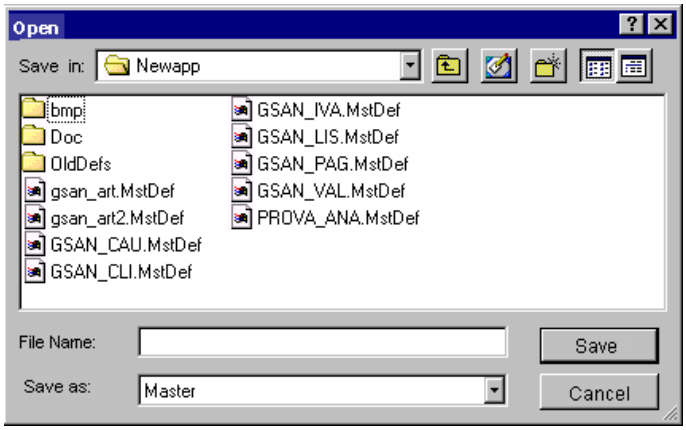
Picture 111 - Open File

The dialog window lists all entities in the project. You can order files by Type, Name, Comment or Group clicking the corresponding column. You can identify the sorting column by the arrow next to the column description.

Picture 112 - Sort Columns

Clicking the '...' button you can browse to search entities of the same type belonging to other project modules.

Picture 113 - Open Dialog Window To Select Design Plans



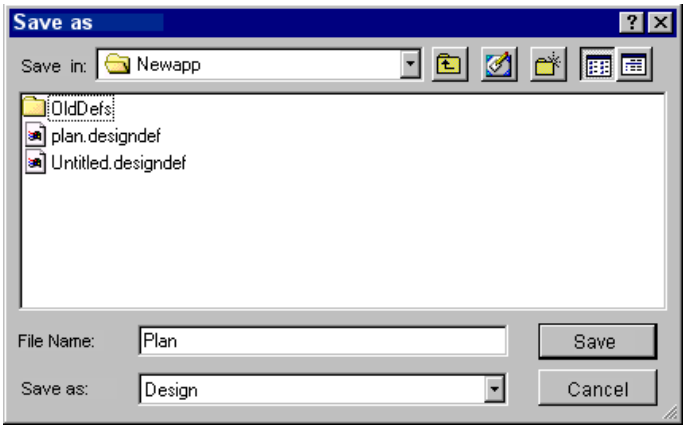
Save

The 'Save' option saves the entity in use.

When you save the entity back-up file (.BAK) is created to store the entity without including the latest changes.

Save As...

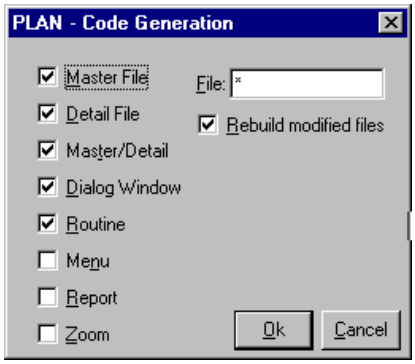
The 'Save As' option saves the entity with a different name.



Picture 114 - Save As Dialog Window

Save and generate

The 'Save And Generate' option saves the current entity and runs the generation. The same is done opening the 'Generation' menu and selecting the 'Design'.



Picture 115 - Code Generation

About

The 'About' option displays information about the product.

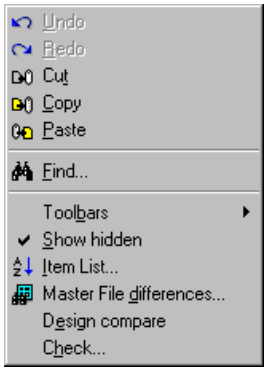
For more information please refer to section 'Product Information'.

Exit

The 'Exit' option makes you exit the tool asking you whether you want to save the changes or not.

4.5.2 Edit

Picture 116 - Edit Menu



Opens a menu from where you can:

- Undo/redo commands.
- Delete, copy and add elements.
- Search, display, replace elements.
- Show/ hide toolbars.
- Show/ hide elements with the 'Editing' flag set to 'Hide'.
- Display the 'Item List'.
- Identify the discrepancies between Codify and the Design.
- Identify errors in the source code.

Undo

The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).

Redo

The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).

Copy

The 'Copy' option copies one or more selected elements. More elements can be either selected or grouped in a selection frame you can draw with the mouse.

Cut

The 'Cut' option deletes selected elements from the plan.

Paste

The 'Paste' option pastes copied or cutted elements in the selected position.

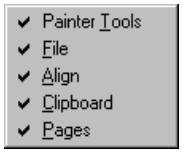
Find...

The 'Find' option opens the Find and Replace dialog window to find and/or replace elements.

Toolbar

The Toolbars option has displays a submenu to enable or disenable toolbars on the Design Painter.

Picture 117 - Menu Toolbar



Show hidden

The 'Show Hidden' option allows you to show or hide variables for which the 'Editing' option has been set to 'Hide'.

Item List...

The 'Item List' option displays the a list of all elements in the window.

File Painter Differences...

Using this option you can compare the entity with the design plan. For more information please refer to section 'Edit Menu Advanced Options'

Design Compare

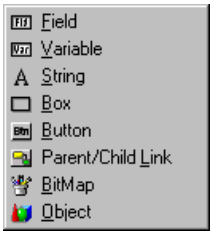
Using this option you can compare the file definition with the entity defined in the Design phase. For more information please refer to section 'Advanced Edit Menu Options -Design Compare'.

Check...

The 'Check' option checks the congruence of expressions and relations defined during the Codify phase. For more information please refer to section 'Edit Menu Advanced Options'.

4.5.3 Items

The 'Items' menu allows adding elements to the entity in use. It has the same functionality as the 'Painter Tools' toolbar.



Picture 118 - menu' Items

Here to follow you will find a brief description of the various elements:

Field

Using this option you can add 'Field' objects corresponding to a field of the associated table to your entity. When this option is selected the 'Fields of Tables ...' window is opened containing the list of the entity's fields. The list is read from the data dictionary.

Variable

Using this option you can add Variable objects to your entity. These objects are memory variables used to support validations and calculations/totals.

String

Using this option you can add String objects to your entity. These objects are descriptive strings.

Box

Using this option you can add Box objects to your entity. These objects allow you to improve the entity's layout adding lines or rectangles that help subdividing the window.

Lines are defined '*collapsing*' box objects using **<Shift>** + **<arrows>** keys, or the mouse.

Button

Using this option you can add Button objects to your entity. These objects are typically defined to launch queries, process procedures, system functions and/or Dialog windows.

N. B.
Using Button objects to integrate the entity with system functions ('Help', 'Query', 'Edit', 'Load', 'Save', 'Delete', 'Quit', 'PgUp', 'PgDn', 'ZoomPrev', 'ZoomNext') does inhibit standard toolbar functionalities in the generated application.

Parent/Child Link

Using this option you can add 'Parent/Child Link' objects to your entity. At Design level 'Parent/Child' relationships mean hierarchical links between two entities. Used as objects they allow you to create buttons that are integrated in the window or that open 'Child' entities.

Right clicking these objects you can access to the 'Open Codify...' option and the other standard options ('Edit', 'Delete', 'Sequence'). The 'Open Codify...' option allows you to open the codify tool of the integrated Child entity.

Bitmap

Using this option you can add Bitmaps to your entity.

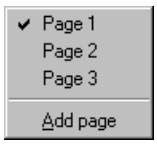
Object

Using this option you can add 'Object' objects to your entity. These objects allow you to integrate external objects to your project. External objects belong to predefined classes. Examples are graphs, calendars, zooms selecting zooms, etc.

For more information on External Object Classes and/or on how to define new classes please refer to the COMPONENT GUIDE manual.

4.5.4 Pages

Using the 'Pages' menu you can browse the entity's pages. This menu has the same functionalities as the 'Pages' toolbar. New pages can be added only using the 'Pages' menu.



Picture 119 - Pages Menu

The menu displays the names of the available pages. The current page has the 'check' symbol next to the name.

Add Page

Adds a page to the entity. To delete a page you need to delete all contained elements. When you open the entity the next time, CodePainter identifies that the page is empty and deletes it automatically.

N. B.
In order to be able to access the pages in the generated application, at least one field needs to be editable. If your project requires that all fields in a page cannot be edited, you need to add two editable buttons with the system functions 'Page Up' and 'Page Down'.

4.5.5 Globals

In the 'Globals' menu you can define general information on the Master File entity in use.

This dialog window displays information defined in the 'Design' phase and does not include changes made in entities' prototypes.

For more information please refer to section 'Globals Menu Advanced Options'.

Picture 120 - Global Menu

Global...

This menu opens the 'Global Definition' window where you can define the generation template, the icon that must be associated to the entity, the author, the user, the developing language, the object that must be launched when the dialog window is saved, etc.

Tables...

In this option you can define the design plan in which the entity is included, the master table used and the work tables that it must open.

Font...

This option allows changing default fonts for the entity in use.

Default fonts are defined in the Front End under 'Project/Project Options'.

Pages Title...

This option allows defining titles for the entities pages and some notes that will be integrated in the application's documentation.

Autonumber...

In this option you can manage Autonumbered fields for the entity in use.

Manual Blocks...

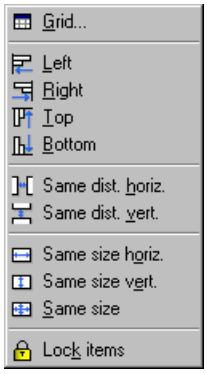
This option displays the list of manual areas defined for the entity in use.

4.5.6 Align

The *Align* menu allows you to position and resize groups of selected elements.

Alignment and re-sizing functionalities are made in relation to a master item. For more information please refer to section 'Selecting And Aligning Elements'.

Picture 121 - Align Menu



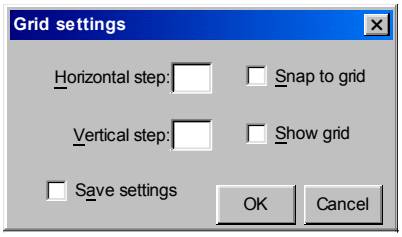
The Align menu allows you to:

- Open the grid management
- Align to the right/left/top/bottom
- Position with the same horizontal/vertical distance.
- Re-size with the same height and width.
- Block changes for manual positioning and resizing.

Grid...

Opens the Grid Setting window to manage the grid:

Picture 122 - Grid Setting



Horizontal Step

Defines the horizontal size of the cell in pixels.

Vertical Step

Defines the vertical size of the cell in pixels.

Snap to grid

Activates the grid to position elements.

Show grid

Displays the grid.

Save setting

Saves the current setting of the grid.

Left

Aligns the left side of selected elements with the left side of the master item.

Right

Aligns the right side of selected elements with the right side of the master item.

Top

Aligns the top of selected elements with the top of the master item.

Bottom

Aligns the bottom of selected elements with the bottom of the master item.

Same dist. horiz.

Selected elements are positioned with the same horizontal distance. The distance is measured between the first and second element starting from the left.

Same dist. vert.

Selected elements are positioned with the same vertical distance. The distance is measured between the first and second element starting from the top.

Same size horiz.

Resizes the width of selected elements like the master item.

Same size vert.

Resizes the height of selected elements like the master item.

Same size

Resizes selected elements like the master item.

Lock items

Inhibits selection and change commands for selected items.

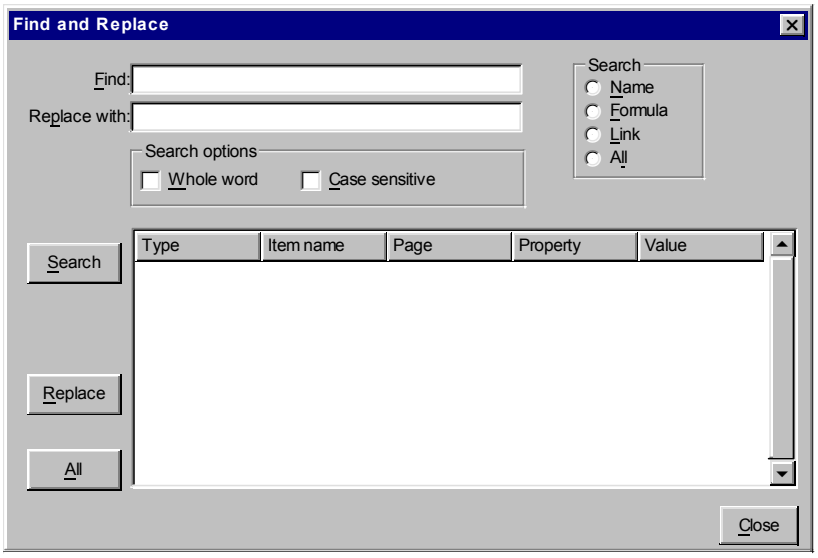
Warning

The commands are inhibited for the mouse and the keyboard while the Align toolbar remains active.

4.6 Edit Menu Advanced Options

This section details the Edit menu functions.

4.6.1 Find And Replace



Picture 123 - Find and Replace

Find

String that must be found.

Replace With

String that must replace the found string.

Whole Word

When this field is flagged the search activity must be performed on whole words only. Otherwise search results could be also substrings of long words.

Case Sensitive

The search activity matches upper and lower Case.

Search Button

Starts the search activity:

Name:

the search activity is perfomed on element names

Formula:

the search activity is perfomed on defined formulas.

Link:

the search activity is perfomed on defined links.

All:

the search activity is perfomed on names, formulas and links.

List Of Found Items

Search result listing elements found.

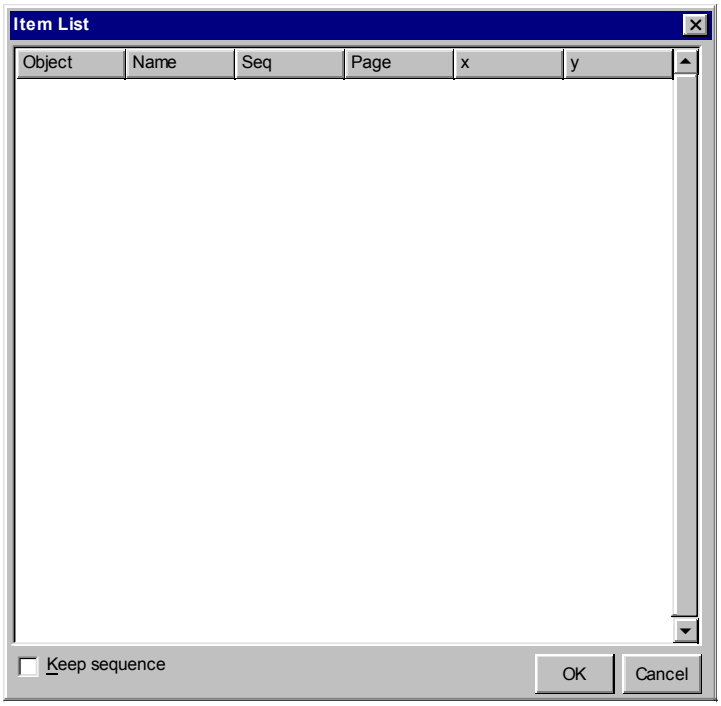
Replace Button

Replaces strings found with the string defined in the textbox 'Replace with' for all elements selected in the list (List of found items).

All Button

Replaces strings found with the string defined in the textbox 'Replace with' for all elements in the list (List of found items) no matter if elements are selected or not.

4.6.2 Item List



Picture 124 - Item List

Elements List

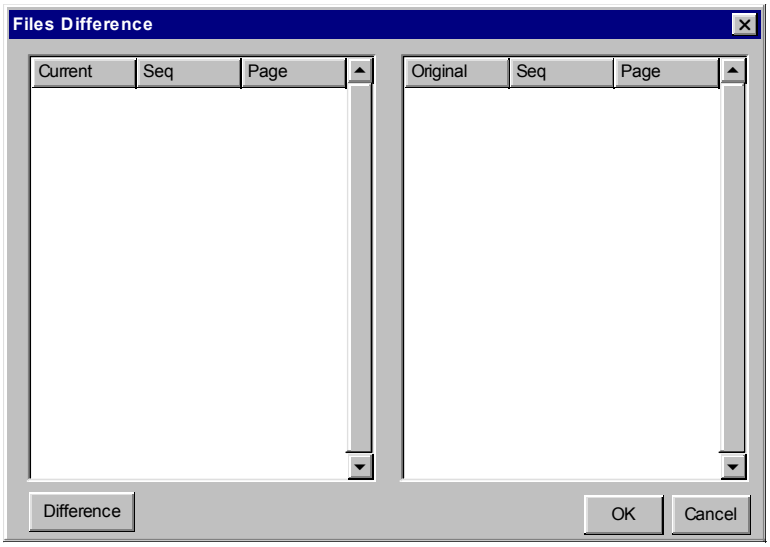
List of elements in the dialog window. This list allows to access elements defined for this entity quickly.

Keep sequence

Flag to maintain the elements' sequence. When the flag is set and the dialog window saved elements are ordered to reflect the sequence defined in the elements' list. To order single elements right click the desired element and select the 'Sequence' option.

4.6.3 Files Difference

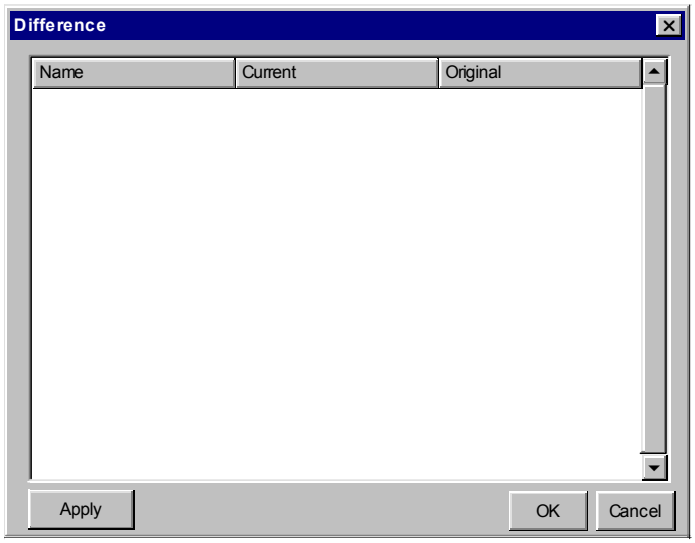
Picture 125 - Files Difference



Difference Button

To compare differences between entities select two entities and open the 'Master File Differences' option.

4.6.4 Difference



Picture 126 - Difference

Difference List

List of differences between two definition files.

Apply Button

The selected difference is applied to the current definition file.

4.6.5 Design Compare

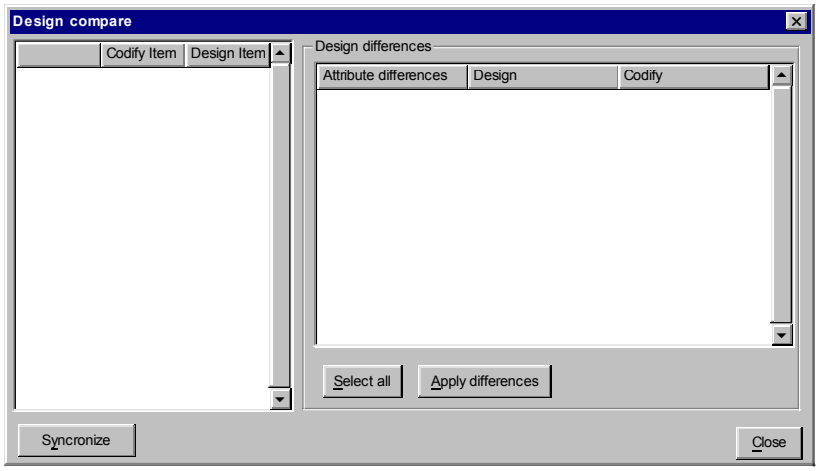
Comparing the Codify phase with the design plan allows you to synchronize entity definitions. It can be done only when the 'Bind to design entity' option in the 'Tables' or 'File and Keys' menu is set.

Once opened the dialog window displays all differences between the properties of design and codify entities, e.g. discrepancies between comments, or fields in a link, or missing fields, i.e. those fields defined in the design plan and not taken over to the Codify phase.

The comparison starts from the design entity. This involves that 'non-design' properties, which have been added during the Codify phase are not listed.

Using the 'Synchronize' or 'Apply differences' button you can synchronize some or all differences identified in the Codify phase.

Picture 127 - Design Compare



Elements List

List of elements for which the comparison has been run. The icon in the first column shows whether differences have been detected or not.

Design differences

List of differences for the element selected from the elements list. The first column briefly describes the attribute for which the difference has been identified. The second column displays the attribute value defined in the design plan. The third column displays the value defined in the Codify phase.

Select all

Selects all elements in the differences list.

Apply differences

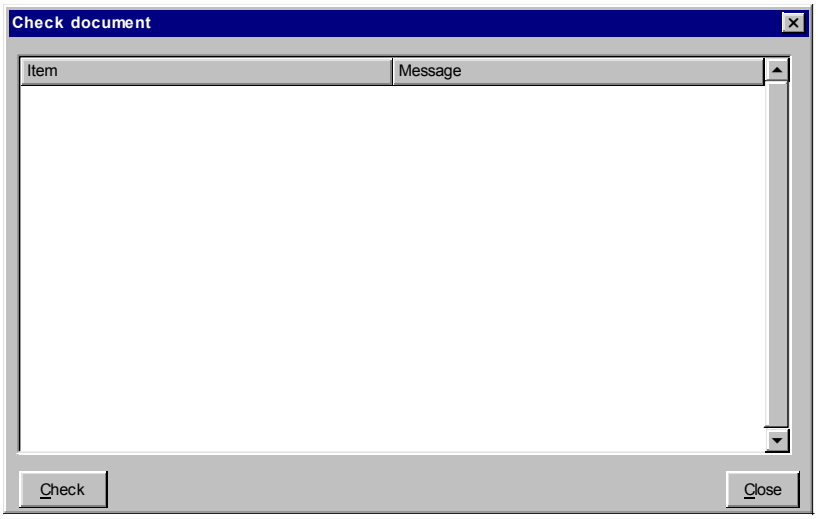
Synchronizes codify attributes for selected elements basing on the design value.

Synchronize

Synchronizes codify attributes for all elements basing on the design value.

4.6.6 Check Document

The Check dialog window executes congruence validations on definitions added using the Codify tool. CodePainter allows you to override values that usually are considered wrong, but may be required to manage specific issues. The Check dialog windows checks that no contrasting situations have been defined that could lead to malfunctions at run-time.



Picture 128 - Check Document

Errors List

Error list.

Check

Starts the validation of added definitions.

4.7 Items Menu Advanced Options

This section explains the Items menu options.

The Items menu options can be accessed opening the 'Items' menu, or from the 'Painter Tools' toolbar or right clicking in the working area. These options allow you to add new elements to the current entity.

Let's now see the various options in detail.

N. B.

Options for the elements 'Field' and 'Variable' are the same. During the Codify Phase and in the generated code a working variable having the prefix 'w_' is generated for each field. The table is changed and the field values stored only when the record is saved. For Detail and Master/Detail entities you cannot have a working variable for each field. For these entities a support file is used in which is up and downloaded when records are read/ saved. Working Variables for repeated fields take on values when the row is changed. This happens by reading values from the support table and saving them when the row is changed.

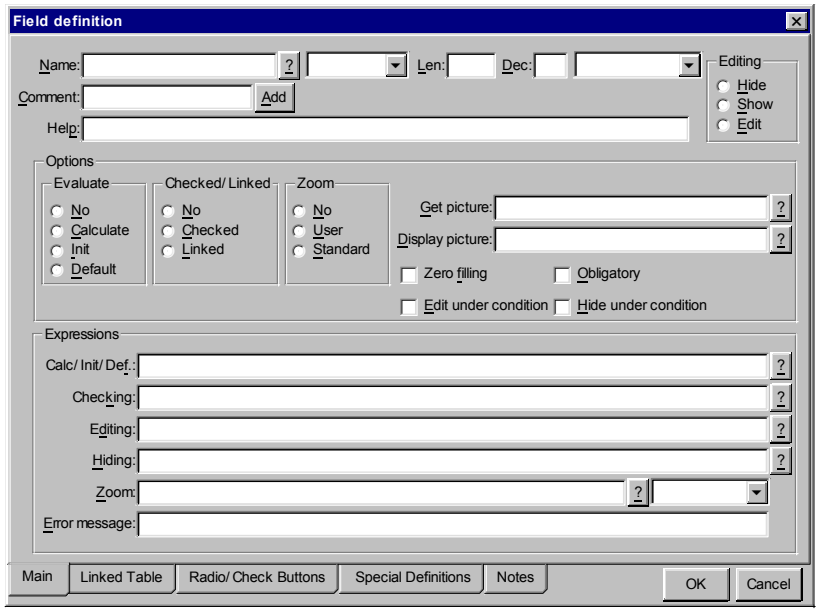
4.7.1 Field Definition

The definition window allows to define properties for fields or variables. Here you can define how elements are displayed, or define control and calculation formulas, or mandatory fields, or whether to associate a checkbox, radio or combobox, etc.

Main Page

The first page defines the main element's properties. The window has three sections: the first groups the element's characteristics, the second activates controls or calculations and in the third expressions to execute controls are defined.

Picture 129 - Field Definition: Page 1



Name

Name of the field or variable.

This name must follow the rules for the construction of variable identifiers for the target environment. Typically it must start with a letter and cannot have blanks, commas or points. Some environments also have length limitations.

CodePainter creates for each element (fields and variables) a working variable that takes on the prefix 'w_'. Fields are initializated when the record is read. The user edits the record using the values in the working variables. When the record is saved the contents of 'w_' variables is saved in the corresponding database fields.

Variables that do not have values stored in the database are initializated when the record is loaded. The working variable either does not contain any value (i.e. blank for strings, or 0 for numbers), or it contains the value defined in the 'Init' expression. During the editing phase they behave as fields as defined above.

When the selected element is a field you can list all fields of the associated table clicking the '?' button. In the same way when the element is a variable you can list all variables that CodePainter identifies as useable, but have not yet been placed on the screen.

Field Type

Kind of element.

This combobox defines the kind of element you are editing. Implemented element types are those typically used by business/ commercial applications, i.e. character, numeric, date, logical, memo. The element type here must match with the associated database field. You can define different types only if the type is compatible with what is read by the files: shorter strings, numbers having less digits, etc.

Len

Element's length.

Here you define the length of the variable associated to the element. The length is expressed in characters for strings and in digits for numbers. For logical elements, dates and memos the default length is used.

Dec

Decimal digits for the element.

For numeric elements you can define the number of decimals required. This number is included in the total length.

Key/Index

Defines whether an elements is part of a key or of a search index.

Elements belonging to the primary key can be edited only when a new record is created. They cannot be edited in the change mode.

In the query mode all elements belonging to primary keys or indexes are enabled. When the user enters a value the search activity is quick, because it is executed on the index and not systematically scanning the entire file.

Normally this option is defined for fields and not for values.

Comment

Brief comment on the element.

This comment is displayed in the element list of the edited entity. Fields are automatically initializated using the description defined in the design plan. For variables this must be set by the programmer.

Using the 'Add' button you can add a string next to the element and define the text and comment. This allows you to obtain a prototype for working variables quickly, which is in line with the dialog window.

Add

Using the 'Add' button you can add a string next to the element and define the text and comment. This allows you to obtain a prototype for working variables quickly, which is in line with the dialog window.

Editing

Editing status of the element.

Elements have three editing statuses:

Hide

Hidden elements are not displayed. They are used for calculations or as supporting variables.

Show

Showed elements are displayed on the screen but cannot be edited. They are used in links as return variables, to store calculation results or as support variable that must be checked by the user.

Edit

Editable elements can be changed by the user.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Evaluate

Defines whether the element is calculated.

Elements can be edited by the user or calculated by the procedure.When you define options involving calculations you need to define the expression in the 'Calc/Init/Def.' option in the 'Expressions' area. There are many calculation options:

'No'

When it is set the element is not calculated. Variables are initialized (blank string for characters, 0 for numbers, etc.).

'Calculate':

The element is calculated. The defined expression is checked every time a value affecting the calculation result is entered in the dialog window. When the calculation is simple this strategy works smoothly. When the calculation implies a slow routine you need to limit re-calculations. In the 'Calc/Link depends on' option you need to define which elements trigger the re-calculation. The calculation will be executed only when one of these elements changes. Calculated elements are normally 'shown' and not 'editable'. To make elements editable the calculation must not be re-executed at each input, but only when trigger elements are changed. Otherwise the calculation is re-executed also when a value is entered, thus overriding the input.

'Init':

The calculation is executed only once when the element is edited. This option is used to define a different default value then the one given automatically by the system.

'Default':

In some cases fields cannot be initializated together with the dialog window. This may happen because the field initialization depends upon some values that the user must input first.When the 'Default' option is set the calculation is executed the first time the cursor enters the element. In the initialization expression you can therefore define all elements that the user must enter first.

'Totalize':

When this option is set the elements sums up all values in the body rows. In the calculation expression you need to define the name of the element that must be totalized.

Checked/Linked

Defines whether the element is controlled or linked to a file.

To control the data that the user has input you need to define one of the options in the table below. If you select an option that requires a control you need to define the validation formula in the 'Checking' line in the 'Expressions' area.

'No':

The element is not controlled. Any value is accepted.

'Checked':

Any user input is controlled executing the 'Checking' formula. When the result is TRUE the value is accepted, otherwise an error message is displayed, the element is emptied and the cursor positioned on the element. The standard error message can be changed defining the 'Error message' expression.

'Linked':

The control activity involves a database table. The input value is searched in the database and is accepted only if the linked table contains a record that has the value as primary key. Detail on how this search activity is performed are defined in the 'Linked Table' page. 'Linked' elements can be also controlled by a 'Checking' formula. The value is first searched on the linked table. If found and returned the 'Checking' formula is executed.

Zoom

Defines whether there is an active zoom on the element.

Zooms are procedures run when the user presses the function key F9 when the cursor is on the related element.

'No'

No action is performed.

'User'

The expression defined in the 'Zoom' formula in the 'Expressions' area is executed.

'Standard'

When an element is liked to a table CodePainter creates a standard zoom to

display and search data in the liked table. Parameters defined in the 'Linked Table' page influence the standard zoom. The 'Zoom' expression is used and a specific zoom configuration executed.

Get picture

Format of the input element.

In this option you can define the format that must be applied when the element is inserted.

You need to define an expression that is checked at run-time. Constant values must be defined between apexes.

Typical formats are:

"999.99": for numeric values.

"XXXXX": for characters.

Repl("!", 20): forces upper case in 20 characters strings.

When this option is not defined and you define a display format the input uses the latter. The '?' button creates a standard format for the element type.

Display picture

Format of the displayed element.

In this option you can define a format to be applied when the value is displayed. You need to define an expression that is checked at run-time. Constant values must be defined between apexes.

Typical formats are:

"999.99": for numeric values.

"XXXXX": for characters.

Repl("!", 20): forces upper case in 20 characters strings.

When no input format is defined the format defined for display is used also when data is input.

Zero filling

The value takes on zeros on its left until the value is filled up.

When the 'Autonumber' option is used character keys entries are created as '00001', '00002' etc. The Zero filling option makes it easier for the user who is not required to manually fill the field with zeros. The system automatically adds zeros until the field length is reached.

This option works only is the entered value is numeric and does not start with '0'.

Obligatory

Mandatory element.

An editing phase cannot be terminated as long as no value has been defined for this element. When the user presses the function key F10 and no value has been entered an error message 'Obligatory field' is displayed. The cursor is positioned on the element and the editing mode is kept so that the user can enter a value.

Edit under condition

Defines whether the element can be edited only under specific conditions.

Elements may be editable only when specific conditions occur. To define conditions set the 'Edit' option to edit and define the 'Editing' formula in the 'Expressions' area that must be checked before the cursor is positioned on the element. When the logical expression returns a TRUE value the element is edited. Otherwise the cursor is passed on to the next element.

Hide under condition

Hides elements.

Elements may be hidden under specific conditions. To define these conditions set the 'Edit' or 'Show' option and define the 'Hiding' formula in the 'Expressions' area that must be checked. When the logical expression returns a TRUE value the element is hidden and therefore not displayed on the dialog window.

Calc/Init/Def.

Formula used for calculations.

When calculations formula are defined in the 'Evaluate' option the expression is used to calculate the element's value.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Simple Calculation

Define three elements: PRZART, QTAART and TOTART. The total (TOTART) must be calculated multiplying price and quantity.

The calculation formula must be defined as follows:

```
w_PRZART * w_QTAART
```

Editable Calculation

Consider the same three fields: PRZART, QTAART and TOTART. The total (TOTART) must be calculated and the element must be editable so that the user can adjust roundings manually.

The calculation formula must be defined as follows:

```
w_PRZART * w_QTAART
```

Further, in the 'Calc/Link depends on' option you need to define:

```
w_PRZART
```

```
w_QTAART
```

The calculation is executed only when the value for one of these fields changes.

Checking

Control formula for the element.

When elements are controlled or linked this expression is used to check whether values are acceptable or not. When the logical expression returns a TRUE value the value is accepted. When the returned value is FALSE the value is refused.

When the value is refused the default value is input in the element depending on the element type, an error message is displayed and the cursor passes on to the next element. The default message is 'Value not accepted', but you can change it defining the desired formula in 'Error message'.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Controlled Field

```
The field PRZART accepts only values greater or equal to zero.:
w_PRZART>=0
```

Editing

Formula to activate the conditional editing.

This formula defines whether the field is editable or not. To define conditional editing you need to define the 'Edit' option in the 'Edit' area, set the 'Edit under condition' checkbox, and define the 'Editing' formula in the 'Expressions' area. When the logical expression returns a TRUE value the element is edited. When the returned value is FALSE the fields is disenable.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Hiding

Formula to hide the element.

This formula defines whether the field must be hidden or not. To define these conditions set the 'Edit' or 'Show' option, set the 'Edit under condition' checkbox and define the 'Hiding' formula in the 'Expressions' area. When the logical expression returns a TRUE value the field is hidden. When the returned value is FALSE the field is displayed.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Zoom

Zoom formula.

When elements have an active zoom this formula is used to define what must be executed when the user presses the function key F9.

When the 'Zoom' option is set to 'User' in this formula you need to define what must be executed. The combobox next the option define the type of zoom. Zoom types are:

'User program': the formula is a program line.

'Mask': the formula defines the dialog window that must be executed as zoom.

'Batch': the formula defines a routine created using the Routine Painter.

'UTK Object': the formula contains the output configuration name that must be displayed as zoom.

When the 'Zoom' option defines a 'Standard' zoom the formula can contain the name of a specific configuration that must be loaded in the zoom window.

Error message

Text for the error message.

When the element is controlled the user could enter values that are refused. In these cases the error message 'Value is not correct' is displayed. In this option you can define custom error messages for each element.

Checked/Linked

Defines whether the element is controlled or linked to a file.

To control the data that the user has input you need to define one of the options in the table below. If you select an option that requires a control you need to define the validation formula in the 'Checking' line in the 'Expressions' area.

'No':

The element is not controlled. Any value is accepted.

'Checked':

Any user input is controlled executing the 'Checking' formula. When the result is TRUE the value is accepted, otherwise an error message is displayed, the element is emptied and the cursor positioned on the element. The standard error message can be changed defining the 'Error message' expression.

'Linked'

The control involves a database file. The input value is searched in the database and is accepted only if the linked table contains a record that has the value as primary key. Detail on how this search activity is performed are defined in the 'Linked Table' page. 'Linked' elements can be also controlled by a 'Checking' formula. The value is first searched on the linked table. If found and returned the 'Checking' formula is executed.

Evaluate

Defines whether the element is calculated or not.

Elements can be edited by the user or calculated by the program. When the calculation option is selected you need to define the expression that produces the result in the 'Calc/Init/Def.' option in the 'Expressions' area. There are many calculation options:

'No'

When 'No' is set the element is not calculated. For variables the blank value is initializated (blank string for characters, 0 for numbers, etc.).

'Calculate':

The element is calculated. The defined expression is checked every time a value affecting the calculation result is entered in the dialog window. When the calculation is simple this strategy works smoothly. When the calculation implies a slow routine you need to limit re-calculations. In the 'Calc/Link depends on' option you need to define which elements trigger the re-calculation. The calculation will be executed only when one of these elements changes. Calculated elements are normally 'shown' and not 'editable'. To make elements editable the calculation must not be re-executed at each input, but only when trigger elements are changed. Otherwise the calculation is re-executed also when a value is entered, thus overriding the input.

'Init':

The calculation is executed only once when the element is edited. This option is used to define a different default value then the one given automatically by the system.

'Default':

In some cases fields cannot be initializated together with the dialog window. This may happen because the field initialization depends upon some values that the user must input first.When the 'Default' option is set the calculation is executed the first time the cursor enters the element. In the initialization expression you can therefore define all elements that the user must enter first.

Linked Table Page

In the second page you need to define the rules for the relationship between an element and a file record. The prototype defaults automatically what you defined in the Design phase.

When you define links in the prototype without going back to the design plan the relationship is not programmed in the database. This means that referential integrity for this relationship is not established. Values are checked only when they are entered in the dialog window.

Picture 130 - Field Definition: Page 2

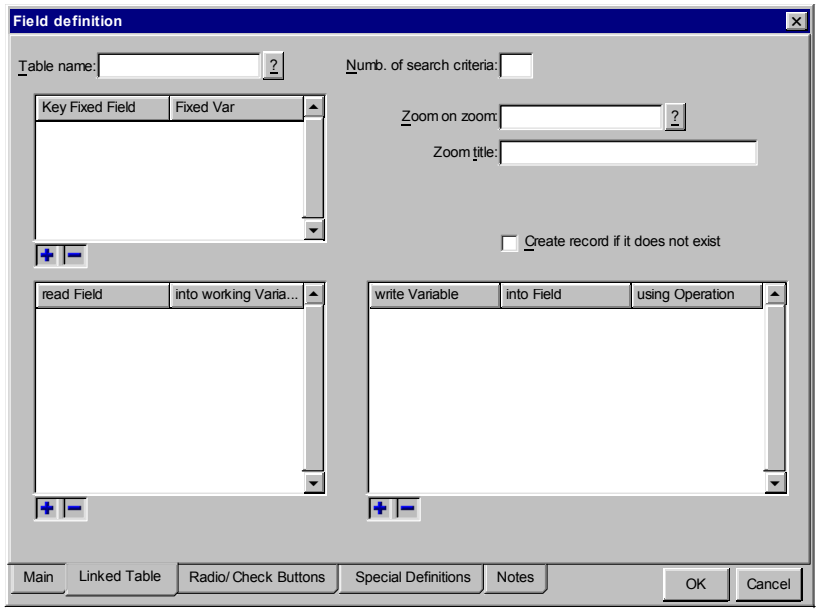


Table name

Name of the related file.

Define the name of the related table. The value is searched in the related table basing on the parameters defined in this dialog window.

The table name is added to the workfiles list so that when the program is run the system checks the existence of the related table. Once you enter the table name, all lists helping the programmer to select fields' names are activated. Moreover, the procedure name that must be activated for the 'zoom on zoom' option is initialized. This procedure is a zoom that can be executed pressing F9 within an active zoom.

The '?' button displays the list of tables available in the database.

Numb. of search criteria

Defines the number of search criteria.

Using this option you can change the way records are searched in the related table. SQL sentences use fields defined in the 'Read Field ... into working Variables' area as search criteria. If the first search goes wrong the SQL sentence uses the second field in the list and so forth as long as there are search criteria.

Define an element 'ARTORD' which is related to the 'Items' file. Your search criteria is CODART and you need to return the field DESART.The standard search criteria is:

```
CODART=w_ARTORD
```

If you define the 'Read Field' list as follows:

```
CODART, w_ARTORD


DESART, e_DA_ORD
```

and the first search criteria does not lead to any result, no error message is displayed and the search activity is executed:

```
DESART=w_ARTORD
```

Only if the second search goes wrong as well the defined value is refused.

The example highlights how the current fields is searched as code first and as description afterwards. If the result is given by the second search criteria values are returned in the defined sequence.

This kind of alternative search is performed for the number of defined criteria, following the list of fields that must be returned. Fields must always be of the same type, you cannot mix e.g. numbers and characters.

Fixed fields belonging to the key are used also used for alternative searches.

Zoom on zoom

Procedure which is executed pressing F9 on an active zoom.

When the user presses F9 a zoom window on the related table is opened. If searched values are not found pressing F9 again (on the active zoom) a new window is opened that allows managing the linked file.

This option allows to define the procedure that must be executed to zoom on an active zoom. Normally the standard procedure to manage the related table is defined. This means that when you select the file reading design definitions, this field is initializated by CodePainter.

Zoom title

Contains the title of the zoom window opened.

Create record if it does not exist

Defines whether a record must be created on the related table.

Linked fields must find a related value in the linked table. This is true for all elements input by the user, because they are validated and not found values are refused.

When linked elements are hidden values are never entered and thus never checked. For example an element may be calculated basing on another element on the screen but linked to another table. It is therefore possible to write totals in a record that does not exits, i.e. the related record is missing.

Using this option the search activity can go on. A new record is created and filled in the related table using the list of fixed key of read fields and totals. Returned values are used to attribute values to fields.

This happens especially when you have Parent/Child relationships whereby totals are required to update the Child entity.

Key Fixed Field

Fields building the primary key, excluding the current field.

This list is used when the primary key of the related table is composite.

The search activity is started when the user enters a value in the current field, which in turn is placed in line with the first element in the list right under the value. Primary key fields are placed next to the corresponding values in the dialog window according to the logic defined in this list. For each field there is value, which has been read by a working variable.

Composite Key

The primary key of the linked table is made of the fields: CODMAG and CODART. In the Dialog Window there are two elements, MAGORD and ARTORD asking the user to input the warehouse code and the item code on which he/she wants to work.

In the ARTORD element you need to define:

```
CODMAG, w_MAGORD
```

In 'read Fields' define:

```
CODART, w_ARTORD
```

The search activity in the related table will have matches of the following kind:

```
MAGART=w_MAGORD and CODART=w_ARTORD
```

The standard zoom uses these fields as filter on the table so that only a selection of the table is displayed and not the entire related table. In the example above after that the user enters a value for the warehouse the zoom window will display only the items in that warehouse.

read Field

List of fields that must be read from the related table.

This list drives the relationship with the related table. When the user enters a value in the current element the search activity is started. The selection criteria is given by the first field in this list and the fixed fields defined in the 'Key Fixed Fields' area (optional).

Example

If the related file has the key CODART and the current element is ARTORD the search activity is as follows:

```
CODART=w_ARTORD
```

If the entered value is found the link is successful and all fields defined in the list are read and values are returned to the corresponding working variables.

If the value is not in the file the error message 'Value is not correct' is displayed and the element is reset.

The search activity can also find partial values, e.g. the user in enters 'Smith' and the value in the file is 'Smith John'. In these cases the value entered in the field is completed with the value found in the table.

When the search activity finds more values (e.g. the user enters 'Smith' and the values in the table are 'Smith John' and 'Smith Bob') a zoom window containing the list of matches found is opened and the user can select the desired record.

When the search activity goes wrong but more search criteria are defined, alternative searches are executed as defined in 'Numb. of search criteria'.

Write Variable

This list defines totalization transactions towards the related table.

This list is made of three columns: the first column defines the variables that will be summed up, the second the fields that will be increased in value and the third the operations that must be performed.

In the third column you can define the constants '+', '-', and '=' to get totals, reversals or a data entry. You can also define fields with one character that contain the transaction that must be executed. When this value is blank no transaction is performed. Transaction types must be stored in a field and not in a variable because CodePainter requires this information to perform reversals in case of changs or deletions.

Example

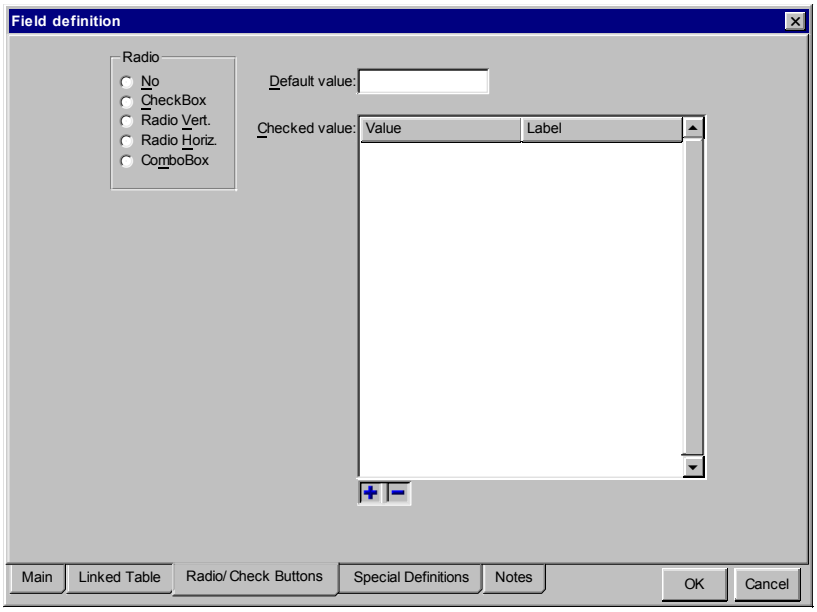
Your application has two tables, namely 'Items' with its fields CODART, DESART and QTAART, and 'Orders' with the fields ARTORD and QTAORD.

You want to sum ordered quantities in QTAART. Define the 'Write Variable' list for the element ARTORD as follows:

```
w_QTAORD, ARTORD, +
```

Radio/Check Buttons Page

The third page contains parameters to change the standard input/ output textbox in other input/ output structures.



Picture 131 - Field Definition: Page 3

Radio

To display the element as checkbox, radio or combobox.

Fields and variables are created as textboxes. This option allows you to change the method in which data is entered and displayed.

Default value

Defaulted value when the user does not select any value.

Value

List of variables and labels.

This list is made of two columns: the first defines the value which is given to the element, the second the label that must be displayed.

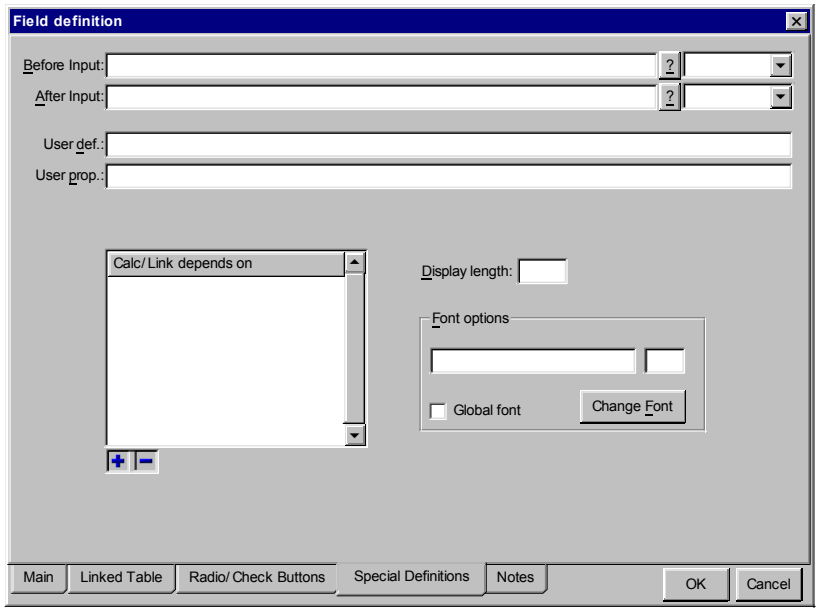
Values must be defined as constants in the target language. Strings must therefore be written between apexes. Labels are always text constants therefore they do not need to be written between apexes.

Special Definitions Page

The fourth page defines properties, which are used seldom.

Rather then using 'Before' and 'After' it is advisable to use the structured options in the first page. 'User Def' require customized templates. 'Depends on' are used for optimizations.

Picture 132 - Field Definition: Page 4



Before input

Program that must be executed before the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low-level and cannot be defined so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

After input

Program that must be executed after the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low-level and cannot be defines so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

User def.

Free definition.

Could be used by a custom template.

User prop.

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to add definitions to the ones generated by CodePainter.

Example

```
forecolor=RGB(255, 0, 0)
```

Calc/Link depends on

List defining the elements on which the calculation execution or the link executions depends on.

Calculations and links are normally executed every time a value is entered. This strategy works smoothly if the calling routine is quick. If calling routines take long to be terminated you need to limit the number of calls.

Defining working variables in this list, calculations and associated links are executed only when the value for one of these variables changes.

This list is also used for calculated elements, which can be edited. Elements will remain editable, but they will be recalculated only when the value for one of these variables changes. Here you should define all calculation parameters associated to the element.

Using this method you avoid executing many line codes. Nevertheless checking the change of the value is costly and therefore this option should be activated only when calculations are complex.

The same applies to link executions.

Example

A field is linked to the 'Items' file. The field is returned to the linked 'VAT' file. When item's data is entered the VAT value must be defaulted. The field must be editable so that the VAT value can be returned by the first link.

You need to define the 'item number' field in the 'Depends on' list of the 'VAT' field. When the user selects an item the link with the VAT file is re-executed. The VAT value can be freely changed. If the user goes back to the 'item number' field and enters a new value, the link to the VAT file is re-executed.

Display length

Length used to display the element.

This option is used so that CodePainter calculates a length, which is different from the one given by the element parameters.

Change Font

Defines the elements font. If no font is defined the dialog widow's default font is used.

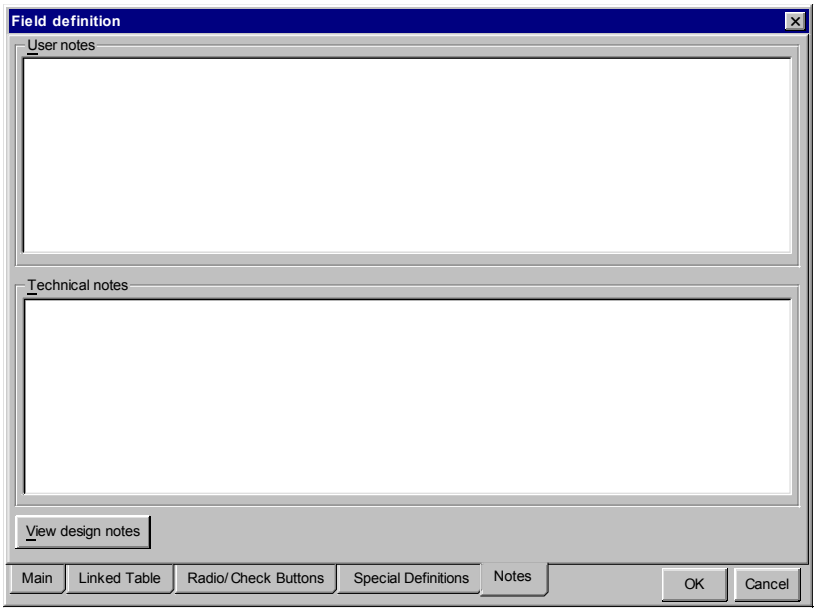
Global font

When selected the font defined in the 'Global Font' menu is applied to the current element.

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.



Picture 133 - Field Definition: Page 5

User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

Technical notes on the element.

These notes are used to create technical documentation.

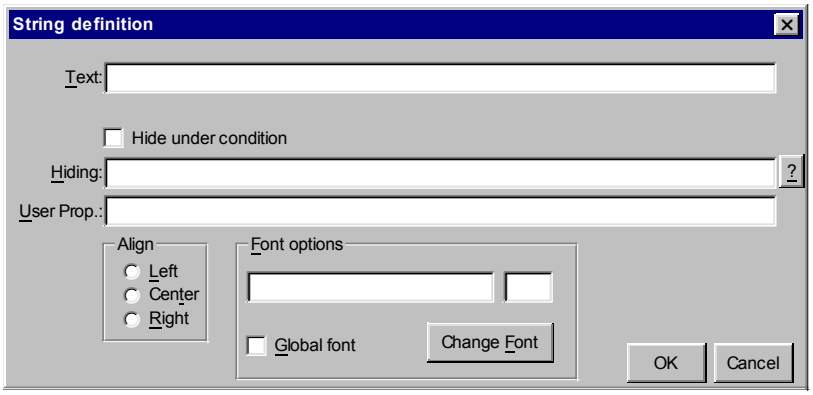
View design notes

Quick access to the Design notes of the current element.

4.7.2 String Definition

Dialog window to define the characteristics of comment strings.

Picture 134 - String Definition



Text

String text.

Hide under condition

Hides the string.

To display strings only when specific conditions are met you need to define the hiding formula in the 'Hiding' textbox. When the returned value is TRUE the string is hidden, and is not displayed.

Hiding

Formula defining the conditions to hide the string.

This formula defines whether the string must be hidden or not. To hide the string under given conditions you need to set the checkbox 'Hide under condition' and to define a logical expression in this field. When the result is TRUE the string is hidden, when the result is FALSE the string is displayed.

Clicking the '?' button the list of elements in the dialog window is displayed. These elements can be used to define expressions. Please note that while in the Editing mode CodePainter stores values in working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

User prop

Control properties defined by the programmer.

Elements on the screen are generated as a 'control'. This option allows you to add other specific controls.

Example

```
forecolor=RGB(255,0,0)
```

Align

String alignment.

Defines how the string must be positioned within the string box.

'Left'

Aligned to the left.

'Center'

Centered.

'Right'

Aligned to the right.

To set the window layout you should align strings to the right. You should not align to the left and create boxes that exactly fit the strings. The end-user may use a different font from the one defined and therefore strings may be bigger or smaller. In these cases strings would be no longer aligned.

When you need to translate the application you need to align strings to the right as words in other languages may be longer or shorter. Using CodePainter you can automatically translate dialog windows and set a different language for each user.

Prototypes are created using these principles, i.e. alignment to the right and field space longer than required.

Change Font

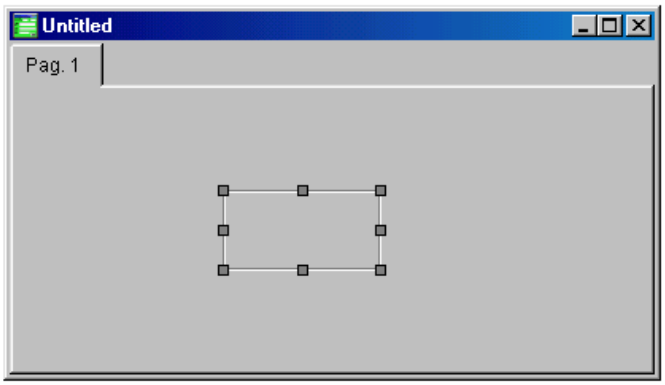
Defines the strings' font. If no font is defined the dialog widow's default font is used.

Global font

When selected the font defined in the 'Global Font' menu is applied to the current element.

4.7.3 Box

This option allows you to add a Box element to the entity in use.



Picture 135 - Box

Boxes are used to graphically divide the form. Boxes can be used as vertical or horizontal lines or rectangles.

Boxes are changed into lines '*collapsing*' them using the mouse or **<Shift>** + **<Cursor Keys>**.

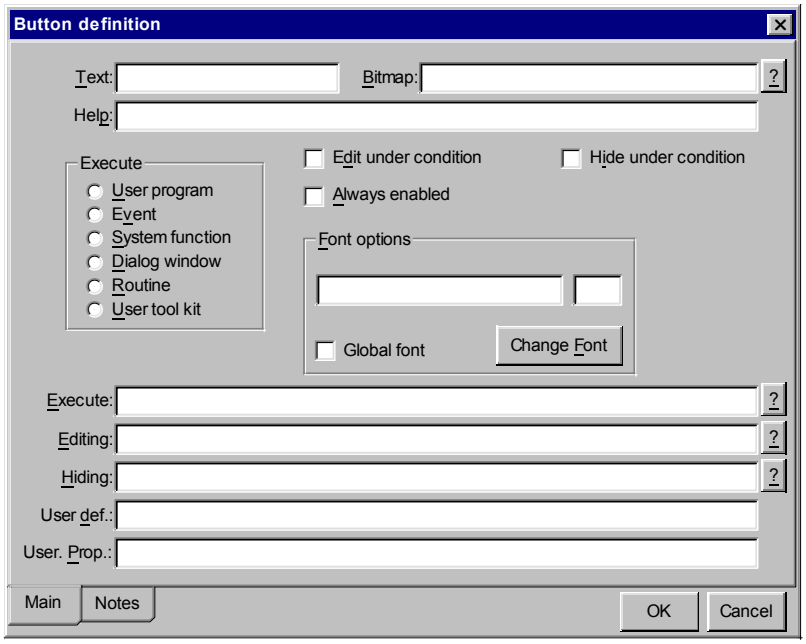
4.7.4 Button Definition

Dialog window defining button options. Buttons are added into dialog windows to execute procedures such as reports, additional dialog windows, etc.

Main Page

In the first page you can define how the button is displayed, activated and its functions.

Picture 136 - Button Definition: Page 1



Text

Text that must be displayed inside the button.

Bitmap

Bitmap displayed inside the button.

Help

Short 'Help' text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Execute

Defines how CodePainter must interpret the execution command line when the button is clicked.

'User program'

Executes a program defined by the developer.

'Event':

Notifies an event.

'System function':

Executes a system function.

'Dialog window':

Opens a dialog window created using the Dialog Window Painter.

'Routine':

Executes a routine developed using the Routine Painter.

'User Tool kit':

Executes an output created using the run-time framework.

Depending on the command type defined in 'Execute' different values are required. 'User Program' is copied in the source. 'Event' requires the name of the event that must be notified. 'System function' requires the name of the function that must be activated. 'User tool kit' requires the name of the query followed by the output format name. In all other cases the name of the file that must be executed is required.

Edit under condition

Defines whether the button is enabled only under specific conditions.

To enable buttons only when specific conditions are met you need to define a formula that must be validated before the cursor is positioned on the element.

Setting this option the 'Editing' formula in the textbox is activated. The formula must be a logical expression. When the returned value is TRUE the button will be enabled, otherwise the cursor is passed on to the next element.

Hide under condition

Hides the button.

To hide buttons under specific conditions you need to define the 'Hiding' formula in the textbox. The formula will determine whether the button must be displayed or hidden. If the logical value TRUE is returned the element is hidden otherwise the button is displayed.

Always enabled

The button is enabled in 'Editing' or 'Query' modes.

Buttons are used to access procedures that are bound to the main dialog window. Using these procedures may make sense in the 'Editing' and/or in the 'Query' mode only. E.g. a button executing a report should always be enabled.

Setting this flag the button will be enabled in the 'Editing' as well ad in the 'Query' mode. When the flag is not set the button is enabled in the 'Editing' mode only.

Execute

Defines what must be executed when the button is pressed.

The meaning of this line depends on what has been defined for the radio button 'Execute'. The button on the left allows quick access to the following definitions:

'User program':

The code line is copied into the source. Using the '?' button you can access the list of variables contained in the dialog window.

'Event':

Defines the name of the Event that must be notified.

'System function':

Executes a CodePainter function. Clicking the '?' button the list of available functions is displayed.

'Dialog window':

Opens a dialog window created using the Dialog Window Painter. The '?' button lists all Dialog Windows available in the project.

'Routine':

Executes a routine developed using the Routine Painter. Beside the routine name you can also define in brackets the parameters that must be passed on to the routine.

'User Tool Kit':

Executes an output created using the run-time framework, typically a query created using the Query Painter. A dialog window is opened in which you can select the kind of output required: preview, report, word, excel or graph. To execute the output you need to add a comma after the query name and digit the output name.

Editing

Formula to set conditions to allow 'Edit' mode.

This formula defines whether the button is active or not. To define conditions you need to set the 'Edit under condition' option and define a logical expression. When the expression result is TRUE the button is enabled. When the result is FALSE the button is disenabled.

Clicking the '?' button the list of all elements in the dialog window is displayed. These elements can be used to define the expression. Please note that during the Editing mode CodePainter stores the values of working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

Hiding

Formula to hide the button.

This formula defines whether the button must be hidden or not. To hide the button you need to set the checkbox 'Hide under condition' and to define a logical expression in the formula. When the result is TRUE the button is hidden, when the result is FALSE the button is displayed.

Clicking the '?' button the list elements in the dialog window is displayed. These elements can be used to define the expression. Please note that while the Editing mode CodePainter stores the values of working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

User def

Free definition.

This option can be used by custom templates.

User prop

Control properties defined by the programmer.

Elements on the screen are generated as a 'control'. This option allows you to add other specific controls.

Example

```
forecolor=RGB(255, 0, 0)
```

Change Font

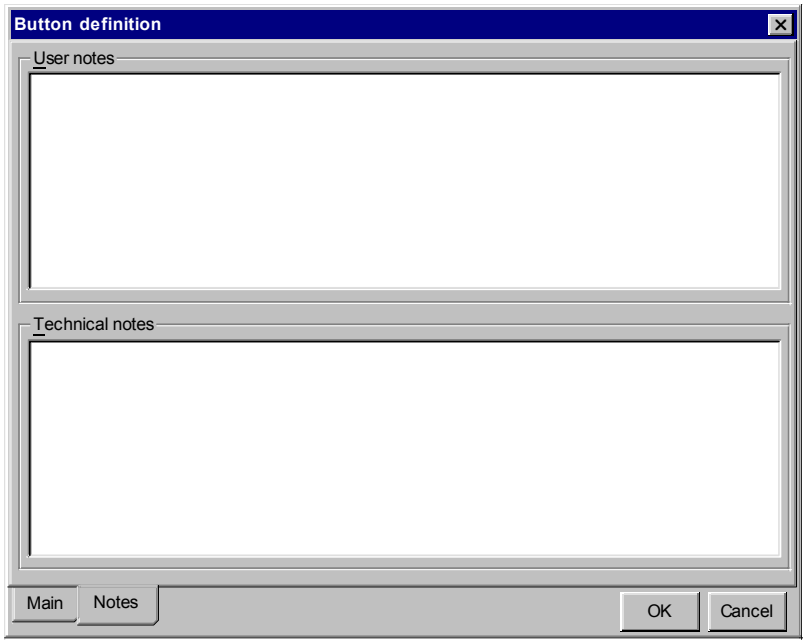
Select the font to be applied to this element. When no font is defined the default font used by the dialog window is applied.

Global font

When selected the font defined in the 'Global Font' menu is applied to the current element.

Notes Page

Notes that are used to create the documentation.



Picture 137 - Button Definition: Page 2

User Notes

Notes on the element.These notes are used to create user documentation.

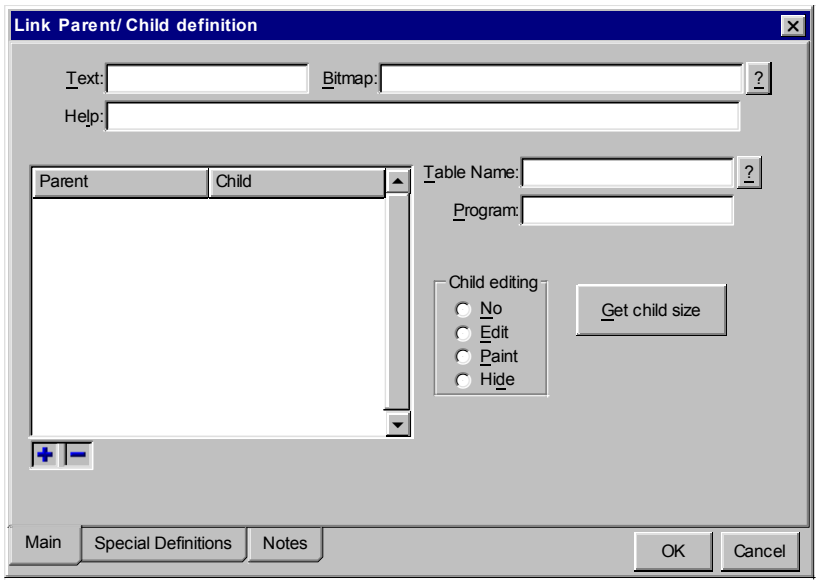
Technical Notes

Technical notes on the element.These notes are used to create technical documentation.

4.7.5 Parent/Child Link Definition

Main Page

Picture 138 - Parent/Children Link Definition: Page 1



Text

Text displayed within the button.

Bitmap

Bitmap displayed within the button.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Parent/Child

List of key fields that bind the Parent to the Child.

This list defines all fields that relate the Parent object to the Child object. CodePainter will pass on these fields when a search activity is required or in Load mode when the Child requires the Parent's key.

Table Name

Name of the Child file.

Program

Name of the Child file's managing procedure.

Child editing

Child's editing mode.

'No'

The Child is activated when the button is clicked.

'Edit'

The Child's managing procedure is integrated in the Parent's dialog window.

'Paint'

The Child is in a different window, but is activated with the Parent window.

'Hide'

The Child is hidden. Data in the Child is canceled when it is canceled in the Parent. Data in the Child cannot be changed.

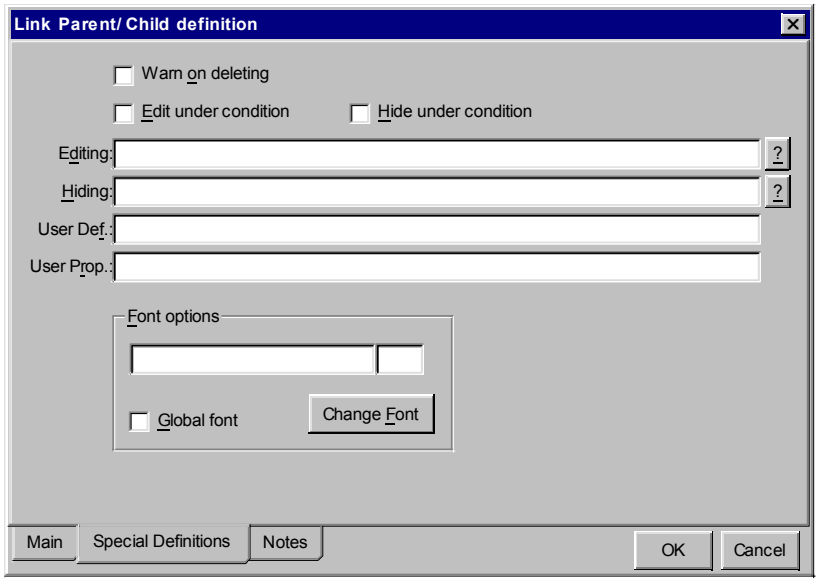
Get child size

Measures the Child's editing window and adjusts the Parent's window so that the Child fits in.

This button is used when the 'Child editing' option is set to 'Edit'.

Special Definitions Page

Picture 139 - Link Parent/Child Definition: Page 2



Warn on deleting

When you are about to delete records of the Parent entity this option warns you that the Child entity data will be deleted as well.

Behind small windows there may be a number of fields hidden in a Parent/Child link button. To avoid deleting this hidden data by mistake you can set this checkbox. Before deleting the system checks the Child's contents. If data is found a message asking you to confirm the delete command is displayed.

Edit under condition

Defines whether the button is enabled only under specific conditions.

Buttons may be enabled only when specific conditions occur. To define conditions set the 'Edit' option to edit and define the 'Editing' formula in the textbox that must be checked before the cursor is positioned on the button. When the logical expression returns a TRUE value the button is enabled. Otherwise the cursor is passed on to the next element.

Hide under condition

Hides buttons

Buttons may be hidden under specific conditions. To define these conditions set the 'Edit' or 'Show' option and define the 'Hiding' formula in the textbox that must be checked. When the logical expression returns a TRUE value the button is hidden and therefore not displayed on the dialog window.

Editing

Formula to activate the conditional enabling.

This formula defines whether the button is enabled or not. To define conditional enabling you need to define the 'Edit' option in the 'Edit' area, set the 'Edit under condition' checkbox, and define the 'Editing' formula in the textbox. When the logical expression returns a TRUE value the button is enabled. When the returned value is FALSE the button is disenable.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Hiding

Formula to hide the button.

This formula defines whether the button must be hidden or not. To define these conditions set he 'Edit under condition' checkbox and define the 'Hiding' formula in the textbox. When the logical expression returns a TRUE value the button is hidden. When the returned value is FALSE the button is displayed.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

User def

Free definition.

Could be used by a custom template.

User prop

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to add definitions to the ones generated by CodePainter.

Example

```
forecolor=RGB(255,0,0)
```

Change Font

Define the elements font. If no font is defined the dialog widow's default font is used.

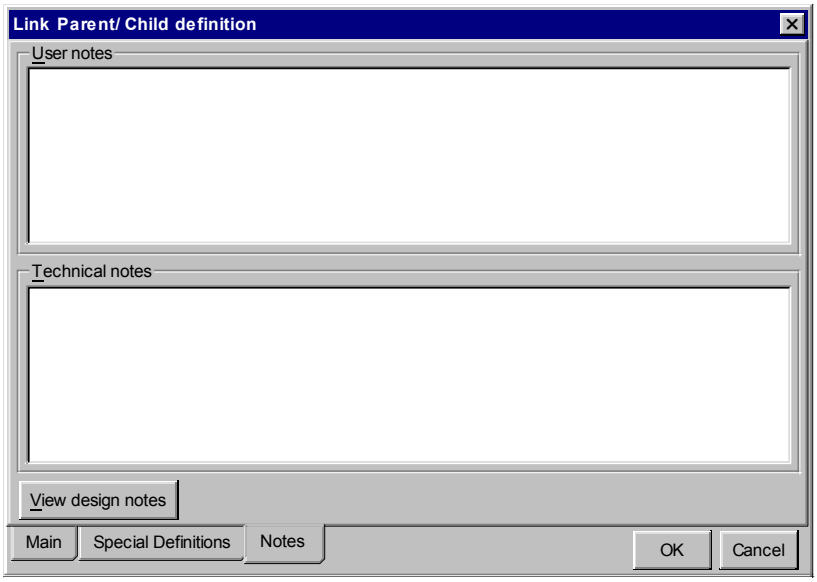
Global font

When selected the font defined in the 'Global Font' menu is applied to the current element.

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.



Picture 140 - Link Parent/Child Definition: Page 3

User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

Technical notes on the element.

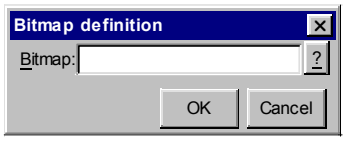
These notes are used to create technical documentation.

View design notes

Quick access to the Design notes of the current element.

4.7.6 Bitmap Definition

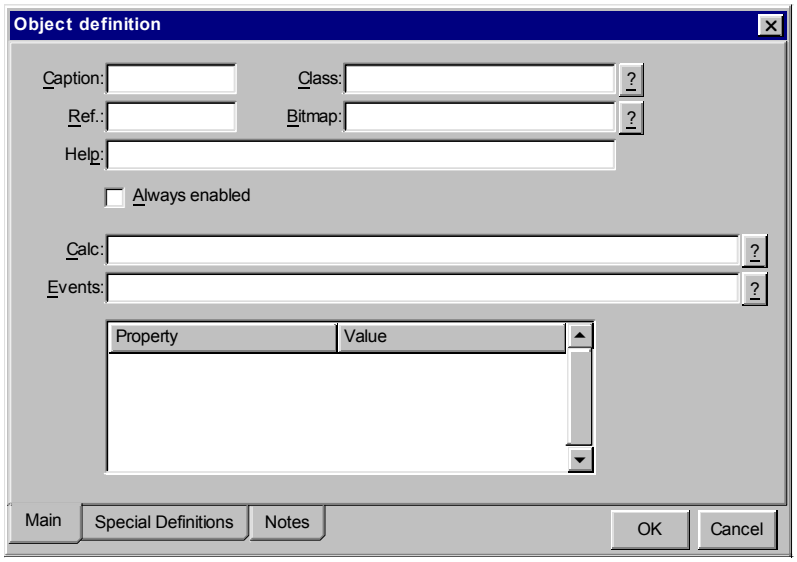
Picture 141 - Bitmap Definition: Page 1



Bitmap

4.7.7 Object definition

Main Page



Picture 142 - Object Definition: Page 1

Caption

Object text.

This text is used as title for those objects that need a string to be displayed.

Class

Object class.

Objects that can be added to dialog windows must belong to predefined classes. CodePainter uses a run-time class library or dedicated templates to generate the relevant source code. The class identifies which tool must be used.

Clicking the '?' button you can access all classes installed in your development environment. For more information on classes and their use, please refer to the COMPONENT GUIDE.

Ref.

Name of the variable associated to the object.

Objects are created as 'controls' in the current form. Associated working variables are not created, because objects are not read by CodePainter's standard procedures.

When you require to access the object from a manual area or a routine, you first need to define the variable name in this property. The working variable created works like the ones associated to fields or variables and is initializated with the pointer to the object.

Bitmap

Bitmap associated to the object.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Always enabled

The object is enabled in 'Editing' and 'Query' mode.

Objects are used to access procedures that are bound to the main dialog window. Using these procedures may make sense in the 'Editing' and/or in the 'Query' mode only. E.g. an object executing a graph should always be enabled.

Setting this flag the object will be enabled in the 'Editing' as well as in the 'Query' mode. When the flag is not set the object is enabled in the 'Editing' mode only.

Calc

Passes on values to the object every time the dialog window is re-calculated.

The input dialog window is recalculated when specific actions are performed: when a new record is input, or when the user inputs specific fields, etc. Each time the dialog window is re-calculated it notifies the object passing on as parameter the value defined in this expression. The object will react according to the class to which it belongs.

For example the 'Dash Board' ('cruscotto') changes the position of the arrow, the 'Calendar' ('calendario') selects the current date, etc.

For more information on how each class exploits recalculations please refer to the COMPONENT GUIDE.

When calculations are complex you can limit re-calculations defining the 'Depends on' property as for any field or variable.

Events

List of events to which the object is hooked.

Objects can communicate with the dialog window in which they are in either through recalculations or events.

Events are notified by the dialog window. The object will react according to the class's specifications to which it belongs. Clicking the '?' button next to properties you access the list of available events. For more information on how each element answers to a given event, please refer to the COMPONENT GUIDE.

Properties

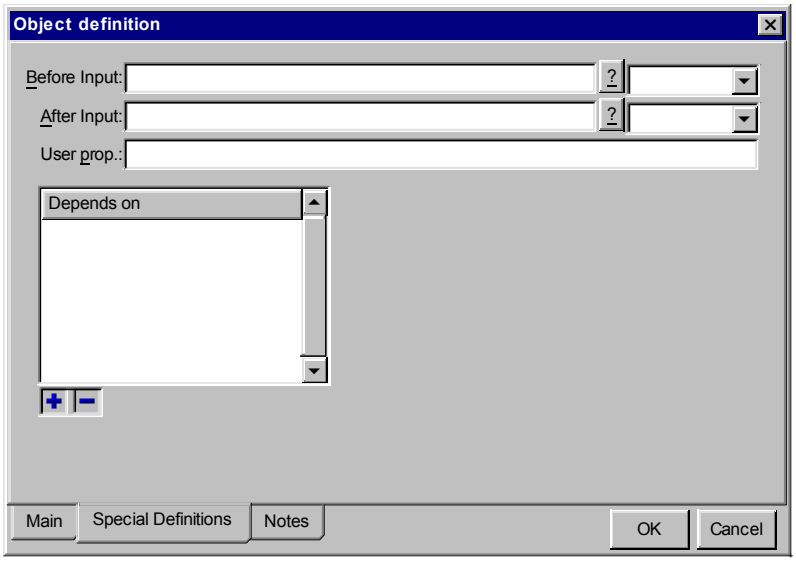
List of object properties.

Objects have a set of properties that differ according to the class to which they belong.

This list displays the property name in the first column and the default value in the second. These values can be changed in order to configure the object. For more information on object properties, please refer to the COMPONENT GUIDE.

Special Definitions Page

Picture 143 - Object Definition: Page 2



Before input

Program that must be executed before the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low level and cannot be defined so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

After input

Program that must be executed after the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low level and cannot be defines so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

User prop

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to adddefinitions to the ones generated by CodePainter.

Depends on

List defining the elements on which the calculation execution for associated objects depends.

Calculations and links are normally executed every time a value is entered. This strategy works smoothly if the calling routine is quick. If calling routines take long to be terminated you need to limit the number of calls.

When defining working variables in this list, calculations and associated links are executed only when the value for one of these variables changes.

In this list you should define all calculation parameters associated to the element.

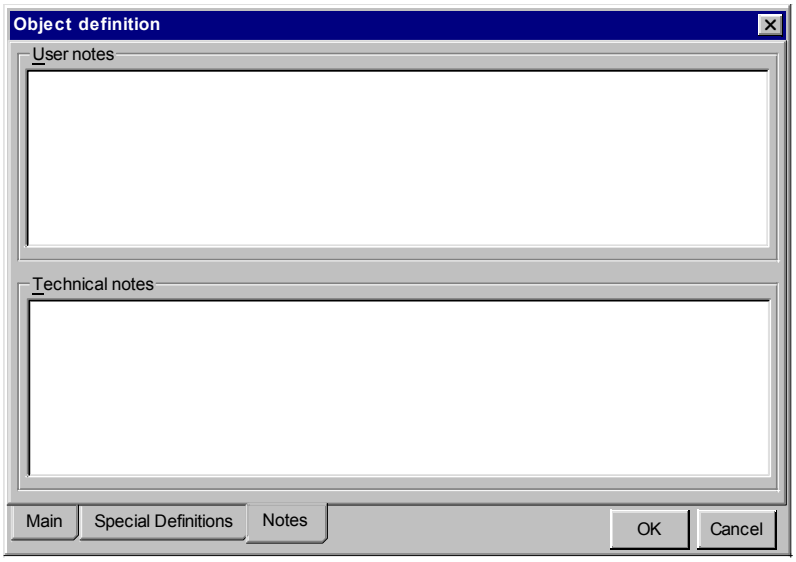
Using this method you avoid executing many code lines. Nevertheless checking the change of the value is costly and therefore this option should be activated only when calculations are complex.

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.

Picture 144 - Object Definition: Page 3



User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

Technical notes on the element.

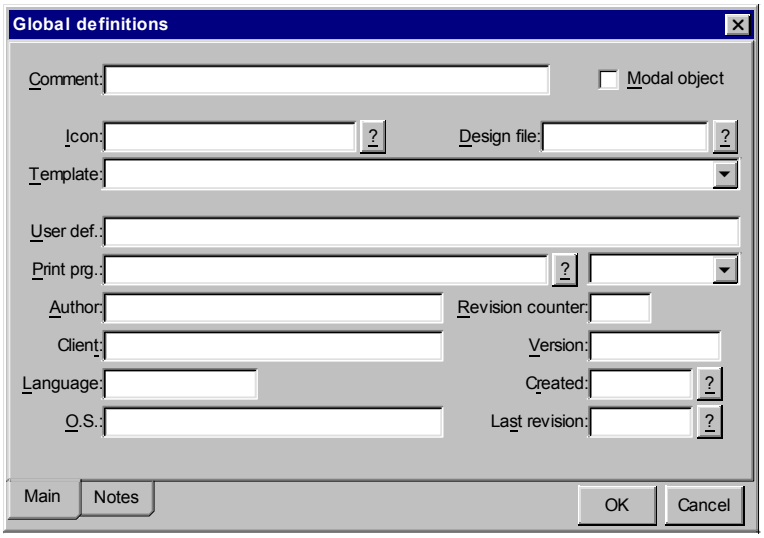
These notes are used to create technical documentation.

4.8 'Globals' Menu Advanced Options

This section details the 'Globals' menu functions.

4.8.1 Global Definitions

Main Page



Picture 145 - Global Definitions: Page 1

Comment

Brief comment on the entity. This comment is displayed in the project file so that you can search entities by file name (i.e. the associated procedure), or by brief description.

Modal object

Defines whether the entity is a modal dialog window or not. Modal dialog windows ask to terminate the input before you can go on to another dialog window. This means that the user cannot use the mouse to go to other windows until the current window is saved (F10) or exit (Esc).

Icon

Icon associated to this entity.

Template

Name of the template used for code generation. In this property you need to define the name of the template used to generate the source code. Templates are procedures' skeletons, finalized by the programmer to generate the required source code. Change the template and the generated source code changes as well.

User def

Free string. Can be used by custom templates.

Print prg.

Program activated during queries pressing F2.

In this property you can define the program that must be called when the user clicks the 'Print' button during queries. The combobox next to the property defines how the command line must be interpreted.

'User program': the command is copied in the source code. 'Mask': a dialog window created with the Dialog Window Painter is opened. 'Batch': a routine created with the Routine Painter is executed. You can also pass on parameters defining them ion brackets. 'UTK Object': defines an output configuration created with user tools such as the Query Painter.

The '?' button next to the property displays a list of possible choices depending on the defined activity.

Author

Author of the program. Short text defining the developer in charge of the entity.

Client

The commissioner. Short text to define the customer who commissioned the entity.

Language

Development language. Short text describing the programming language used to develop this entity.

O.S.

Operating System. Short text defining the limits given to the entity by the operating system.

Revision counter

Revision counter. This read only property shows you how often the entity has been changed. The number is increased automatically every time the entity is saved. This property can be used to check whether copied objects have been changed by the original author.

Version

Entity version number.

Created

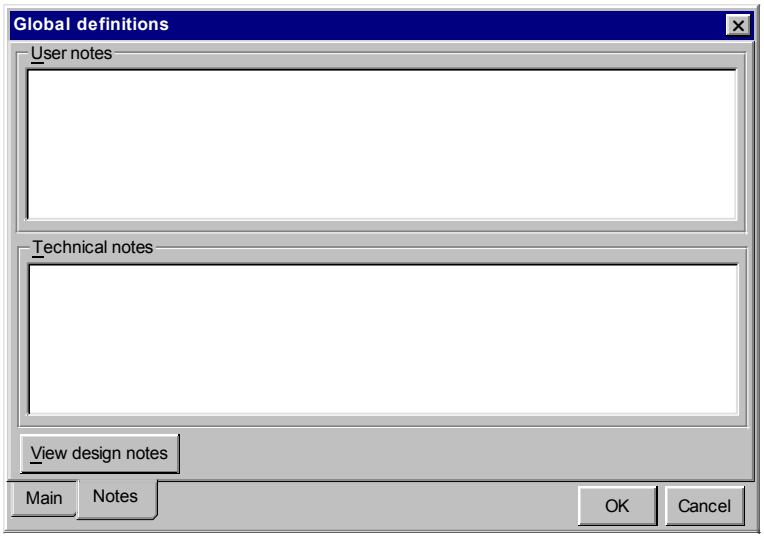
Date in which the entity was created.

Last revision

Date in which the last revision was made.

Notes Page

Picture 146 - Global Definitions: Page 2



User Note

Notes on the element. These notes are used to create user documentation.

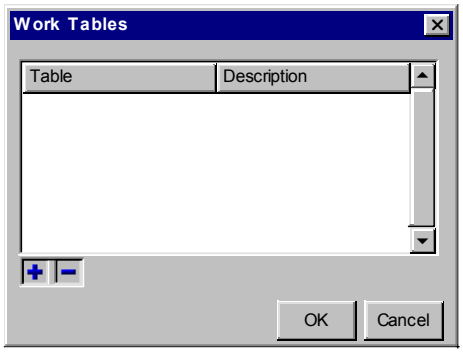
Technical Notes

Technical notes on the element. These notes are used to create technical documentation.

View design notes

Quick access to the Design notes of the current element.

4.8.2 Tables



Picture 147 - Tables

Tables

List of tables.

The generated procedure controls that the database contains all tables displayed in this list. When one or more tables are missing an error message is displayed and the procedure is not executed. This is done to avoid errors.

The procedure can also use tables which are not included in the list. Programs should not be executed if they can generate errors.

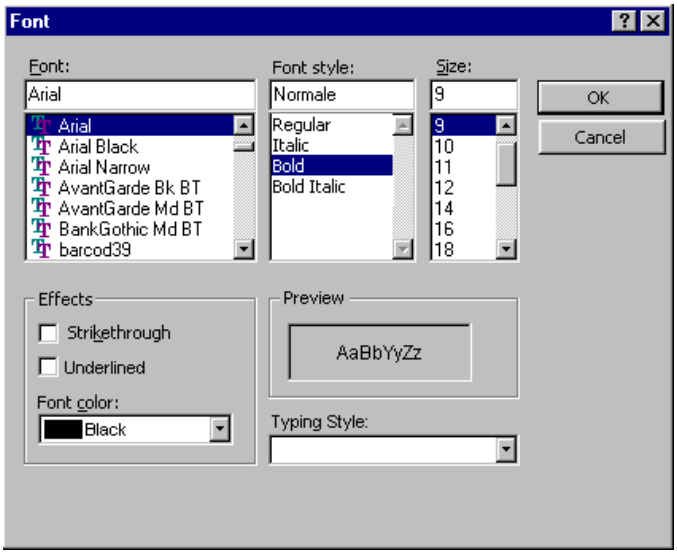
Tables used in links are automatically added to the list.

4.8.3 Font

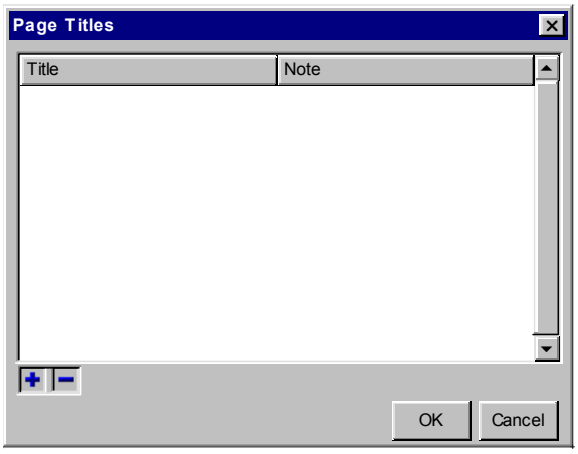
Allows changing default fonts for the entity in use.

Default fonts in this area are the ones you defined in CodePainter's Front End in the 'Project' menu in the 'Project Options' option.

Picture 148 - Font Definition



4.8.4 Page Titles



Picture 149 - Page Titles: Page 1

Titles

Page titles. Entities can have dialog windows having more pages. The default title is 'Pag.' followed by the page number. This list allows changing pages' title.

4.8.5 Autonumber

Manages progressive numbering automatically.

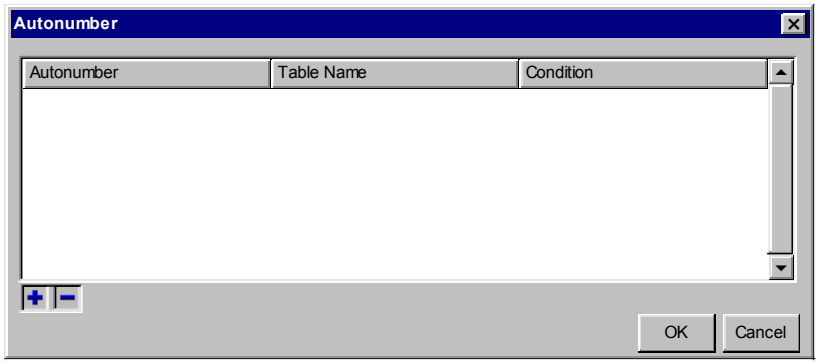
You can define fields that are numbered automatically in each entity. Once the starting number is defined the field is increased every time a new record is input.

When the 'autonumber' option for a field is defined, the field is initialized with the next value in the numbering sequence when a new record is input. If the user does not change the value and the record is saved the table containing progressive numbering is read. The system checks whether any other user has taken the 'booked' number. If the 'booked' number is still available the record is saved using it. If not the next available number is identified, the record saved with this new number and a message displayed so that the user is notified of the change. This method makes sure that in multiuser environments all available numbers will be used and that no numbering gaps are created.

When the user edits the 'booked' value the field value is overridden. The 'custom' number could be greater or less than the one given by the system. When the number is greater than the one given by the system the record is saved with that number and a numbering gap is created. When the number is less than the one given by the system the record is saved without changing the 'booked' field value nor the autonumbering table.

This second method allows to postpone data entry for known quantities. For example Invoices are input by the Head office. Unit B has issued 10 invoices but has not yet send the data. Head office can leave a gap of ten units overriding the next invoice number. When the data from Unit B arrives it can be input using the 10 numbers left empty.

Picture 150 - Autonumber



Autonumber

List of progressive numbers.

This list contains the fields building the progressive number, the reference table, and if required the expression that defines whether the progressive system must be used or not.

Autonumber

Name of the field that will be linked to a progressive table. You can define more fields simply dividing them with commas. In this latter case the last field is the one taking on progressive numbering, whereas the other fields are the 'variations'.

Example

You need to number a specific document and you also need to differentiate in-coming from out-going documents. You need to define two fields:TIPDOC and NUMDOC. The system will create progressive numbers for each TIPDOC value and inputting the progressive number in NUMDOC. Fields receiving progressive numbers can be either numeric or alphabetical. In order to match the alphabetical order with the sequence character types '0' are added to in front of the letter until the field is filled in completely.

Table name

Name of the table containing the progressive value. Progressive values are stored in a dedicated table. This field contains a symbolic name required to identify which numbering must be used. You can create separate numberings, e.g. 'cli' for the customer key, 'art' for the item code) or numberings that can be used by more entities, e.g. different documents such as orders and invoices, using the same numbering.

Condition

Sometimes you may be required to use progressive numbering only under certain conditions. For example you may need to save in-coming and out-going in one entity only. You define the field NUMDOC that must be automatically numbered for out-going documents, but in which you want to manually input numbers for in-coming documents. If out-going documents have the value 'O' in the field TIPDOC you need to define the condition TIPDOC='O' so that progressive numbers are used only for these documents.

Autonumber

You want automatic progressive numbering on invoices.

The field TIPODOC contains the document type ('I' for invoices, 'R' for receipts). The field NUMDOC must contain the progressive number for invoices and the manually inputted numbers for other documents.

The list will contain:

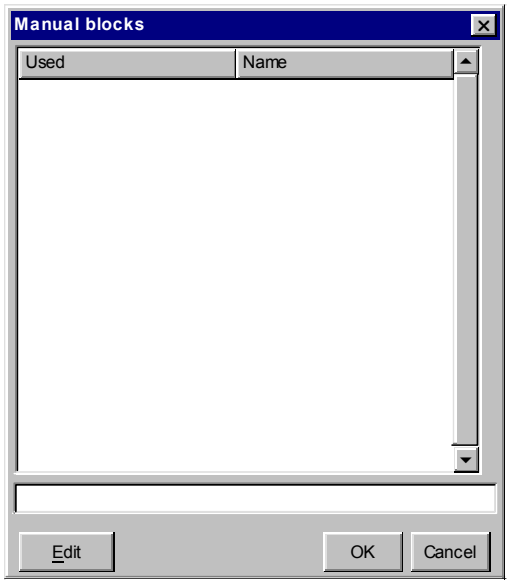
'w_NUMDOC' in the 'Autonumber' column.

'doc' in the 'Table Name' column.

'TIPDOC'="F" in the 'Condition' column.

4.8.6 Manual Blocks

Picture 151 - Manual Blocks: Page 1



Used - Name (List of manual areas)

List of manual areas contained in the template.

Manual areas containing some code are highlighted in the left column.

Manual Areas

Name of the manual area that must be edited.

You can select manual area either from the list or defining different name in case the desired manual area is not included in the list.

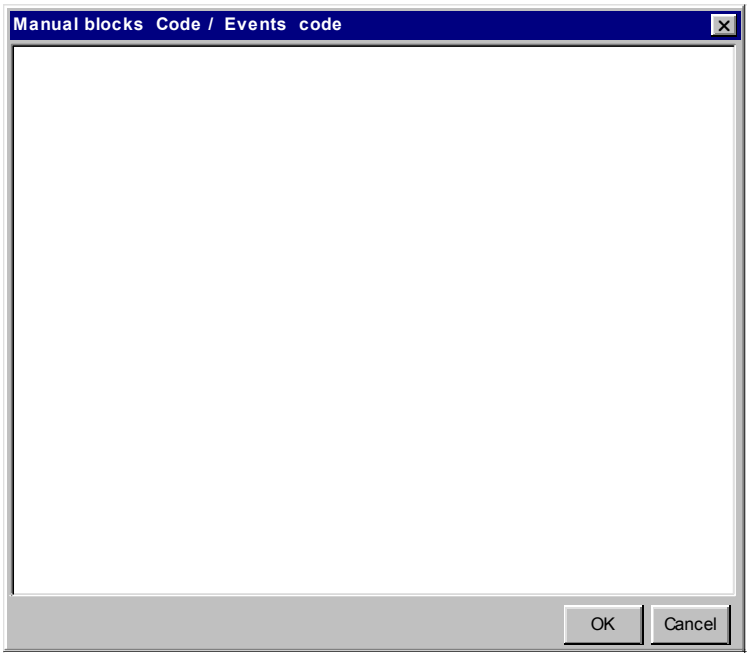
Edit

Access the code defined in the selected manual area.

Double clicking the manual area a dialog window is opened where you can define the code :

4.8.7 Manual Blocks Code

Editing dialog window for manual areas.



Picture 152 - Manual Blocks Code

Code

Source code contained in the manual area.

4.9 Selection Lists

CodePainter has a set of lists based on the project's data dictionary.

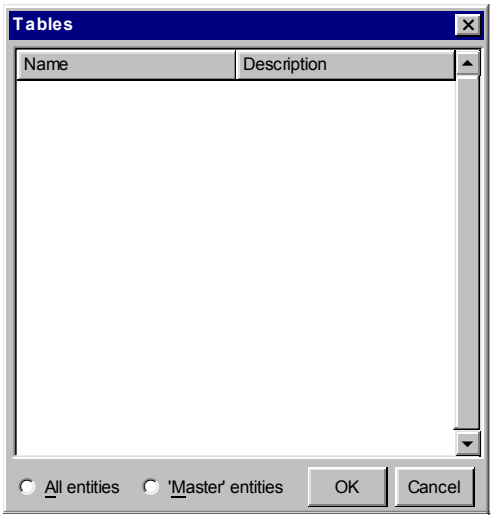
These list are generally activated in the Codify tool using the '?' or '...' buttons. These lists make information required for development available.

These lists can be sorted double clicking the column title. Selected values are input in the desired window section.

The sequence of selected elements can be moving them up or down in the list. To move elements you need to position the mouse on the first column on the left and wait until the mouse changes into a hand.

This section details the use of selection lists.

4.9.1 Tables



Picture 153 - Tables

Tables List

List of available tables.

Entity Type

All entities

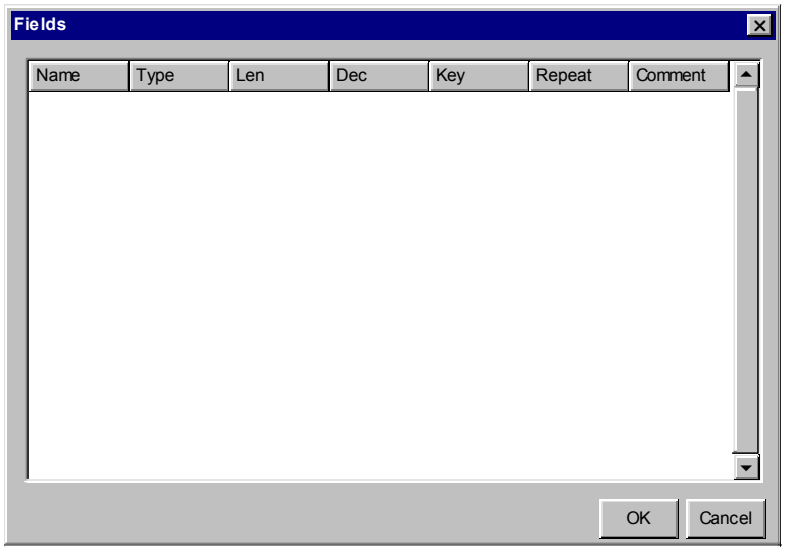
All tables are displayed.

'Master' Entities

Displays only tables belonging to predefined entity classes.

4.9.2 Fields

Picture 154 - Fields

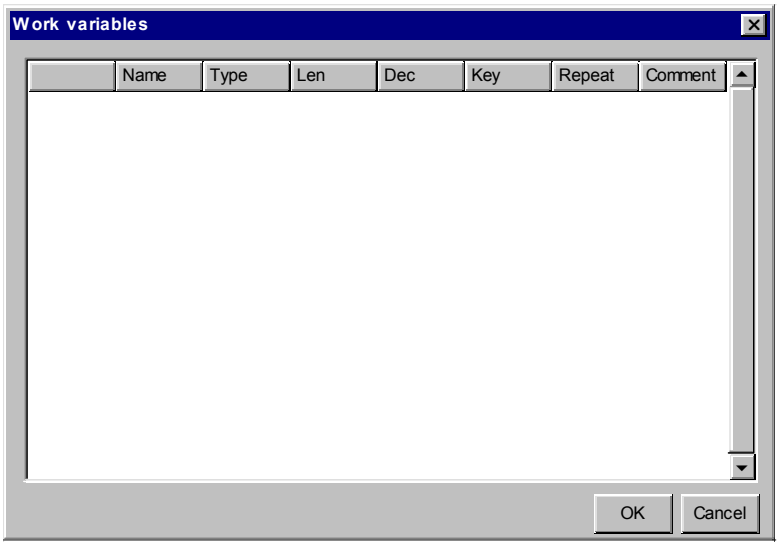


Fields List

List of fields. The list contains all fields defined.

4.9.3 Work Variables

List of variables that have not been implemented so far.



Picture 155 - Work Variables

Variables

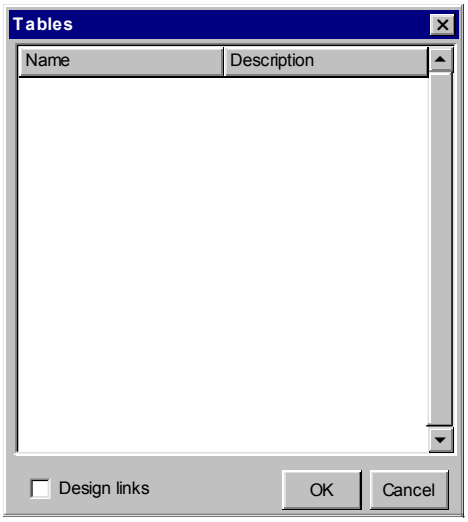
List of declared variables that have not been implemented so far.

Very often when you define links, variables are defined before they are displayed on the screen. This list analyzes all link definitions and details all available variables. The main advantage is that data is read directly from the data dictionary. Therefore it also reads data type and length, so that the new element can be added to the screen with the correct settings.

4.9.4 Tables

List of files that can be used to define links.

Picture 156 - Tables



Files

List of all tables in the design plan.

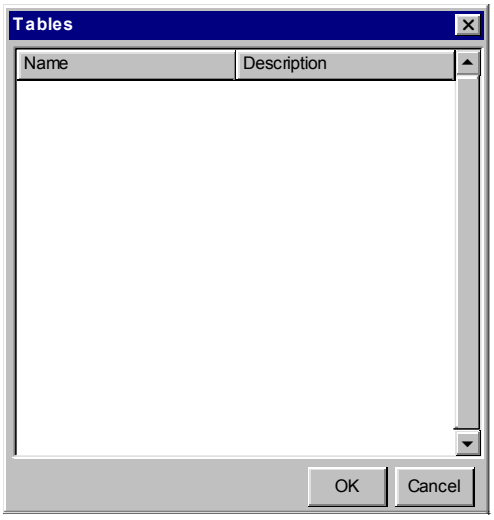
Design Links

Defines whether to list all tables in the design plan or only those for which a link has been defined during the Design phase.

In the design plan each entity has a set of declared links. During the Codify phase you can create new links for which the source code is generated but not the database referential integrity. It is recommended to use declared links only.

When this flag is set the list above shows declared links. When the flag is not set all tables are displayed.

4.9.5 Tables



Picture 157 - Tables

Tables list

List of available tables.

Entity Type

All entities

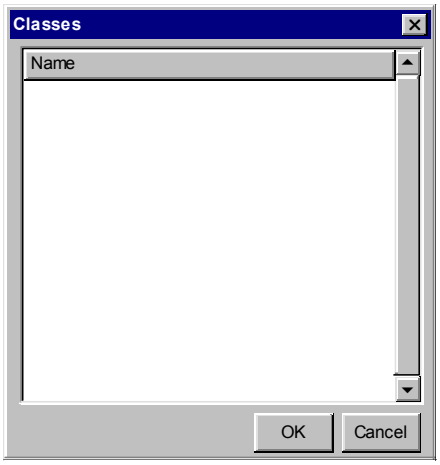
Displays all entities.

'Master' entities

Displays only entities belonging to the Master class.

4.9.6 Classes

Picture 158 - Classes



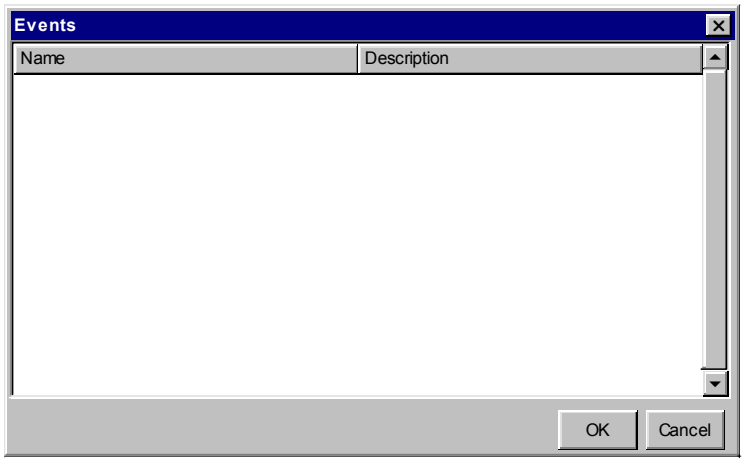
Classes List

List of available classes.

This list contains the name of object classes defined in CodePainter.

All defined classes are stored in the file CLASSES.CPL under the Painter's Classes directory. To add new classes you simply need to define them in this file.

4.9.7 Events



Picture 159 - Events

Events list

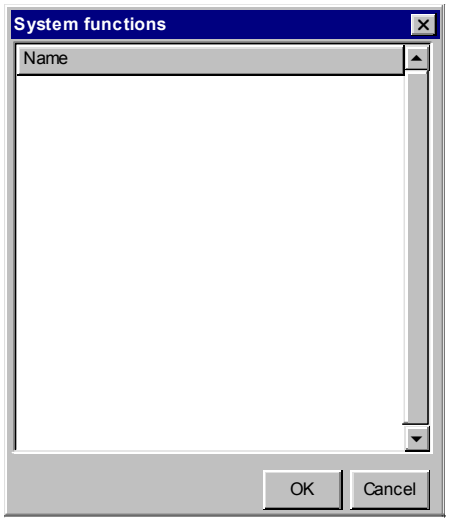
List of Events.

At run-time CodePainter notifies events to the objects in the dialog windows. Objects will therefore be able to react to given situations, such as data loading, change of an element or saving a record.

This list contains the name and brief description of all events that can be notified.

4.9.8 System Functions

Picture 160 - System Functions

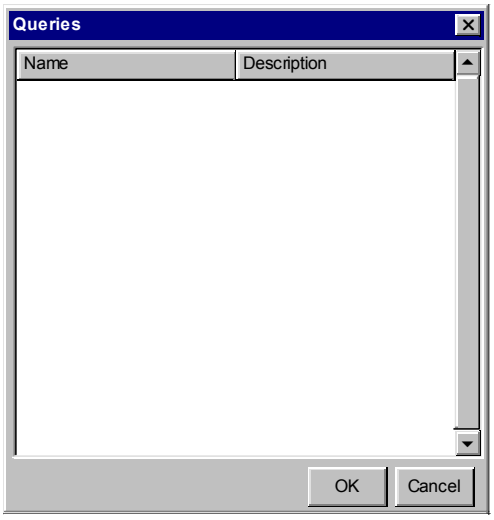


System Functions

List of system functions.

System functions are identified by strings. This list contains all system functions implemented by CodePainter's run-time system.

4.9.9 Queries



Picture 161 - Queries

Name - Description (Tables list)

List of available tables.

All entities - Master entities

All entities

Displays all tables.

'Master' entities

Displays only tables that belong to a to an entity of this class.

Design links

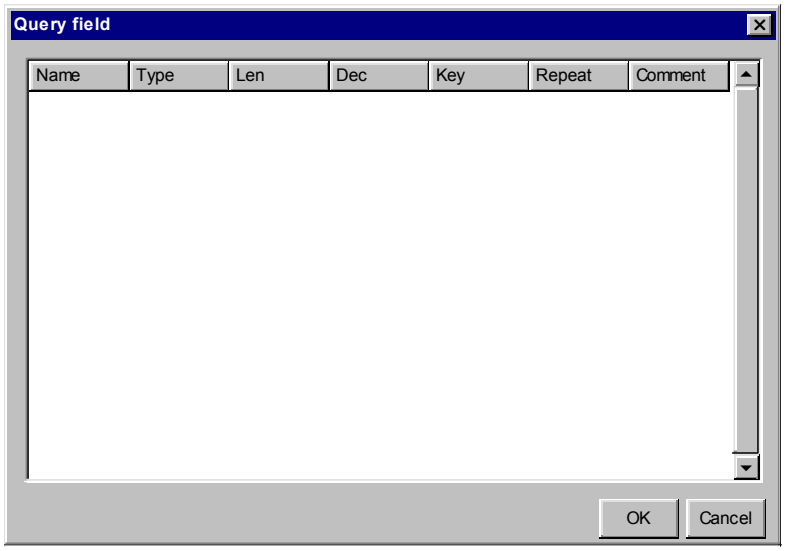
Defines whether to display all tables or only those associated to a table for which a link has been defined in the design plan.

When the flag is set the list displays only linked tables having referential integrity. When the flag is not set the list displays all tables.

You can create links in the Codify phase without going back to the design plan. The code to manage the link is created but the referential integrity to the database is omitted.

4.9.10 Query Field

Picture 162 - Query Field: Page 1



Fields list

Field list.

This list contains all fields of the selected query.

Fields list

Field list.

This list contains all files of the selected entity.

4.9.11 Design Notes

Picture 163 - Design Notes

Design Notes

Displays the notes added to the elements during the Design phase.

4.10 Product Information

The Help menu option 'About' displays information on the product.

4.10.1 About

The About dialog window contains information on the procedure in use.

The logo CODE**PAINTER REVOLUTION** is displayed and information on the version, series number and build number are given.

The build number helps you identifying CodePainter's distribution, no matter if the product is released totally or partially or through patches.

Picture 164 - About: Page 1

Chapter 5

Detail File

5.1 Detail File Painter

Business/commercial applications often require to manage documents in which fields must be repeated.

Detail File Entities have a '*Header*' containing general data, a '*Body*' in which fields can be repeated, and a'*Footer*', containing other general data and/or totalization values.

Detail Entities manage single tables and processes all records having the same primary key together, considering these records as a single document. Using this kind of entity you can e.g. create an order document dealing with many repeated order lines.

Another typical example for Detail File Entities is the management of warehouse transactions. The file contains one record for each item transaction. Each record contains item code and quantity. Header and Footer contain general information such as warehouse code or transaction code.

Detail Files can be created also in the Codify phase basing on the application's data dictionary. In this case you have two different procedures that manage the same database table.

The Detail File Painter is used to improve the prototype's layout and functionalities, or to create new Detail File entities that will be included in the design plan.

Using the Detail File Painter you can improve the window's layout adjusting elements' size and position and you can also add strings, variables, buttons etc. using the WYSIWYG (What You See Is What You Get) methodology. You can further define initialization properties, calculations, validation checks, etc.

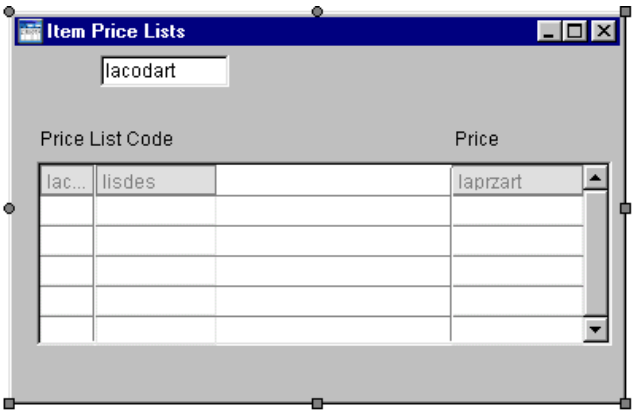
When you require to make changes to the Design plan you can maintain the changes made and generate the prototype only for changed elements in the entity. This is achieved using comparison functions that will be explained later on.

5.2 The Working Logic

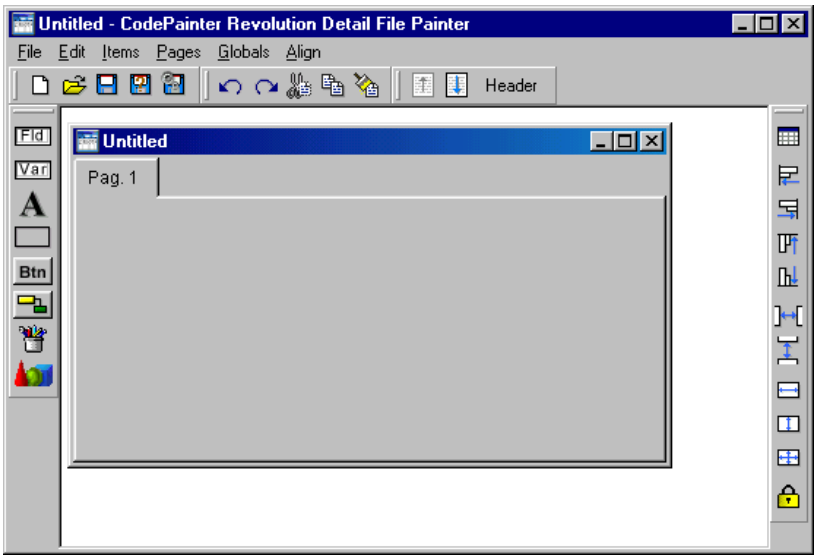
In the Detail File Painter you can open prototyped entities. A set of toolbars help you interacting with the prototyped entities. You can either open entities defined in the design plan or create new ones basing on the application's data dictionary. Opened Detail File entities have a variable number of rows.

To open defined entities open the 'File' menu and select the 'Open' option. All existing Detail File entities are listed. Fields defined for the Detail File in the design plan have been prototyped. These fields and descriptions defined in the linked table are included in the Detail File prototype and are sorted basing on the data dictionary.

Picture 165 - The Detail Painter Prototype



To create new entities open the 'File' menu and select the 'New' option. You can also use blank window which is defaulted when you open the Detail File Painter.



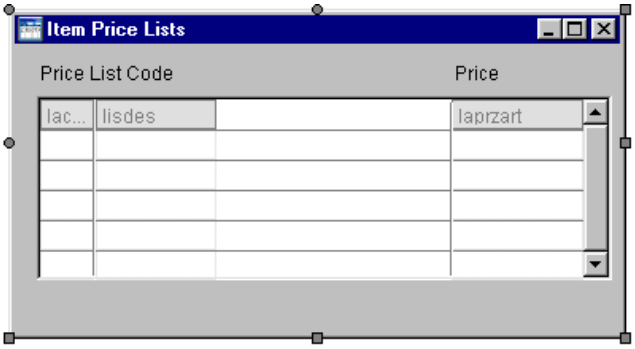
Picture 166 - The Detail Painter: New Window

Using the Detail File Painter you can '*design*' the interface and build in validations. Using the 'Code Generation' option you obtain a working procedure without the need to regenerate the entire design.

In the opened entity you can select, move and resize elements as you would do in any other MS Window application.

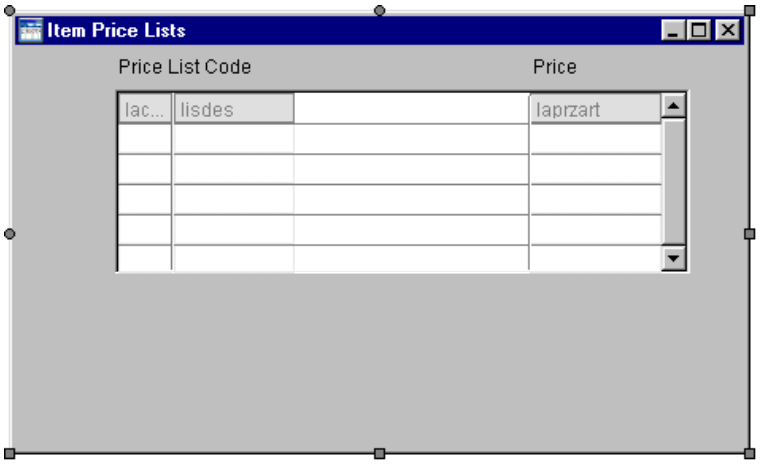
To move the entity window click the title (Caption Bar) and drag it in the desired position. You can notice that while you move the window the mouse changes into a cross.

Picture 167 - Window Selection

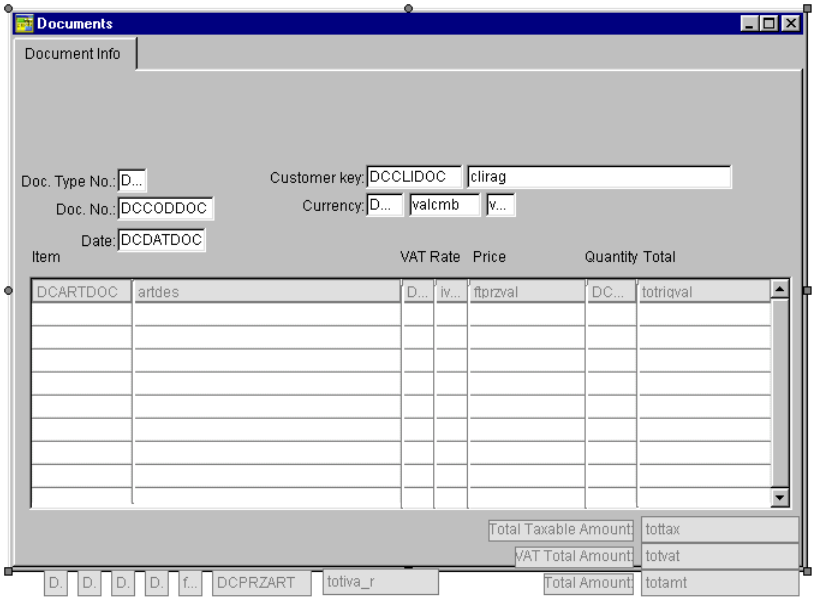


To resize the window you need to select the entity and drag the displayed anchors. The size is always changed according to the direction in which you are dragging when the mouse becomes a 'two ways' arrow. The elements in the window remain fixed compared to the to left corner.

Picture 168 - Window Re-sizing By Dragging



When you right click and drag one of the circles the elements remain fixed compared to the bottom right corner. You can notice that while you right click and drag the circle an anchor is displayed under the 'two ways' arrow.



Picture 169 - Window Re-zing By Right Clicking Circles

Similarly you can move and re-size any element in the entity window.

5.3 Selecting And Aligning Elements

In Detail File and Master/Detail entities you can select elements of the header, footer and body simultaneously and use all 'Align' menu functionalities.

Picture 170 - Aligning Elements OF Header, Body And Footer

Double clicking elements you can edit the definition dialog window even if the selected element does not belong to the current block (header, body and footer).

Warning
Header, body and footer must not be intended as separate pages, but as a logical organization of elements.

Groups of elements can be selected in different ways:

Left clicking the mouse you can drow a rectangle around the elements you wish to select.

Picture 171 - Selected Group Of Elements

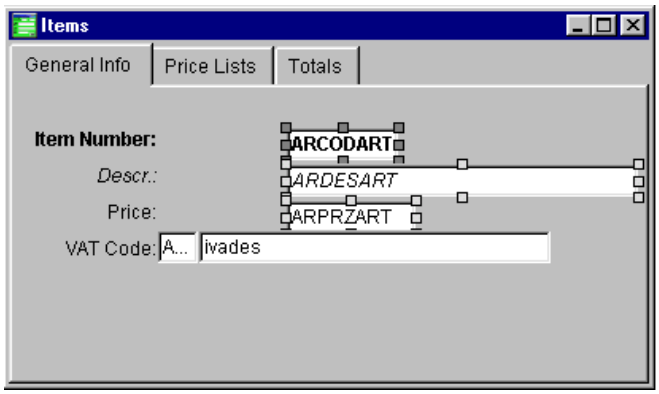
You can also keep the **<Shift>** key pressed and click the elements you wish to select. Single elements are deselected still keeping the **<Shift>** key pressed and clicking again the desired element. All elements are deselected clicking on the working form.

Selected elements are highlighted by handles, little *squares* around the element. Most elements with handles are light grey and one only dark grey. The latter one is the *Master Item* and is used as reference for resizing and aligning the other elements

Warning
When resizing and realigning elements the result changes depending on the Master Item selected.

Aligning the Master Item

Picture 172 - Align to the Left Of The Master Item



Picture 173 - Align to the Left Of The Master Item

The two pictures above show you the different results you will get selecting two different master items. The Master Item can be changed also once the group of elements has been selected simply clicking the desired item again.

You can align and re-size group of elements using both the 'Align' menu or toolbar.

To open the definition dialog window double click the desired element.

Right clicking elements a menu is opened that allows you to edit the element's properties, delete the element or change the editing sequence.

Pressing **<Del>** you can delete one or more selected elements.

Pressing **<Enter>** you can open the definition dialog window of the selected item or in case of multiple selection of the Master Item.

Pressing **<Cursor Keys>** you can move one or more selected elements.

Pressing **<Shift>** and **<Cursor Keys>** you can resize the selected element or in case of multiple selection the Master Item keeping the top right corner fixed.

Pressing **<Ctrl>** and **<Shift>** and **<Cursor Keys>**you can resize the selected element or in case of multiple selection the Master Item keeping the bottom left corner fixed.

Using keyboard keys you can also scroll elements.

Pressing **<Tab>** you can scroll forward.

Pressing **<Shift>** and **<Tab>** you can scroll backwards.

Pressing **<Ctrl>** and **<Tab>** you can scroll a selected group of elements forward.

Pressing **<Ctrl>** and **<Shift>** and **<Tab>** you can scroll a selected group of elements backwards.

To scroll the window elements standard Windows rules apply:

Pressing **<Tab>** you can scroll forward.

Pressing **<Shift>** and **<Tab>** you can scroll backwards.

When you are positioned on tabstrips you can press the **<Cursor Keys>** to scroll the option pages.

Pressing **<Alt>** and the <UnderlinedLetter> you can edit the corresponding option.

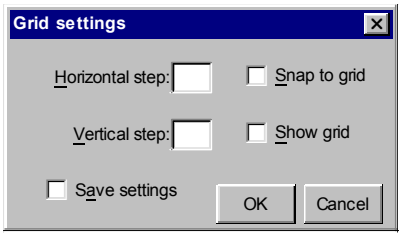
In selection lists you can sort columns. You can also add and delete elements using either the mouse or the **<Ins>** and **<Del>** keys.

When you move elements using the **<Cursor Keys>** changes are madein pixels. When you align elements you can define whether to use a positioning grid or not.

5.3.1 Grid Settings

Design grid definition.

Picture 174 - Grid Settings: Page 1



dx

Horizontal path. The grid has anchors. The distance between anchors is given by the number of pixel defined starting from the left border of the design window.

dy

Vertical path. The grid has anchors. The number of anchors is given by the number of pixel defined starting from the top border of the design window. When the dialog window has more pages the anchor is placed on the top border of the tab-strip.

Snap to grid

Defines whether the elements must be within the grid. When the flag is not selected the elements can be placed anywhere in the window. When the flag is set elements must be in fixed places.

Show grid

Defines whether the grid must be displayed or not. When elements in the dialog window are many and close to each other the grid makes the window unreadable. When this flag is not selected the grid is still active, but is not displayed.

Save settings

Defines whether grid settings must be saved or not. Grid settings can be defined and saved for each dialog window. When this flag is set settings are maintained.

5.4 The Detail File Painter Toolbars

Let us now analyze the various Detail File Painter toolbars. These toolbars help you interacting with Detail Files.

5.4.1 Painter Tools Toolbar

The Painter Tools toolbar allows you to add new elements to the current entity. The toolbar has the same functionalities as the 'Items' menu.

figura 7 - Painter Tools Toolbar

Here to follow you will find a brief description of the files and their meanings.

Button	Meaning
	Adds database table 'Field' objects.
	Adds 'Memory Variable' objects to support validations and calculations/totals.
	Adds descriptive 'String' objects.
	Adds 'Box' objects that allow you to improve the entity's layout adding lines or rectangles that help subdividing the window.
	Adds 'Button' objects. These objects run queries, procedures, system functions and/or dialog windows.
	Adds 'Parent/Child Link' objects that allow you to create buttons that are integrated in the window or that open 'Child' entities.
	Adds Bitmap objects.
	Adds 'Object' objects that allow you to integrate external objects to your project. External objects belong to predefined classes. Examples are graphs, calendars, zooms, zooms with selection, etc.

5.4.2 File Toolbar

The 'File' toolbar has a set of button that help you interacting with the tool. This toolbar has the same functionalities as the 'File' menu.

Picture 175 - File Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Creates new entities.
	Opens existing entities.
	Saves the changes made to the current entity.
	Saves the current entity with a different name.
	Saves and generates the source code for the current entity.

5.4.3 Align Toolbar

The *Align* menu allows you to position and resize groups of selected elements.

figura 8 - Align Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Opens the grid management
	Aligns elements to the left in comparison to the Master Item.
	Aligns elements to the right in comparison to the Master Item.
	Aligns elements to the top in comparison to the Master Item.
	Aligns elements to the bottom in comparison to the Master Item.
	Positions elements with the same horizontal distance in comparison to Master Item and the selected elements.
	Positions elements with the same vertical distance in comparison to Master Item and the selected elements.
	Re-sizes selected elements with the same height as the Master Item.
	Re-sizes selected elements with the same width as the Master Item.
	Re-sizes all elements with the same height and width as the Master Item.
	Blocks the possibility to move and re-size elements (the functionalities are not inhibited in the 'Align' menu.

5.4.4 Clipboard Toolbar

The 'Clipboard' toolbar allows you to execute typical Window's commands such as Undo, Redo, Cut, Copy and Paste.

Picture 176 - Clipboard Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).
	The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).
	The 'Copy' option copies one or more selected elements.
	The 'Cut' option deletes selected elements.
	The 'Paste' option pastes copied or cutted elements in the selected position.

5.4.5 Pages Toolbar

Using the 'Pages' toolbar you can browse the entity's pages. The toolbar shows two buttons and the number of the page in which you are positioned.

Picture 177 - Pages Toolbar

Here to follow you will find a brief description of the buttons and their meanings

Button	Meaning
	Moves the edit functionality from the previous page to the current page. This button does not work if you are on the first page.
	Moves the edit functionality from the following page to the current page. This button does not work if you are on the last page.

5.5 Main Manu

5.5.1 File

Picture 178 - File Menu

The 'File' menu has a set of options to:

- Create new entities.
- Load existing entities.
- Save changes to the current entity.
- Save the current entity with a different name.
- Save and generate the current entity.
- Read information about CODE**PAINTER REVOLUTION**.
- Exit the tool.

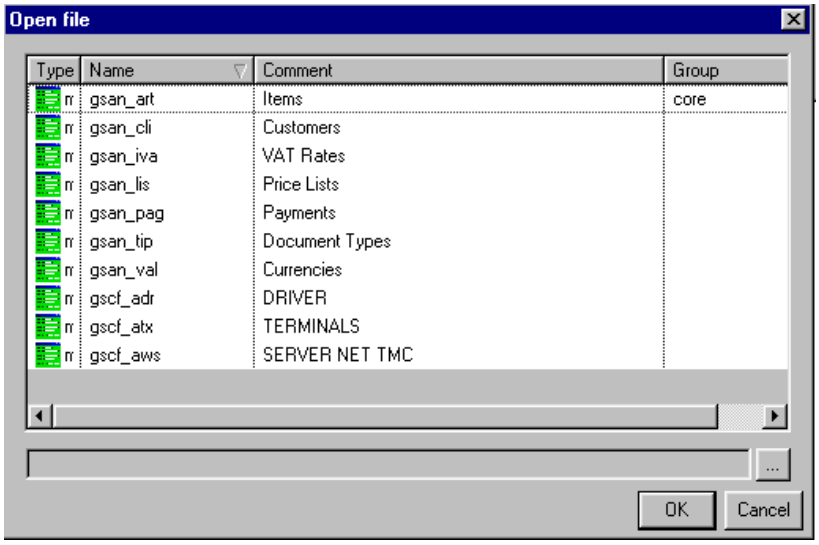
New

The 'New' option closes the current entity asking whether you want to save the changes or not and opens a new entity.

Open...

The 'Open' option loads definition files belonging to the current project.

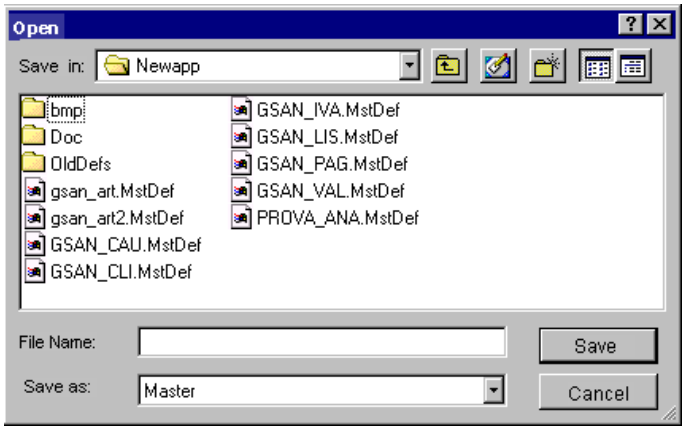
Picture 179 - Open File



The dialog window lists all entities in the project. You can order files by Type, Name, Comment or Group clicking the corresponding column. You can identify the sorting column by the arrow next to the column description.

Picture 180 - Sort Columns

Clicking the '...' button you can browse to search entities of the same type belonging to other project modules.



Picture 181 - Open Dialog Window To Select Design Plans

Save

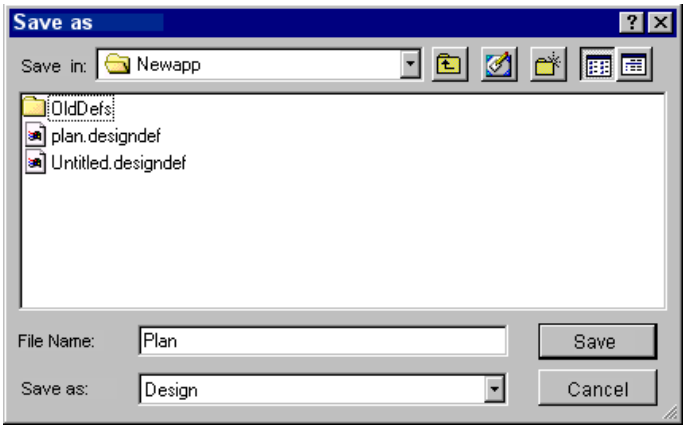
The 'Save' option saves the entity in use.

When you save the entity back-up file (.BAK) is created to store the entity without including the latest changes.

Save As...

The 'Save As' option saves the entity with a different name.

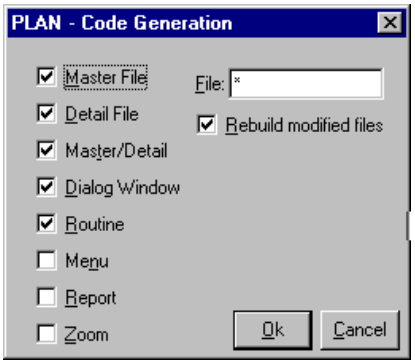
Picture 182 - Save As Dialog Window



Save and generate

The 'Save And Generate' option saves the current entity and runs the generation. The same is done opening the 'Generation' menu and selecting the 'Design'.

Picture 183 - Code Generation



About

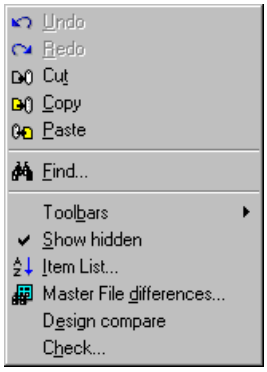
The 'About' option displays information about the product.

For more information please refer to section 'Product Information'.

Exit

The 'Exit' option makes you exit the tool asking you whether you want to save the changes or not.

5.5.2 Edit



Picture 184 - Edit Menu

Opens a menu from where you can:

- Undo/redo commands.
- Delete, copy and add elements.
- Search, display, replace elements.
- Show/ hide toolbars.
- Show/ hide elements with the 'Editing' flag set to 'Hide'.
- Display the 'Item List'.
- Identify the discrepancies between Codify and the Design.
- Identify errors in the source code.

Undo

The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).

Redo

The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).

Copy

The 'Copy' option copies one or more selected elements. More elements can be either selected or grouped in a selection frame you can draw with the mouse.

Cut

The 'Cut' option deletes selected elements from the plan.

Paste

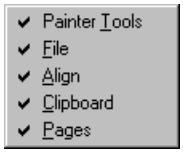
The 'Paste' option pastes copied or cutted elements in the selected position.

Find...

The 'Find' option opens the Find and Replace dialog window to find and/or replace elements.

Toolbar

The Toolbars option has displays a submenu to enable or disenable toolbars on the Design Painter.



Picture 185 - Menu Toolbar

Show hidden

The 'Show Hidden' option allows you to show or hide variables for which the 'Editing' option has been set to 'Hide'.

Item List...

The 'Item List' option displays the a list of all elements in the window.

File Painter Differences...

Using this option you can compare the entity with the design plan. For more information please refer to section 'Edit Menu Advanced Options'

Design Compare

Using this option you can compare the file definition with the entity defined in the Design phase. For more information please refer to section 'Advanced Edit Menu Options -Design Compare'.

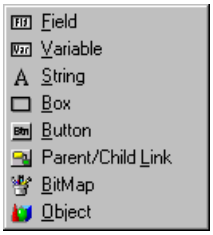
Check...

The 'Check' option checks the congruence of expressions and relations defined during the Codify phase. For more information please refer to section 'Edit Menu Advanced Options'.

5.5.3 Items

The 'Items' menu allows adding elements to the entity in use. It has the same functionality as the 'Painter Tools' toolbar.

Picture 186 - menu' Items



Here to follow you will find a brief description of the various elements:

Field

Using this option you can add 'Field' objects corresponding to a field of the associated table to your entity. When this option is selected the 'Fields of Tables ...' window is opened containing the list of the entity's fields. The list is read from the data dictionary.

Variable

Using this option you can add Variable objects to your entity. These objects are memory variables used to support validations and calculations/totals.

String

Using this option you can add String objects to your entity. These objects are descriptive strings.

Box

Using this option you can add Box objects to your entity. These objects allow you to improve the entity's layout adding lines or rectangles that help subdividing the window.

Lines are defined '*collapsing*' box objects using **<Shift>** + **<arrows>** keys, or the mouse.

Button

Using this option you can add Button objects to your entity. These objects are typically defined to launch queries, process procedures, system functions and/or Dialog windows.

N. B.
Using Button objects to integrate the entity with system functions ('Help', 'Query', 'Edit', 'Load', 'Save', 'Delete', 'Quit', 'PgUp', 'PgDn', 'ZoomPrev', 'ZoomNext') does inhibit standard toolbar functionalities in the generated application.

Parent/Child Link

Using this option you can add 'Parent/Child Link' objects to your entity. At Design level 'Parent/Child' relationships mean hierarchical links between two entities. Used as objects they allow you to create buttons that are integrated in the window or that open 'Child' entities.

Right clicking these objects you can access to the 'Open Codify...' option and the other standard options ('Edit', 'Delete', 'Sequence'). The 'Open Codify...' option allows you to open the codify tool of the integrated Child entity.

Bitmap

Using this option you can add Bitmaps to your entity.

Object

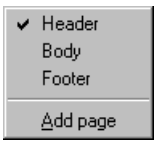
Using this option you can add 'Object' objects to your entity. These objects allow you to integrate external objects to your project. External objects belong to predefined classes. Examples are graphs, calendars, zooms selecting zooms, etc.

For more information on External Object Classes and/or on how to define new classes please refer to the COMPONENT GUIDE manual.

5.5.4 Pages

Using the 'Pages' menu you can browse the three entity's pages, namely Header, Body and Footer, as well as other additional pages. This menu has the same functionalities as the 'Pages' toolbar. New pages can be added only using the 'Pages' menu.

Picture 187 - Pages Menu



The menu displays the names of the available pages. The current page has the 'check' symbol next to the name.

Add Page

Adds a page to the entity. To delete a page you need to delete all contained elements. When you open the entity the next time, CodePainter identifies that the page is empty and deletes it automatically.

5.5.5 Global

The 'Global' menu allows you to add general information on the current entity, such as pages title, default fonts, templates, information on the author, etc. For more information please refer to 'Globals menu'.

Picture 188 - Global Menu

Globals...

Opens the 'Global Definition' window in which you can define the generation template, the program that must be launched when the dialog window is confirmed, general information on the author, etc.

Tables...

Allows to define which working tables must be opened.

Page Structure...

Allows defining pages characteristics, such as number of rows, activation of the scroll bar, line spacing, etc.

Font...

Allows changing the default font, size, style, etc. for the current entity.

Pages Title...

Allows adding titles to defined pages.

Autonumber...

Opens the window that manages autonumbering on fields of the entity.

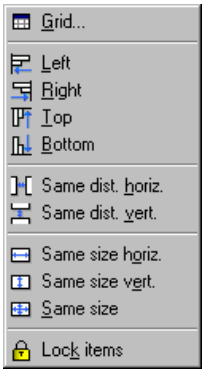
Manual Blocks...

Allows adding manual areas. In the manual areas you can add lines of code to integrate SW application functionalities.

5.5.6 Align

The *Align* menu allows you to position and resize groups of selected elements.

Alignment and re-sizing functionalities are made in relation to a master item. For more information please refer to section 'Selecting And Aligning Elements'.



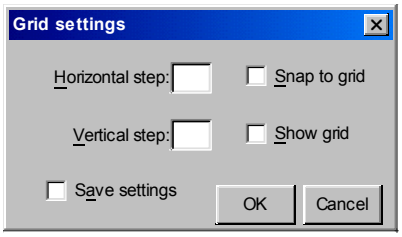
Picture 189 - Align Menu

The Align menu allows you to:

- Open the grid management
- Align to the right/left/top/bottom
- Position with the same horizontal/vertical distance.
- Re-size with the same height and width.
- Block changes for manual positioning and resizing.

Grid...

Opens the Grid Setting window to manage the grid:



Picture 190 - Grid Setting

Horizontal Step

Defines the horizontal size of the cell in pixels.

Vertical Step

Defines the vertical size of the cell in pixels.

Snap to grid

Activates the grid to position elements.

Show grid

Displays the grid.

Save setting

Saves the current setting of the grid.

Left

Aligns the left side of selected elements with the left side of the master item.

Right

Aligns the right side of selected elements with the right side of the master item.

Top

Aligns the top of selected elements with the top of the master item.

Bottom

Aligns the bottom of selected elements with the bottom of the master item.

Same dist. horiz.

Selected elements are positioned with the same horizontal distance. The distance is measured between the first and second element starting from the left.

Same dist. vert.

Selected elements are positioned with the same vertical distance. The distance is measured between the first and second element starting from the top.

Same size horiz.

Resizes the width of selected elements like the master item.

Same size vert.

Resizes the height of selected elements like the master item.

Same size

Resizes selected elements like the master item.

Lock items

Inhibits selection and change commands for selected items.

Warning

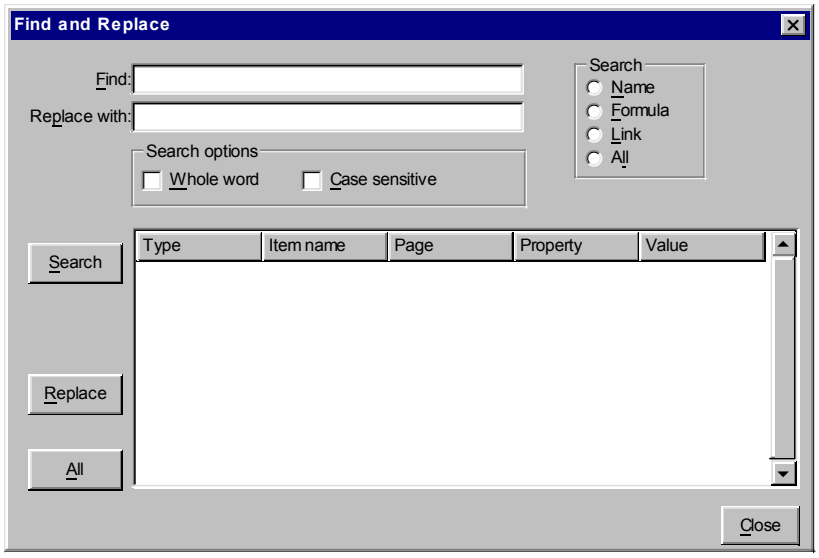
The commands are inhibited for the mouse and the keyboard while the Align toolbar remains active.

5.6 Edit Menu Advanced Options

This section details the Edit Menu options.

5.6.1 Find And Replace

Picture 191 - Find and Replace



Find

String that must be found.

Replace With

String that must replace the found string.

Whole Word

When this field is flagged the search activity must be performed on whole words only. Otherwise search results could be also substrings of long words.

Case Sensitive

The search activity matches upper and lower Case.

Search Button

Starts the search activity:

Name:

the search activity is perfomed on element names

Formula:

the search activity is perfomed on defined formulas.

Link:

the search activity is perfomed on defined links.

All:

the search activity is perfomed on names, formulas and links.

List Of Found Items

Search result listing elements found.

Replace Button

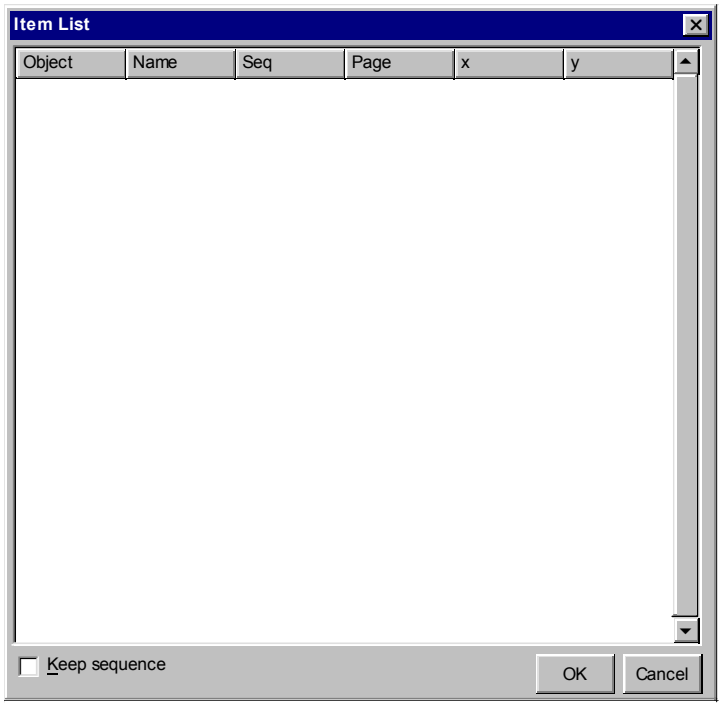
Replaces strings found with the string defined in the textbox 'Replace with' for all elements selected in the list (List of found items).

All Button

Replaces strings found with the string defined in the textbox 'Replace with' for all elements in the list (List of found items) no matter if elements are selected or not.

5.6.2 Item List

Picture 192 - Item List



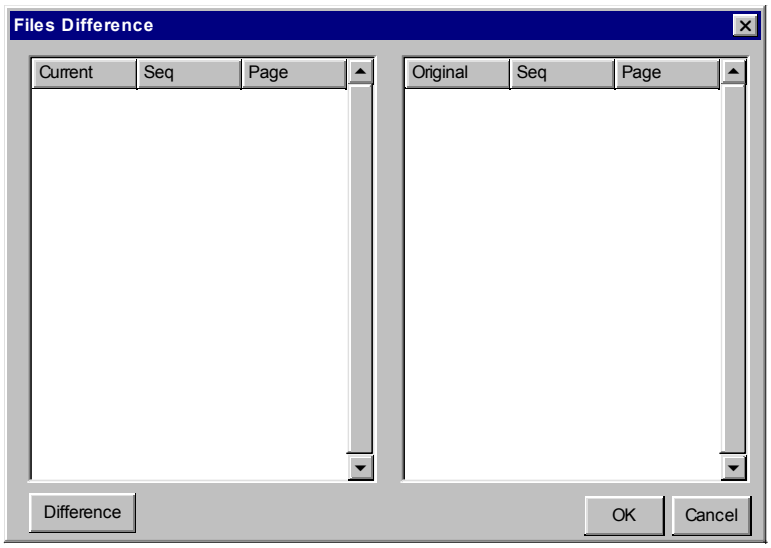
Elements List

List of elements in the dialog window. This list allows to access elements defined for this entity quickly.

Keep sequence

Flag to maintain the elements' sequence. When the flag is set and the dialog window saved elements are ordered to reflect the sequence defined in the elements' list. To order single elements right click the desired element and select the 'Sequence' option.

5.6.3 Files Difference



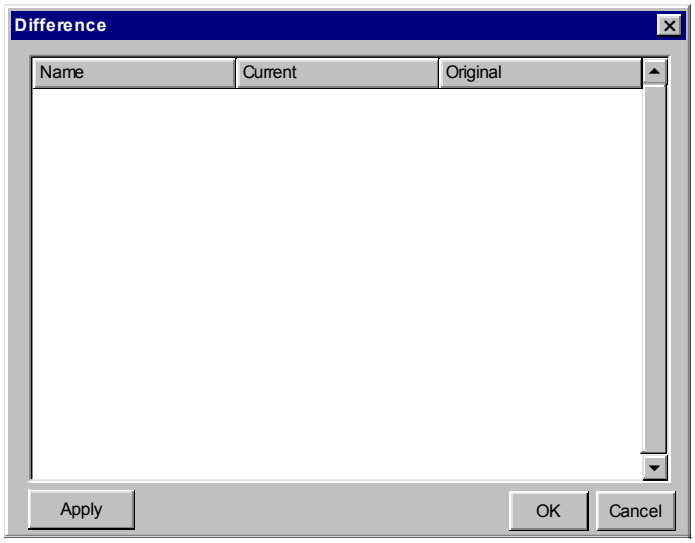
Picture 193 - Files Difference

Difference Button

You can compare two entities using the 'Detail File Differences' option.

5.6.4 Difference

Picture 194 - Difference



Difference List

List of differences between two definition files.

Apply Button

The selected difference is applied to the current definition file.

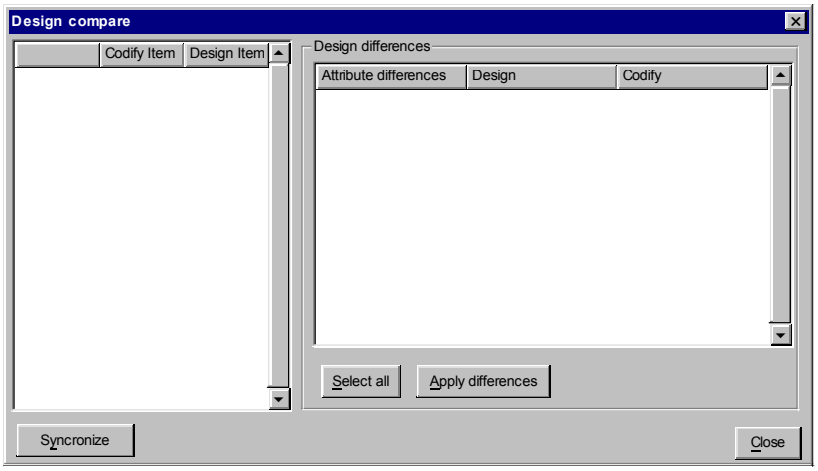
5.6.5 Design Compare

Comparing the Codify phase with the design plan allows you to synchronize entity definitions. It can be done only when the 'Bind to design entity' option in the 'Tables' or 'File and Keys' menu is set.

Once opened the dialog window displays all differences between the properties of design and codify entities, e.g. discrepancies between comments, or fields in a link, or missing fields, i.e. those fields defined in the design plan and not taken over to the Codify phase.

The comparison starts from the design entity. This involves that 'non-design' properties, which have been added during the Codify phase are not listed.

Using the 'Synchronize' or 'Apply differences' button you can synchronize some or all differences identified in the Codify phase.



Picture 195 - Design Compare

Elements List

List of elements for which the comparison has been run. The icon in the first column shows whether differences have been detected or not.

Design differences

List of differences for the element selected from the elements list. The first column briefly describes the attribute for which the difference has been identified. The second column displays the attribute value defined in the design plan. The third column displays the value defined in the Codify phase.

Select all

Selects all elements in the differences list.

Apply differences

Synchronizes codify attributes for selected elements basing on the design value.

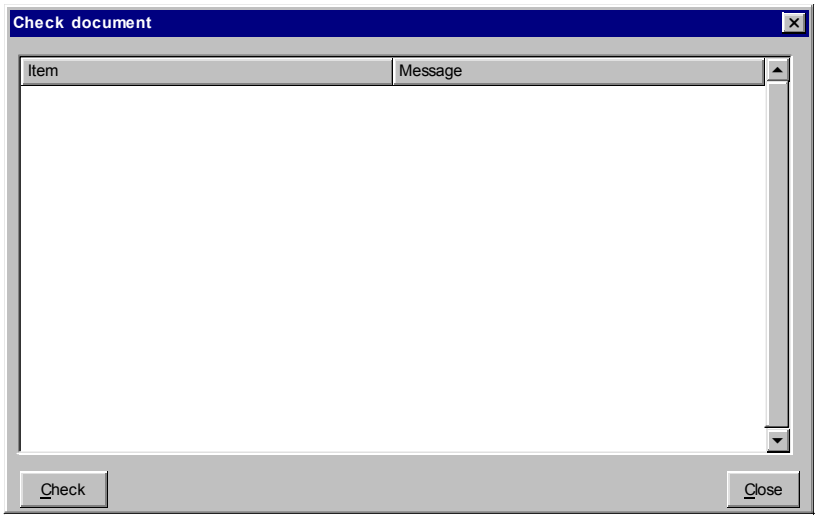
Synchronize

Synchronizes codify attributes for all elements basing on the design value.

5.6.6 Check Document

The Check dialog window executes congruence validations on definitions added using the Codify tool. CodePainter allows you to override values that usually are considered wrong, but may be required to manage specific issues. The Check dialog windows checks that no contrasting situations have been defined that could lead to malfunctions at run-time.

Picture 196 - Check Document



Errors List

Error list.

Check

Starts the validation of added definitions.

5.7 Items Menu Advanced Options

The Items menu is accessed opening the Edit - Toolbar menu or right clicking the working area and is used to add new elements to the entity in use. Let's see the various options in detail.

N. B.

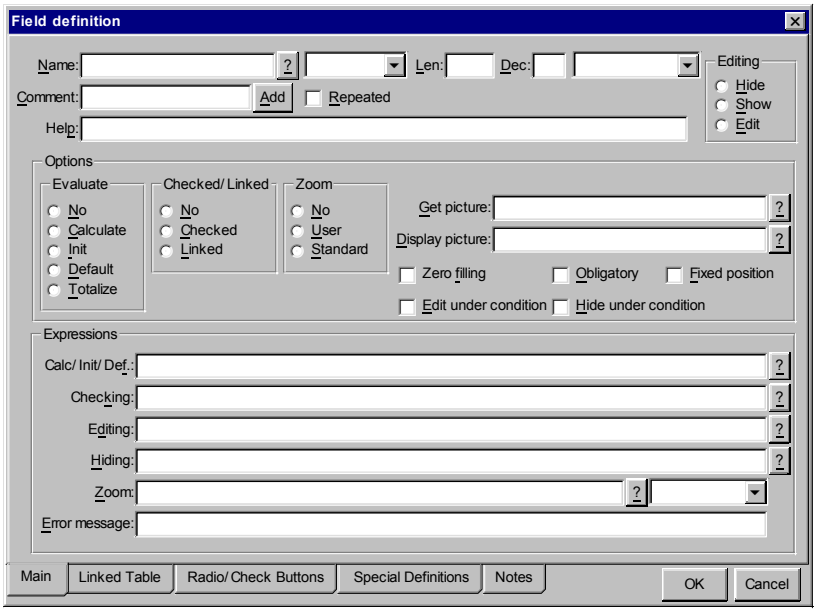
Options for variables and fields are the same. For each field a working variable is created. The working variable is saved in the field only when the Master is closed.Variables and fields are treated in the same way as they are both variables.

5.7.1 Field definition

The definition window allows to define properties for fields or variables. Here you can define how elements are displayed, or define control and calculation formulas, or mandatory fields, or whether to associate a checkbox, radio or combobox, etc.

Main Page

The first page defines the main element's properties. The window has three sections: the first groups the element's characteristics, the second activates controls or calculations and in the third expressions to execute controls are defined.



Picture 197 - Field Definition: Page 1

Name

Name of the field or variable.

This name must follow the rules for the construction of variable identifiers for the target environment. Typically it must start with a letter and cannot have blanks, commas or points. Some environments also have length limitations.

CodePainter creates for each element (fields and variables) a working variable that takes on the prefix 'w_'. Fields are initializated when the record is read. The user edits the record using the values in the working variables. When the record is saved the contents of 'w_' variables is saved in the corresponding database fields.

Variables that do not have values stored in the database are initializated when the record is loaded. The working variable either does not contain any value (i.e. blank for strings, or 0 for numbers), or it contains the value defined in the 'Init' expression. During the editing phase they behave as fields as defined above.

When the selected element is a field you can list all fields of the associated table clicking the '?' button. In the same way when the element is a variable you can list all variables that CodePainter identifies as useable, but have not yet been placed on the screen.

Field Type

Kind of element.

This combobox defines the kind of element you are editing. Implemented element types are those typically used by business/ commercial applications, i.e. character, numeric, date, logical, memo. The element type here must match with the associated database field. You can define different types only if the type is compatible with what is read by the files: shorter strings, numbers having less digits, etc.

Len

Element's length.

Here you define the length of the variable associated to the element. The length is expressed in characters for strings and in digits for numbers. For logical elements, dates and memos the default length is used.

Dec

Decimal digits for the element.

For numeric elements you can define the number of decimals required. This number is included in the total length.

Key/Index

Defines whether an elements is part of a key or of a search index.

Elements belonging to the primary key can be edited only when a new record is created. They cannot be edited in the change mode.

In the query mode all elements belonging to primary keys or indexes are enabled. When the user enters a value the search activity is quick, because it is executed on the index and not systematically scanning the entire file.

Normally this option is defined for fields and not for values.

Comment

Brief comment on the element.

This comment is displayed in the element list of the edited entity. Fields are automatically initializated using the description defined in the design plan. For variables this must be set by the programmer.

Using the 'Add' button you can add a string next to the element and define the text and comment. This allows you to obtain a prototype for working variables quickly, which is in line with the dialog window.

Add

Using the 'Add' button you can add a string next to the element and define the text and comment. This allows you to obtain a prototype for working variables quickly, which is in line with the dialog window.

Repeated

Defines whether elements are repeated or not.

Repeated elements are stored in a temporary file. Each body row has its value. When you move to a new row you can edit the value associated to the selected element.

Repeated elements must not necessarily be within the grid. You can place them outside the grid setting the 'Fixed Position' option. These elements are displayed only once but as you move the cursor from one row to the other the fixed element displays the value associated to the highlighted row.

Editing

Editing status of the element.

Elements have three editing statuses:

Hide

Hidden elements are not displayed. They are used for calculations or as supporting variables.

Show

Showed elements are displayed on the screen but cannot be edited. They are used in links as return variables, to store calculation results or as support variable that must be checked by the user.

Edit

Editable elements can be changed by the user.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Evaluate

Defines whether the element is calculated.

Elements can be edited by the user or calculated by the procedure. When you define options involving calculations you need to define the expression in the 'Calc/Init/Def.' option in the 'Expressions' area. There are many calculation options:

'No'

When it is set the element is not calculated. Variables are initialized (blank string for characters, 0 for numbers, etc.).

'Calculate':

The element is calculated. The defined expression is checked every time a value affecting the calculation result is entered in the dialog window. When the calculation is simple this strategy works smoothly. When the calculation implies a slow routine you need to limit re-calculations. In the 'Calc/Link depends on' option you need to define which elements trigger the re-calculation. The calculation will be executed only when one of these elements changes. Calculated elements are normally 'shown' and not 'editable'. To make elements editable the calculation must not be re-executed at each input, but only when trigger elements are changed. Otherwise the calculation is re-executed also when a value is entered, thus overriding the input.

'Init':

The calculation is executed only once when the element is edited. This option is used to define a different default value then the one given automatically by the system.

'Default':

In some cases fields cannot be initializated together with the dialog window. This may happen because the field initialization depends upon some values that the user must input first.When the 'Default' option is set the calculation is executed the first time the cursor enters the element. In the initialization expression you can therefore define all elements that the user must enter first.

'Totalize':

When this option is set the elements sums up all values in the body rows. In the calculation expression you need to define the name of the element that must be totalized.

Checked/Linked

Defines whether the element is controlled or linked to a file.

To control the data that the user has input you need to define one of the options in the table below. If you select an option that requires a control you need to define the validation formula in the 'Checking' line in the 'Expressions' area.

'No':

The element is not controlled. Any value is accepted.

'Checked':

Any user input is controlled executing the 'Checking' formula. When the result is TRUE the value is accepted, otherwise an error message is displayed, the element is emptied and the cursor positioned on the element. The standard error message can be changed defining the 'Error message' expression.

'Linked':

The control activity involves a database table. The input value is searched in the database and is accepted only if the linked table contains a record that has the value as primary key. Detail on how this search activity is performed are defined in the 'Linked Table' page. 'Linked' elements can be also controlled by a 'Checking' formula. The value is first searched on the linked table. If found and returned the 'Checking' formula is executed.

Zoom

Defines whether there is an active zoom on the element.

Zooms are procedures run when the user presses the function key F9 when the cursor is on the related element.

'No'

No action is performed.

'User'

The expression defined in the 'Zoom' formula in the 'Expressions' area is executed.

'Standard'

When an element is liked to a table CodePainter creates a standard zoom to display and search data in the liked table. Parameters defined in the 'Linked Table' page influence the standard zoom. The 'Zoom' expression is used and a specific zoom configuration executed.

Get picture

Format of the input element.

In this option you can define the format that must be applied when the element is inserted.

You need to define an expression that is checked at run-time. Constant values must be defined between apexes.

Typical formats are:

"999.99": for numeric values.

"XXXXX": for characters.

Repl("!", 20): forces upper case in 20 characters strings.

When this option is not defined and you define a display format the input uses the latter. The '?' button creates a standard format for the element type.

Display picture

Format of the displayed element.

In this option you can define a format to be applied when the value is displayed. You need to define an expression that is checked at run-time. Constant values must be defined between apexes.

Typical formats are:

"999.99": for numeric values.

"XXXXX": for characters.

Repl("!", 20): forces upper case in 20 characters strings.

When no input format is defined the format defined for display is used also when data is input.

Zero filling

The value takes on zeros on its left until the value is filled up.

When the 'Autonumber' option is used on character keys entries are created as '00001', '00002' etc. The Zero filling option makes it easier for the user who is not required to manually fill the field with zeros. The system automatically adds zeros until the field length is reached.

This option works only is the entered value is numeric and does not start with '0'.

Obligatory

Mandatory element.

An editing phase cannot be terminated as long as no value has been defined for this element. When the user presses the function key F10 and no value has been entered an error message 'Obligatory field' is displayed. The cursor is positioned on the element and the editing mode is kept so that the user can enter a value.

Fixed position

Repeated elements in a fixed position.

In complex programs the grid associated to the body may be too small to contain all required values. You can fix the position of body's elements. The element is displayed only once in the dialog window. The element takes on different values depending on the active row.

During data input you will first input all fields in the grid, pass on to fixed elements and then go on to the next row.

Edit under condition

Defines whether the element can be edited only under specific conditions.

Elements may be editable only when specific conditions occur. To define conditions set the 'Edit' option to edit and define the 'Editing' formula in the 'Expressions' area that must be checked before the cursor is positioned on the element. When the logical expression returns a TRUE value the element is edited. Otherwise the cursor is passed on to the next element.

Hide under condition

Hides elements.

Elements may be hidden under specific conditions. To define these conditions set the 'Edit' or 'Show' option and define the 'Hiding' formula in the 'Expressions' area that must be checked. When the logical expression returns a TRUE value the element is hidden and therefore not displayed on the dialog window.

Calc/Init/Def.

Formula used for calculations.

When calculations formula are defined in the 'Evaluate' option the expression is used to calculate the element's value.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Simple Calculation

Define three elements: PRZART, QTAART and TOTART. The total (TOTART) must be calculated multiplying price and quantity.

The calculation formula must be defined as follows:

```
w_PRZART * w_QTAART
```

Editable Calculation

Consider the same three fields: PRZART, QTAART and TOTART. The total (TOTART) must be calculated and the element must be editable so that the user can adjust roundings manually.

The calculation formula must be defined as follows:

```
w_PRZART * w_QTAART
```

Further, in the 'Calc/Link depends on' option you need to define:

```
w_PRZART
```

```
w_QTAART
```

The calculation is executed only when the value for one of these fields changes.

Checking

Control formula for the element.

When elements are controlled or linked this expression is used to check whether values are acceptable or not. When the logical expression returns a TRUE value the value is accepted. When the returned value is FALSE the value is refused.

When the value is refused the default value is input in the element depending on the element type, an error message is displayed and the cursor passes on to the next element. The default message is 'Value not accepted', but you can change it defining the desired formula in 'Error message'.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Controlled Field

The field PRZART accepts only values greater or equal to zero:

```
w_PRZART>=0
```

Editing

Formula to activate the conditional editing.

This formula defines whether the field is editable or not. To define conditional editing you need to define the 'Edit' option in the 'Edit' area, set the 'Edit under condition' checkbox, and define the 'Editing' formula in the 'Expressions' area. When the logical expression returns a TRUE value the element is edited. When the returned value is FALSE the fields is disenable.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Hiding

Formula to hide the element.

This formula defines whether the field must be hidden or not. To define these conditions set the 'Edit' or 'Show' option, set the 'Edit under condition' checkbox and define the 'Hiding' formula in the 'Expressions' area. When the logical expression returns a TRUE value the field is hidden. When the returned value is FALSE the field is displayed.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Zoom

Zoom formula.

When elements have an active zoom this formula is used to define what must be executed when the user presses the function key F9.

When the 'Zoom' option is set to 'User' in this formula you need to define what must be executed. The combobox next the option define the type of zoom. Zoom types are:

'User program': the formula is a program line.

'Mask': the formula defines the dialog window that must be executed as zoom.

'Batch': the formula defines a routine created using the Routine Painter.

'UTK Object': the formula contains the output configuration name that must be displayed as zoom.

When the 'Zoom' option defines a 'Standard' zoom the formula can contain the name of a specific configuration that must be loaded in the zoom window.

Error message

Text for the error message.

When the element is controlled the user could enter values that are refused. In these cases the error message 'Value is not correct' is displayed. In this option you can define custom error messages for each element.

Checked/Linked

Defines whether the element is controlled or linked to a file.

To control the data that the user has input you need to define one of the options in the table below. If you select an option that requires a control you need to define the validation formula in the 'Checking' line in the 'Expressions' area.

'No':

The element is not controlled. Any value is accepted.

'Checked':

Any user input is controlled executing the 'Checking' formula. When the result is TRUE the value is accepted, otherwise an error message is displayed, the element is emptied and the cursor positioned on the element. The standard error message can be changed defining the 'Error message' expression.

'Linked'

The control involves a database file. The input value is searched in the database and is accepted only if the linked table contains a record that has the value as primary key. Detail on how this search activity is performed are defined in the 'Linked Table' page. 'Linked' elements can be also controlled by a 'Checking' formula. The value is first searched on the linked table. If found

and returned the 'Checking' formula is executed.

Evaluate

Defines whether the element is calculated or not.

Elements can be edited by the user or calculated by the program. When the calculation option is selected you need to define the expression that produces the result in the 'Calc/Init/Def.' option in the 'Expressions' area. There are many calculation options:

'No'

When 'No' is set the element is not calculated. For variables the blank value is initializated (blank string for characters, 0 for numbers, etc.).

'Calculate':

The element is calculated. The defined expression is checked every time a value affecting the calculation result is entered in the dialog window. When the calculation is simple this strategy works smoothly. When the calculation implies a slow routine you need to limit re-calculations. In the 'Calc/Link depends on' option you need to define which elements trigger the re-calculation. The calculation will be executed only when one of these elements changes. Calculated elements are normally 'shown' and not 'editable'. To make elements editable the calculation must not be re-executed at each input, but only when trigger elements are changed. Otherwise the calculation is re-executed also when a value is entered, thus overriding the input.

'Init':

The calculation is executed only once when the element is edited. This option is used to define a different default value then the one given automatically by the system.

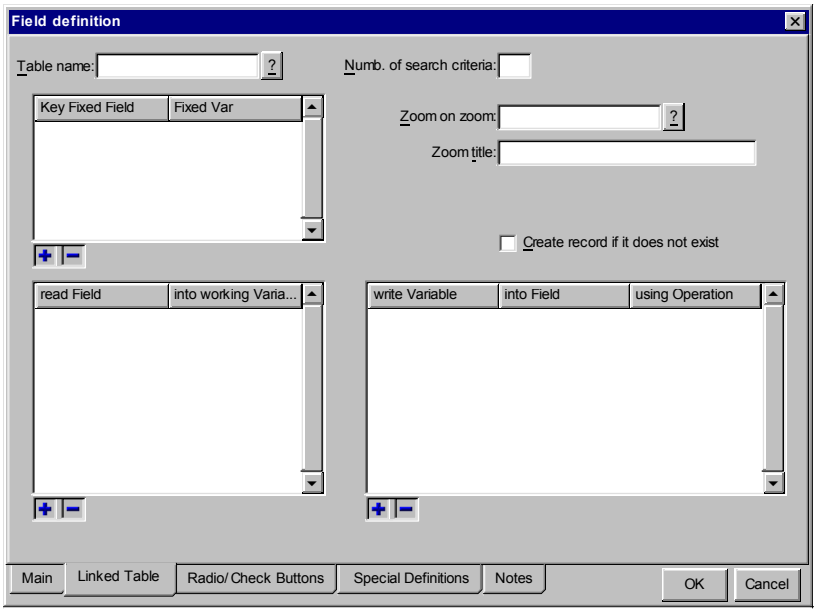
'Default':

In some cases fields cannot be initializated together with the dialog window. This may happen because the field initialization depends upon some values that the user must input first.When the 'Default' option is set the calculation is executed the first time the cursor enters the element. In the initialization expression you can therefore define all elements that the user must enter first.

Linked Table Page

In the second page you need to define the rules for the relationship between an element and a file record. The prototype defaults automatically what you defined in the Design phase.

When you define links in the prototype without going back to the design plan the relationship is not programmed in the database. This means that referential integrity for this relationship is not established. Values are checked only when they are entered in the dialog window.



Picture 198 - Field Definition: Page 2

Table name

Name of the related table.

Define the name of the related table. The value is searched in the related table basing on the parameters defined in this dialog window.

The table name is added to the workfiles list so that when the program is run the system checks the existence of the related table. Once you enter the table name, all lists helping the programmer to select fields' names are activated. Moreover, the procedure name that must be activated for the 'zoom on zoom' option is initialized. This procedure is a zoom that can be executed pressing F9 within an active zoom.

The '?' button displays the list of tables available in the database.

Numb. of search criteria

Defines the number of search criteria.

Using this option you can change the way records are searched in the related table. SQL sentences use fields defined in the 'Read Field ... into working Variables' area as search criteria. If the first search goes wrong the SQL sentence uses the second field in the list and so forth as long as there are search criteria.

Define an element 'ARTORD' which is related to the 'Items' file. Your search criteria is CODART and you need to return the field DESART.The standard search criteria is:

```
CODART=w_ARTORD
```

If you define the 'Read Field' list as follows:

```
CODART, w_ARTORD


DESART, e_DA_ORD
```

and the first search criteria does not lead to any result, no error message is displayed and the search activity is executed:

```
DESART=w_ARTORD
```

Only if the second search goes wrong as well the defined value is refused.

The example highlights how the current fields is searched as code first and as description afterwards. If the result is given by the second search criteria values are returned in the defined sequence.

This kind of alternative search is performed for the number of defined criteria, following the list of fields that must be returned. Fields must always be of the same type, you cannot mix e.g. numbers and characters.

Fixed fields belonging to the key are used also used for alternative searches.

Zoom on zoom

Procedure which is executed pressing F9 on an active zoom.

When the user presses F9 a zoom window on the related table is opened. If searched values are not found pressing F9 again (on the active zoom) a new window is opened that allows managing the linked file.

This option allows to define the procedure that must be executed to zoom on an active zoom. Normally the standard procedure to manage the related table is defined. This means that when you select the file reading design definitions, this field is initializated by CodePainter.

Zoom title

Contains the title of the zoom window opened.

Create record if it does not exist

Defines whether a record must be created on the related table.

Linked fields must find a related value in the linked table. This is true for all elements input by the user, because they are validated and not found values are refused.

When linked elements are hidden values are never entered and thus never checked. For example an element may be calculated basing on another element on the screen but linked to another table. It is therefore possible to write totals in a record that does not exits, i.e. the related record is missing.

Using this option the search activity can go on. A new record is created and filled in the related table using the list of fixed keys of read fields and totals. Returned values are used to attribute values to fields.

This happens especially when you have Parent/Child relationships whereby totals are required to update the Child entity.

Key Fixed Field

Fields building the primary key, excluding the current field.

This list is used when the primary key of the related table is composite.

The search activity is started when the user enters a value in the current field, which in turn is placed in line with the first element in the list right under the value. Primary key fields are placed next to the corresponding values in the dialog window according to the logic defined in this list. For each field there is value, which has been read by a working variable.

Composite Key

The primary key of the linked table is made of the fields: CODMAG and CODART. In the Dialog Window there are two elements, MAGORD and ARTORD asking the user to input the warehouse code and the item code on which he/she wants to work.

In the ARTORD element you need to define:

```
CODMAG, w_MAGORD
```

In 'read Fields' define:

```
CODART, w_ARTORD
```

The search activity in the related table will have matches of the following kind:

```
MAGART=w_MAGORD and CODART=w_ARTORD
```

The standard zoom uses these fields as filter on the table so that only a selection of the table is displayed and not the entire related table. In the example above after that the user enters a value for the warehouse the zoom window will display only the items in that warehouse.

read Field

List of fields that must be read from the related table.

This list drives the relationship with the related table. When the user enters a value in the current element the search activity is started. The selection criteria is given by the first field in this list and the fixed fields defined in the 'Key Fixed Fields' area (optional).

Example

If the related file has the key CODART and the current element is ARTORD the search activity is as follows:

```
CODART=w_ARTORD
```

If the entered value is found the link is successful and all fields defined in the list are read and values are returned to the corresponding working variables.

If the value is not in the file the error message 'Value is not correct' is displayed and the element is reset.

The search activity can also find partial values, e.g. the user in enters 'Smith' and the value in the file is 'Smith John'. In these cases the value entered in the field is completed with the value found in the table.

When the search activity finds more values (e.g. the user enters 'Smith' and the values in the table are 'Smith John' and 'Smith Bob') a zoom window containing the list of matches found is opened and the user can select the desired record.

When the search activity goes wrong but more search criteria are defined, alternative searches are executed as defined in 'Numb. of search criteria'.

Write Variable

This list defines totalization transactions towards the related table.

This list is made of three columns: the first column defines the variables that will be summed up, the second the fields that will be increased in value and the third the operations that must be performed.

In the third column you can define the constants '+', '-', and '=' to get totals, reversals or a data entry. You can also define fields with one character that contain the transaction that must be executed. When this value is blank no transaction is performed. Transaction types must be stored in a field and not in a variable because CodePainter requires this information to perform reversals in case of changs or deletions.

Example

Your application has two tables, namely 'Items' with its fields CODART, DESART and QTAART, and 'Orders' with the fields ARTORD and QTAORD.

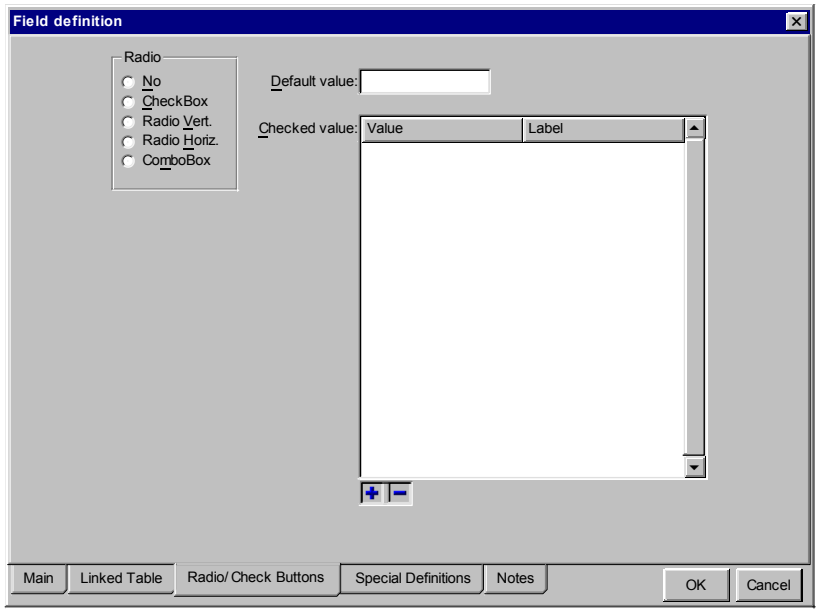
You want to sum ordered quantities in QTAART. Define the 'Write Variable' list for the element ARTORD as follows:

```
w_QTAORD, ARTORD, +
```

Radio/Check Buttons Page

The third page contains parameters to change the standard input/ output textbox in other input/ output structures.

Picture 199 - Field Definition: Page 3



Radio

To display the element as checkbox, radio or combobox.

Fields and variables are created as textboxes. This option allows you to change the method in which data is entered and displayed.

Default value

Defaulted value when the user does not select any value.

Value

List of variables and labels.

This list is made of two columns: the first defines the value which is given to the element, the second the label that must be displayed.

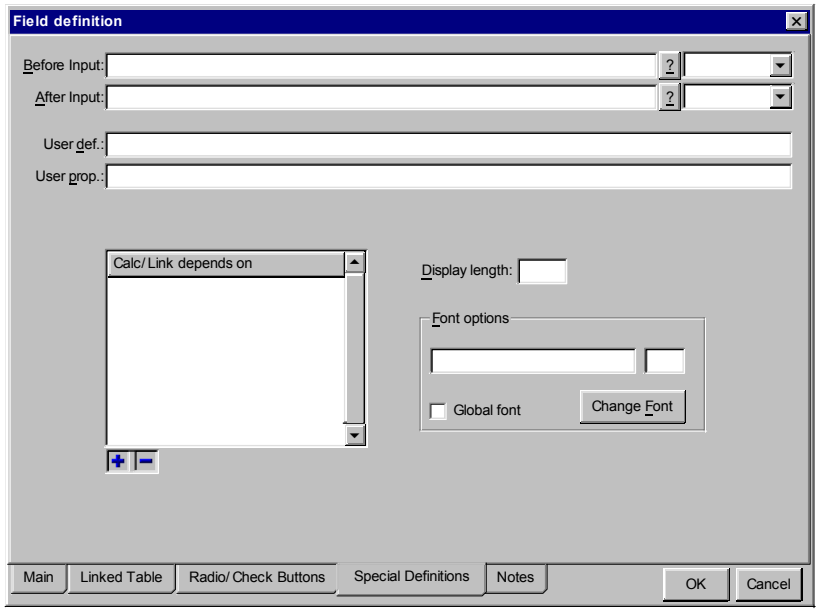
Values must be defined as constants in the target language. Strings must therefore be written between apexes. Labels are always text constants therefore they do not need to be written between apexes.

Special Definitions Page

The fourth page defines properties, which are used seldom.

Rather then using 'Before' and 'After' it is advisable to use the structured options in the first page. 'User Def' require customized templates. 'Depends on' are used for optimizations.

Picture 200 - Field Definition: Page 4



Before input

Program that must be executed before the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low-level and cannot be defined so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

After input

Program that must be executed after the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low-level and cannot be defines so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

User def.

Free definition.

Could be used by a custom template.

User prop.

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to add definitions to the ones generated by CodePainter.

Example

```
forecolor=RGB(255, 0, 0)
```

Calc/Link depends on

List defining the elements on which the calculation execution or the link execution depends on.

Calculations and links are normally executed every time a value is entered. This strategy works smoothly if the calling routine is quick. If calling routines take long to be terminated you need to limit the number of calls.

Defining working variables in this list, calculations and associated links are executed only when the value for one of these variables changes.

This list is also used for calculated elements, which can be edited. Elements will remain editable, but they will be recalculated only when the value for one of these variables changes. Here you should define all calculation parameters associated to the element.

Using this method you avoid executing many line codes. Nevertheless checking the change of the value is costly and therefore this option should be activated only when calculations are complex.

The same applies to link executions.

Example

A field is linked to the 'Items' file. The field is returned to the linked 'VAT' file. When item's data is entered the VAT value must be defaulted. The field must be editable so that the VAT value can be returned by the first link.

You need to define the 'item number' field in the 'Depends on' list of the 'VAT' field. When the user selects an item the link with the VAT file is re-executed. The VAT value can be freely changed. If the user goes back to the 'item number' field and enters a new value, the link to the VAT file is re-executed.

Display length

Length used to display the element.

This option is used so that CodePainter calculates a length, which is different from the one given by the element parameters.

Change Font

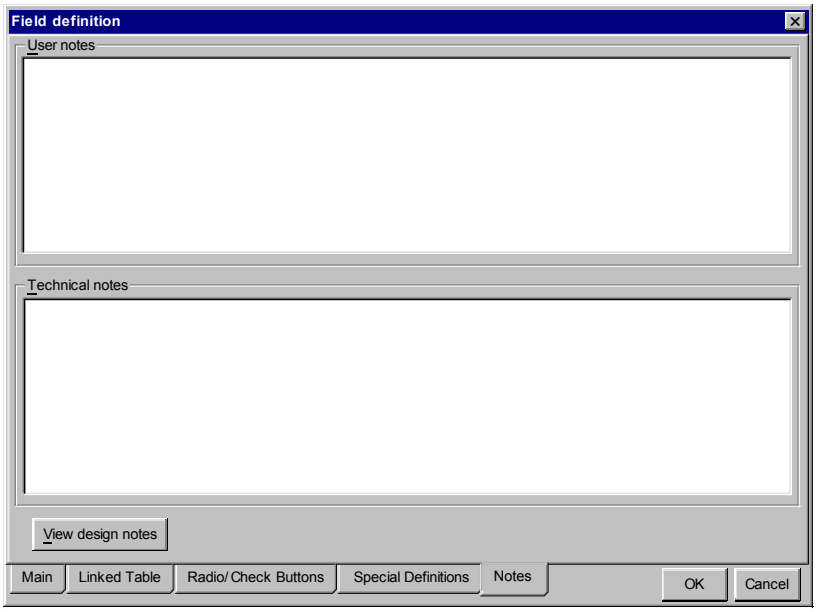
Defines the elements font. If no font is defined the dialog widow's default font is used.

Global font

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.



Picture 201 - Field Definition: Page 5

User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

Technical notes on the element.

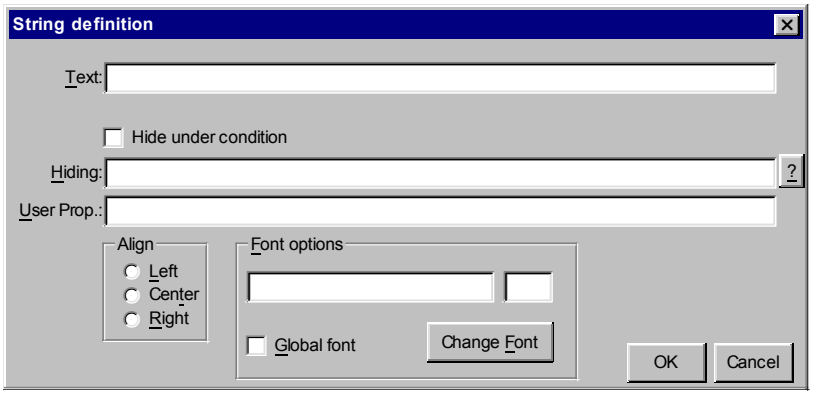
These notes are used to create technical documentation.

View design notes

5.7.2 String Definition

Dialog window to define the characteristics of comment strings.

Picture 202 - String Definition



Text

String text.

Hide under condition

Hides the string.

To display strings only when specific conditions are met you need to define the hiding formula in the 'Hiding' textbox. When the returned value is TRUE the string is hidden, and is not displayed.

Hiding

Formula defining the conditions to hide the string.

This formula defines whether the string must be hidden or not. To hide the string under given conditions you need to set the checkbox 'Hide under condition' and to define a logical expression in this field. When the result is TRUE the string is hidden, when the result is FALSE the string is displayed.

Clicking the '?' button the list of elements in the dialog window is displayed. These elements can be used to define expressions. Please note that while in the Editing mode CodePainter stores values in working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

User prop

Control properties defined by the programmer.

Elements on the screen are generated as a 'control'. This option allows you to add other specific controls.

Example

```
forecolor=RGB(255,0,0)
```

Align

String alignment.

Defines how the string must be positioned within the string box.

'Left'

Aligned to the left.

'Center'

Centered.

'Right'

Aligned to the right.

To set the window layout you should align strings to the right. You should not align to the left and create boxes that exactly fit the strings. The end-user may use a different font from the one defined and therefore strings may be bigger or smaller. In these cases strings would be no longer aligned.

When you need to translate the application you need to align strings to the right as words in other languages may be longer or shorter. Using CodePainter you can automatically translate dialog windows and set a different language for each user.

Prototypes are created using these principles, i.e. alignment to the right and field space longer than required.

Change Font

Defines the strings' font. If no font is defined the dialog widow's default font is used.

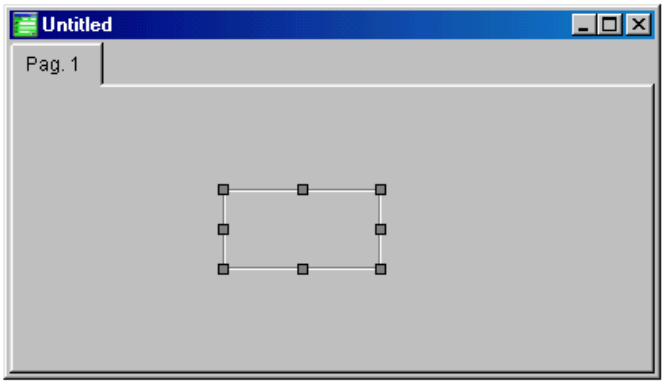
Global font

When selected the font defined in the 'Global Font' menu is applied to the current element.

5.7.3 Box

This option allows you to add a Box element to the entity in use.

Picture 203 - Box



Boxes are used to graphically divide the form. Boxes can be used as vertical or horizontal lines or rectangles.

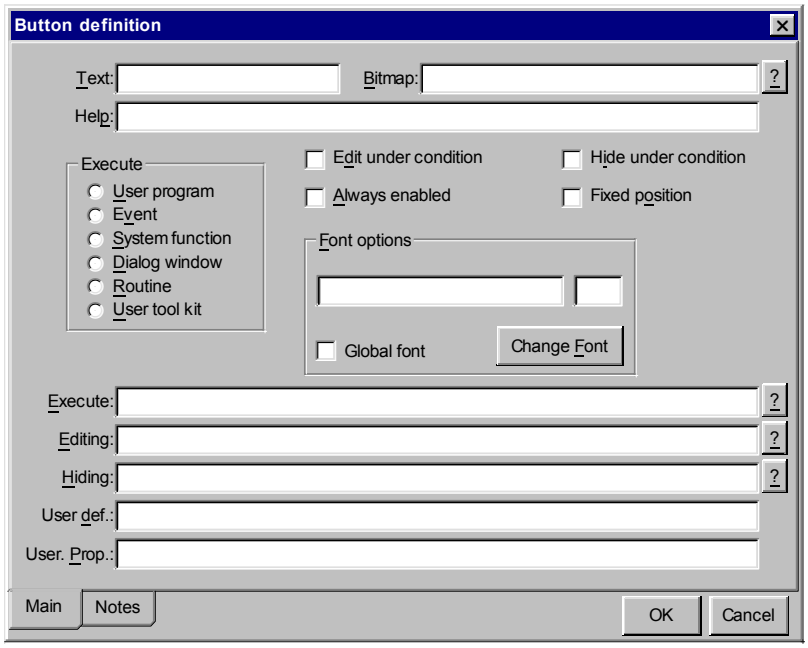
Boxes are changed into lines '*collapsing*' them using the mouse or **<Shift>** + **<Cursor Keys>**.

5.7.4 Button Definition

Dialog window defining button options. Buttons are added into dialog windows to execute procedures such as reports, additional dialog windows, etc.

Main Page

In the first page you can define how the button is displayed, activated and its functions.



Picture 204 - Button Definition: Page 1

Text

Text that must be displayed inside the button.

Bitmap

Bitmap displayed inside the button.

Help

Short 'Help' text.

The text defined in this option is used to associate a tooltip to the element. When the user positions that mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Execute

Defines how CodePainter must interpret the execution command line when the button is clicked.

'User program'

Executes a program defined by the developer.

'Event':

Notifies an event.

'System function':

Executes a system function.

'Dialog window':

Opens a dialog window created using the Dialog Window Painter.

'Routine':

Executes a routine developed using the Routine Painter.

'User Tool kit':

Executes an output created using the run-time framework.

Depending on the command type defined in 'Execute' different values are required. 'User Program' is copied in the source. 'Event' requires the name of the event that must be notified. 'System function' requires the name of the function that must be activated. 'User tool kit' requires the name of the query followed by the output format name. In all other cases the name of the file that must be executed is required

Edit under condition

Defines whether the button is enabled only under specific conditions.

To enable buttons only when specific conditions are met you need to define a formula that must be validated before the cursor is positioned on the element.

Setting this option the 'Editing' formula in the textbox is activated. The formula must be a logical expression. When the returned value is TRUE the button will be enabled, otherwise the cursor is passed on to the next element.

Hide under condition

Hides the button.

To hide buttons under specific conditions you need to define the 'Hiding' formula in the textbox. The formula will determine whether the button must be displayed or hidden. If the logical value TRUE is returned the element is hidden otherwise the button is displayed.

Always enabled

The button is enabled in 'Editing' or 'Query' modes.Buttons are used to access procedures that are bound to the main dialog window. To use these procedures may make sense in the 'Editing' and/or in the 'Query' mode only. E.g. a button executing a report should always be enabled.

Setting this flag the button will be enabled in the 'Editing' as well ad in the 'Query' mode. When the flag is not set the button is enabled in the 'Editing' mode only.

Fixed position

The button is repeated in a fixed position.

In complex programs the grid associated to the body may be too small to contain all values. You can define that a specific button in the body is in a fixed position outside the grid and that it is displayed only once in the window. The button is still linked to the row but is displayed only once.

During data input you will first input all fields in the grid, pass on to fixed elements and then go on to the next row.

Execute

Defines what must be executed when the button is pressed.

The meaning of this line depends on what has been defined for the radio button 'Execute'. The button on the left allows quick access to the following definitions:

'User program':

The code line is copied into the source. Using the '?' button you can access the list of variables contained in the dialog window.

'Event':

Defines the name of the Event that must be notified.

'System function':

Executes a CodePainter function. Clicking the '?' button the list of available functions is displayed.

'Dialog window':

Opens a dialog window created using the Dialog Window Painter. The '?' button lists all Dialog Windows available in the project.

'Routine':

Executes a routine developed using the Routine Painter. Beside the routine name you can also define in brackets the parameters that must be passed on to the routine.

'User Tool Kit':

Executes an output created using the run-time framework, typically a query created using the Query Painter. A dialog window is opened in which you can select the kind of output required: preview, report, word, excel or graph. To execute the output you need to add a comma after the query name and digit the output name.

Editing

Formula to set conditions to allow the 'Edit' mode.

This formula defines whether the button is active or not. To define conditions you need to set the 'Edit under condition' option and define a logical expression. When the expression result is TRUE the button is enabled. When the result is FALSE the button is disenabled.

Clicking the '?' button the list of all elements in the dialog window is displayed. These elements can be used to define the expression. Please note that during the Editing mode CodePainter stores the values of working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

Hiding

Formula to hide the button.

This formula defines whether the button must be hidden or not. To hide the button you need to set the checkbox 'Hide under condition' and to define a logical expression in the formula. When the result is TRUE the button is hidden, when the result is FALSE the button is FALSE.

Clicking the '?' button the list elements in the dialog window is displayed. These elements can be used to define the expression. Please note that while in the Editing mode CodePainter stores the values of working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

User def

Free definition.

This option can be used by custom templates.

User prop

Control properties defined by the programmer.

Elements on the screen are generated as a 'control'. This option allows you to add other specific controls.

Example

```
forecolor=RGB(255, 0, 0)
```

Change Font

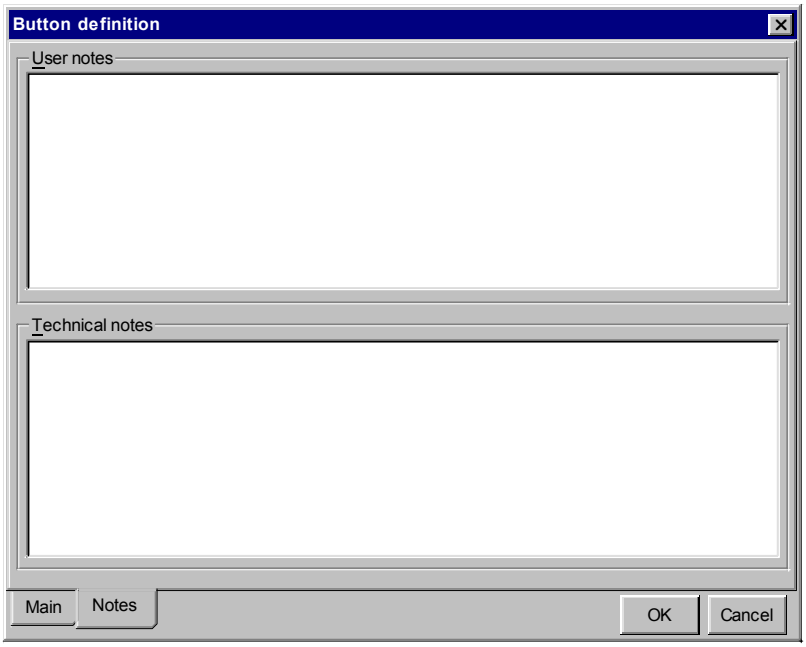
Select the font to be applied to this element. When no font is defined the default font used by the dialog window is applied.

Global font

Notes Page

Notes that are used to create the documentation.

Picture 205 - Button definition: Page 2



User Notes

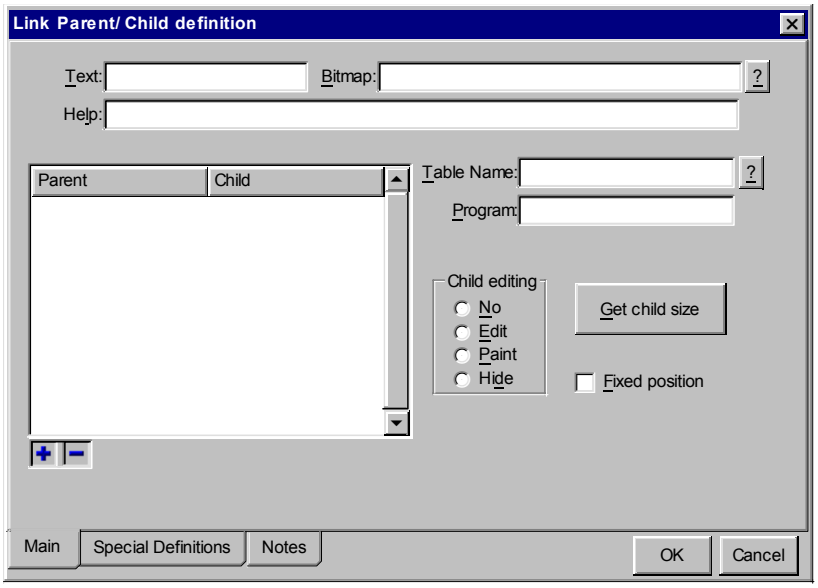
Notes on the element.These notes are used to create user documentation.

Technical Notes

Technical notes on the element.These notes are used to create technical documentation.

5.7.5 Link Parent/Child Definition

Main Page



Picture 206 - Link Parent/Child Definition: Page 1

Text

Text displayed within the button.

Bitmap

Bitmap displayed within the button.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Parent/Child

List of key fields that bind the Parent to the Child.

This list defines all fields that relate the Parent object to the Child object. CodePainter will pass on these fields when a search activity is required or in Load mode when the Child requires the Parent's key.

Table Name

Name of the Child file.

Program

Name of the Child file's managing procedure.

Child editing

Child's editing mode.

'No'

The Child is activated when the button is clicked.

'Edit'

The Child's managing procedure is integrated in the Parent's dialog window.

'Paint'

The Child is in a different window, but is activated with the Parent window.

'Hide'

The Child is hidden. Data in the Child is canceled when it is canceled in the Parent. Data in the Child cannot be changed.

Get child size

Measures the Child's editing window and adjusts the Parent's window so that the Child fits in.

This button is used when the 'Child editing' option is set to 'Edit'.

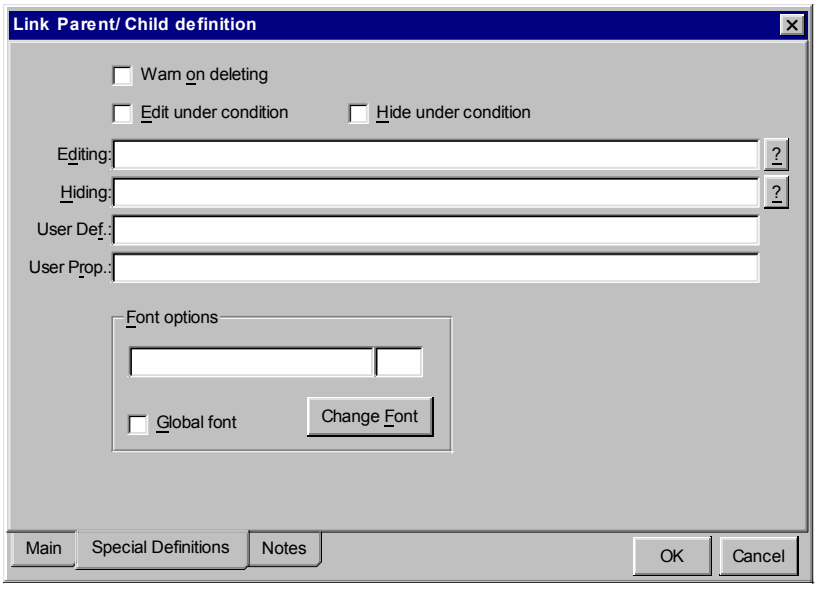
Fixed position

Repeated button in a fixed position.

In complex programs the grid associated to the body may be too small to contain all required values. You can fix the position of the button in the body. The button is displayed only once in the dialog window, and is still bound to the row taking up only reduced space.

During data input you will first input all fields in the grid, pass on to fixed elements and then go on to the next row.

Special Definitions Page



Picture 207 - Link Parent/Child Definition: Page 2

Warn on deleting

When you are about to delete records of the Parent entity this option warns you that the Child entity data will be deleted as well.

Behind small windows there may be a number of fields hidden in a Parent/Child link button. To avoid deleting this hidden data by mistake you can set this checkbox. Before deleting the system checks the Child's contents. If data is found a message asking you to confirm the delete command is displayed.

Edit under condition

Defines whether the button is enabled only under specific conditions.

Buttons may be enabled only when specific conditions occur. To define conditions set the 'Edit' option to edit and define the 'Editing' formula in the textbox that must be checked before the cursor is positioned on the button. When the logical expression returns a TRUE value the button is enabled. Otherwise the cursor is passed on to the next element.

Hide under condition

Hides buttons

Buttons may be hidden under specific conditions. To define these conditions set the 'Edit' or 'Show' option and define the 'Hiding' formula in the textbox that must be checked. When the logical expression returns a TRUE value the button is hidden and therefore not displayed on the dialog window.

Editing

Formula to activate the conditional enabling.

This formula defines whether the button is enabled or not. To define conditional enabling you need to define the 'Edit' option in the 'Edit' area, set the 'Edit under condition' checkbox, and define the 'Editing' formula in the textbox. When the logical expression returns a TRUE value the button is enabled. When the returned value is FALSE the button is disenable.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Hiding

Formula to hide the button.

This formula defines whether the button must be hidden or not. To define these conditions set the set the 'Edit under condition' checkbox and define the 'Hiding' formula in the textbox. When the logical expression returns a TRUE value the button is hidden. When the returned value is FALSE the button is displayed.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

User def

Free definition.

Could be used by a custom template.

User prop

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to add definitions to the ones generated by CodePainter.

Example

```
forecolor=RGB(255,0,0)
```

Change Font

Defines the elements font. If no font is defined the dialog widow's default font is used.

Global font

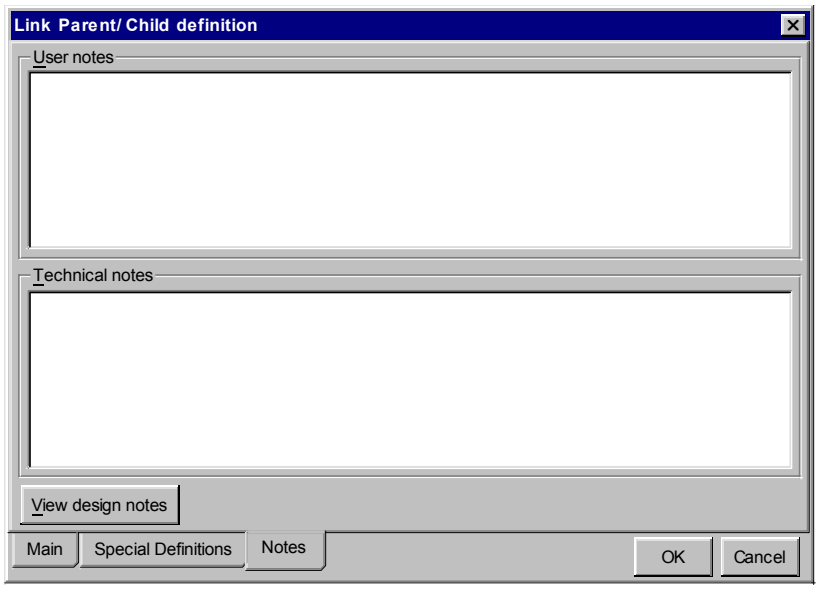
When selected the font defined in the 'Global Font' menu is applied to the current element.

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.

Picture 208 - Link Parent/Child Definition: Page 3



User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

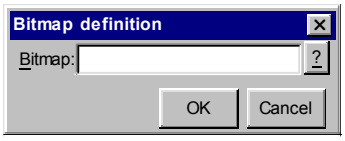
Technical notes on the element.

These notes are used to create technical documentation.

View design notes

Quick access to the Design notes of the current element.

5.7.6 Bitmap Definition



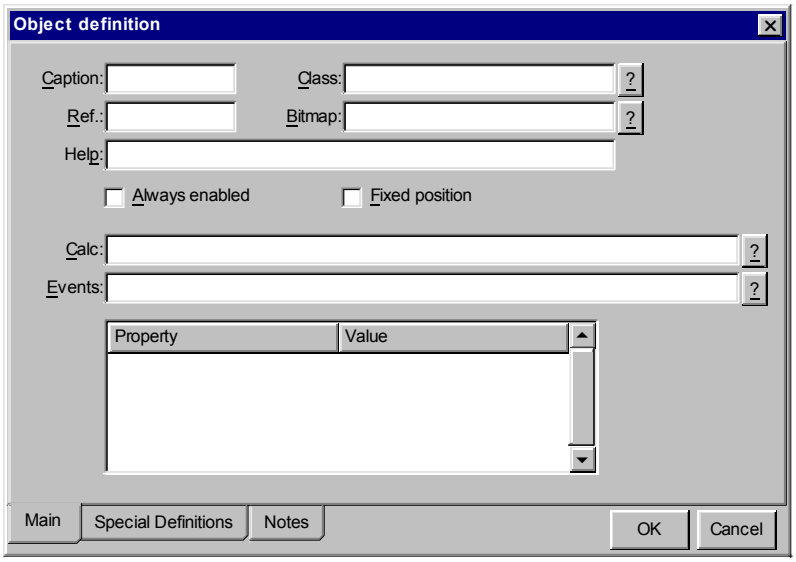
Picture 209 - Bitmap Definition: Page 1

Bitmap

5.7.7 Object definition

Main Page

Picture 210 - Object Definition: Page 1



Caption

Object text.

This text is used as title for those objects that need a string to be displayed.

Class

Object class.

Objects that can be added to dialog windows must belong to predefined classes. CodePainter uses a run-time class library or dedicated templates to generate the relevant source code. The class identifies which tool must be used.

Clicking the '?' button you can access all classes installed in your development environment. For more information on classes and their use, please refer to the COMPONENT GUIDE.

Ref.

Name of the variable associated to the object.

Objects are created as 'controls' in the current form. Associated working variables are not created, because objects are not read by CodePainter's standard procedures.

When you require to access the object from a manual area or a routine, you first need to define the variable name in this property. The working variable created works like the ones associated to fields or variables and is initializated with the pointer to the object.

Bitmap

Bitmap associated to the object.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Always enabled

The object is enables in 'Editing' and 'Query' mode.

Objects are used to access procedures that are bound to the main dialog window. Using these procedures may make sense in the 'Editing' and/or in the 'Query' mode only. E.g. an object executing a graph should always be enabled.

Setting this flag the object will be enabled in the 'Editing' as well as in the 'Query' mode. When the flag is not set the object is enabled in the 'Editing' mode only.

Fixed position

The repeated object is in a fixed position.

In complex programs the grid associated to the body may be too small to contain all required values. You can fix the position of body's elements. The element is displayed only once in the dialog window. The element takes on different values depending on the active row, but takes up only reduced space.

During data input you will first input all fields in the grid, pass on to fixed elements and then go on to the next row.

Calc

Passes on values to the object every time the dialog window is re-calculated.

The input dialog window is recalculated when specific actions are performed: when a new record is input, or when the user inputs specific fields, etc. Each time the dialog window is re-calculated it notifies the object passing on as parameter the value defined in this expression. The object will react according to the class to which it belongs.

For example the 'Dash Board' ('cruscotto') changes the position of the arrow, the 'Calendar' ('calendario') selects the current date, etc.

For more information on how each class exploits recalculations please refer to the COMPONENT GUIDE.

When calculations are complex you can limit re-calculations defining the 'Depends on' property as for any field or variable.

Events

List of events to which the object is hooked.

Objects can communicate with the dialog window in which they are in either through recalculations or events.

Events are notified by the dialog window. The object will react according to the class's specifications to which it belongs. Clicking the '?' button next to properties you access the list of available events. For more information on how each element answers to a given event, please refer to the COMPONENT GUIDE.

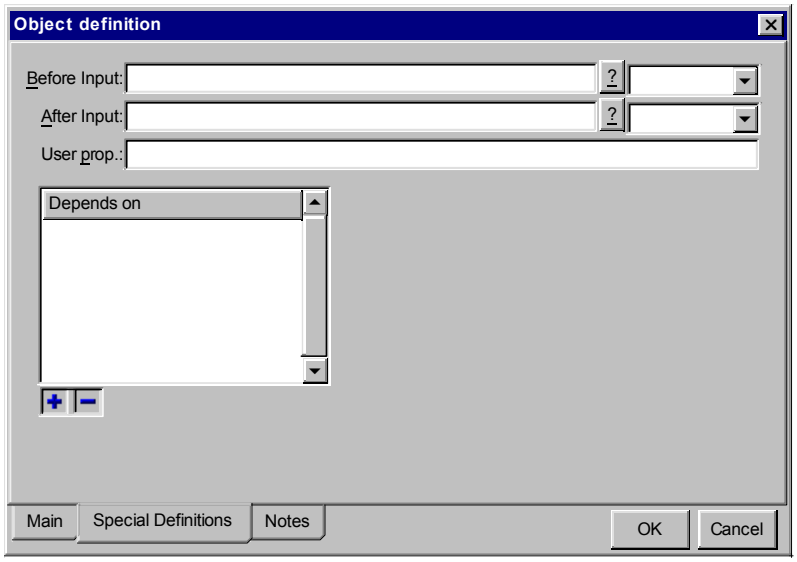
Properties

List of object properties.

Objects have a set of properties that differ according to the class to which they belong.

This list displays the property name in the first column and the default value in the second. These values can be changed in order to configure the object. For more information on object properties, please refer to the COMPONENT GUIDE.

Special Definitions Page



Picture 211 - Object Definition: Page 2

Before input

Program that must be executed before the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low level and cannot be defined so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

After input

Program that must be executed after the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low level and cannot be defines so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

User prop

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to adddefinitions to the ones generated by CodePainter.

Depends on

List defining the elements on which the calculation execution for associated objects depends.

Calculations and links are normally executed every time a value is entered. This strategy works smoothly if the calling routine is quick. If calling routines take long to be terminated you need to limit the number of calls.

When defining working variables in this list, calculations and associated links are executed only when the value for one of these variables changes.

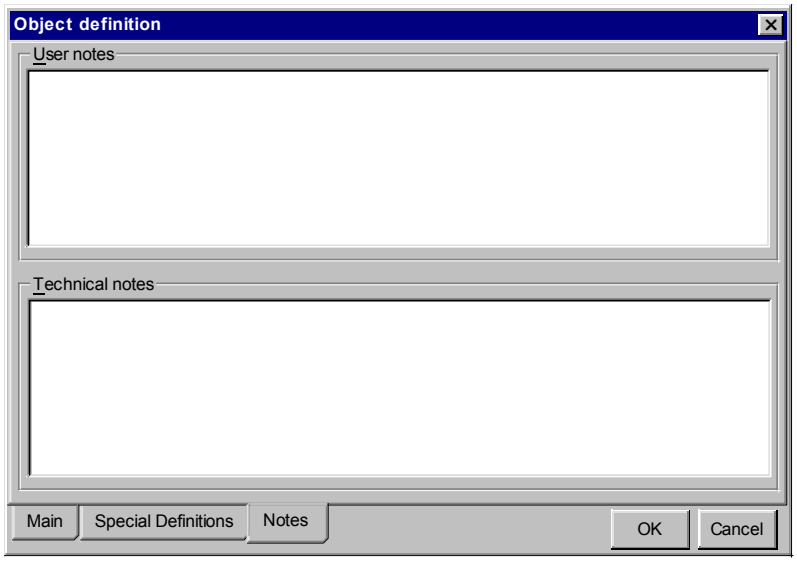
In this list you should define all calculation parameters associated to the element.

Using this method you avoid executing many code lines. Nevertheless checking the change of the value is costly and therefore this option should be activated only when calculations are complex.

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.



Picture 212 - Object Definition: Page 3

User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

Technical notes on the element.

These notes are used to create technical documentation.

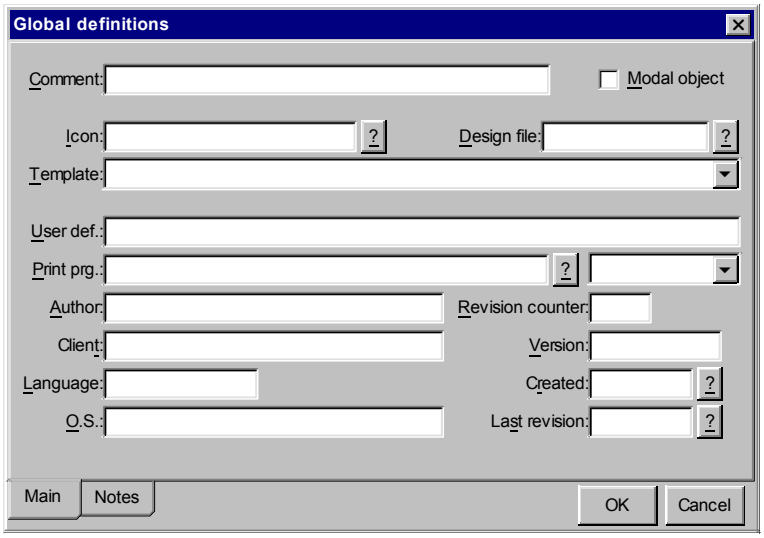
5.8 'Globals' Menu Advanced Options

This section details the 'Globals' menu functions.

5.8.1 Global Definitions

Main Page

Picture 213 - Global Definitions: Page 1



Comment

Brief comment on the entity. This comment is displayed in the project file so that you can search entities by file name (i.e. the associated procedure), or by brief description.

Modal object

Defines whether the entity is a modal dialog window or not. Modal dialog windows ask to terminate the input before you can go on to another dialog window. This means that the user cannot use the mouse to go to other windows until the current window is saved (F10) or exit (Esc).

Icon

Icon associated to this entity.

Template

Name of the template used for code generation. In this property you need to define the name of the template used to generate the source code. Templates are procedures' skeletons, finalized by the programmer to generate the required source code. Change the template and the generated source code changes as well.

User def

Free string. Can be used by custom templates.

Print prg.

Program activated during queries pressing F2.

In this property you can define the program that must be called when the user clicks the 'Print' button during queries. The combobox next to the property defines how the command line must be interpreted.

'User program': the command is copied in the source code. 'Mask': a dialog window created with the Dialog Window Painter is opened. 'Batch': a routine created with the Routine Painter is executed. You can also pass on parameters defining them ion brackets. 'UTK Object': defines an output configuration created with user tools such as the Query Painter.

The '?' button next to the property displays a list of possible choices depending on the defined activity.

Author

Author of the program. Short text defining the developer in charge of the entity.

Client

The commissioner. Short text to define the customer who commissioned the entity.

Language

Development language. Short text describing the programming language used to develop this entity.

O.S.

Operating System. Short text defining the limits given to the entity by the operating system.

Revision counter

Revision counter. This read only property shows you how often the entity has been changed. The number is increased automatically every time the entity is saved. This property can be used to check whether copied objects have been changed by the original author.

Version

Entity version number.

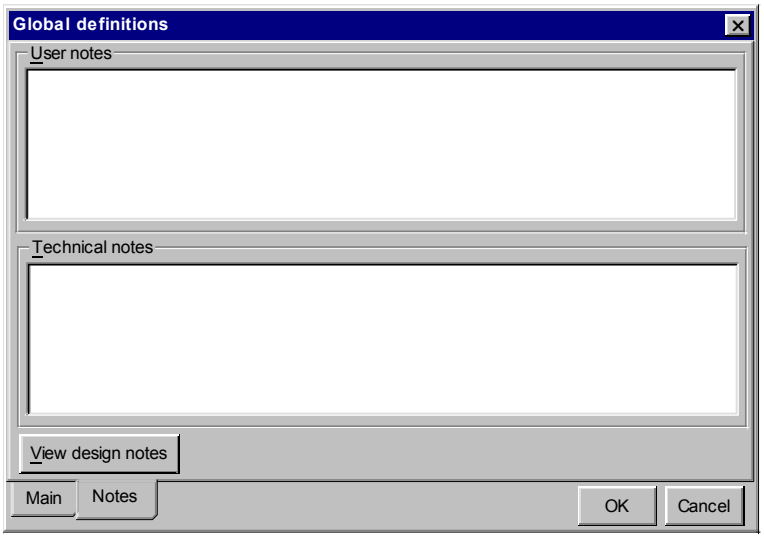
Created

Date in which the entity was created.

Last revision

Date in which the last revision was made.

Notes Page



Picture 214 - Global Definitions: Page 2

User Note

Notes on the element. These notes are used to create user documentation.

Technical Notes

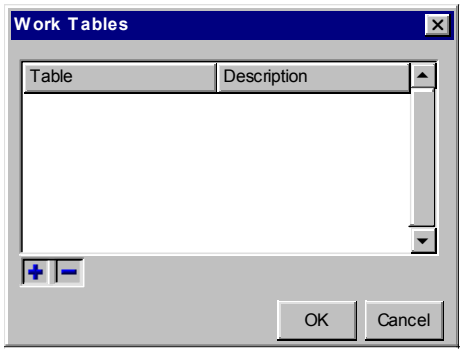
Technical notes on the element. These notes are used to create technical documentation.

View design notes

Quick access to the Design notes of the current element.

5.8.2 Tables

Picture 215 - Tables



Tables

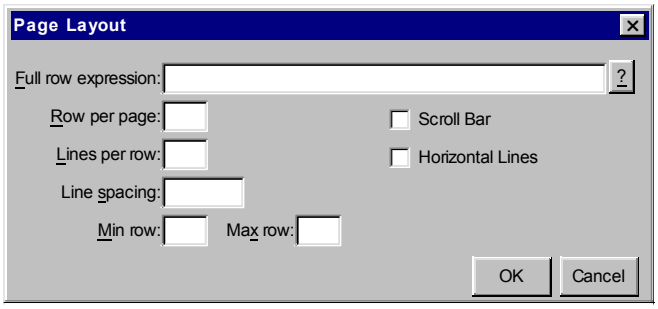
List of tables.

The generated procedure controls that the database contains all tables displayed in this list. When one or more tables are missing an error message is displayed and the procedure is not executed. This is done to avoid errors.

The procedure can also use tables which are not included in the list. Programs should not be executed if they can generate errors.

Tables used in links are automatically added to the list.

5.8.3 Page Layout



Picture 216 - Page Layout: Page 1

Full Row Expression

Full row expression.

Rows are considered completed only when the expression is TRUE. When the user tries to go from one row to another this expression is checked. When the row is considered full and the cursor is on the last row of the body a new row is added, otherwise the cursor goes to the first field of the footer.

Row per page

Number of rows in the body.

In this property you need to define the number of rows in the entity's body. By increasing this number the grid is enlarged so that it can display more rows.

Line per row

Number of lines per row.

The high of each row is a slightly higher than input fields basing on default fonts. Changing this property you can increase rows' size and have more space for data in the body.

Increasing rows' space the overall space used by the body increases as well. The dialog window margins may be overridden. To stay within margins you need to decrease the number of rows per page.

Line spacing

Space between single lines in the row.

Each body row is made of one or more lines. In this property you can define the lines' heigh.

Min. row

Minimum number of rows required for the entity.

When the data input is saved the number of rows in the body is checked. When the number is lower than the one defined an error message is displayed and the body is edited again.

Max rows

Maximum number of rows allowed for the entity.

When the data input is saved the number of rows in the body is checked. When the number is higher than the one defined an error message is displayed and the body is edited again.

Scroll Bar

When selected the vertical scroll bar is enabled.

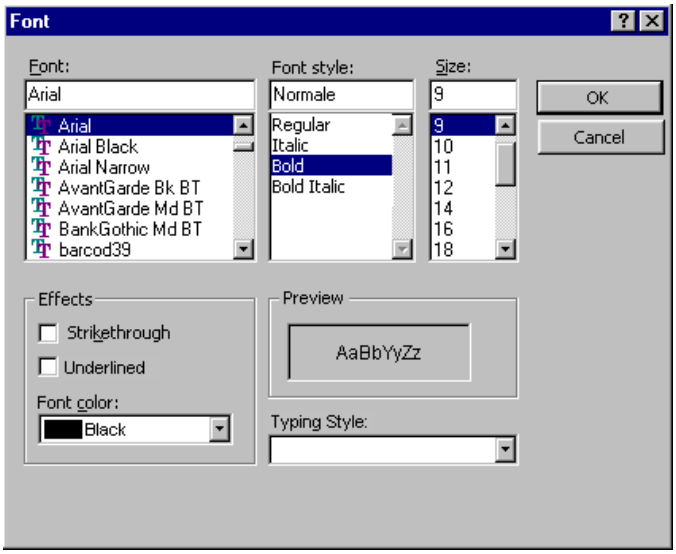
Horizontal Lines

When selected the horizontal lines dividing page's elements are enabled.

5.8.4 Font

Allows changing default fonts for the entity in use.

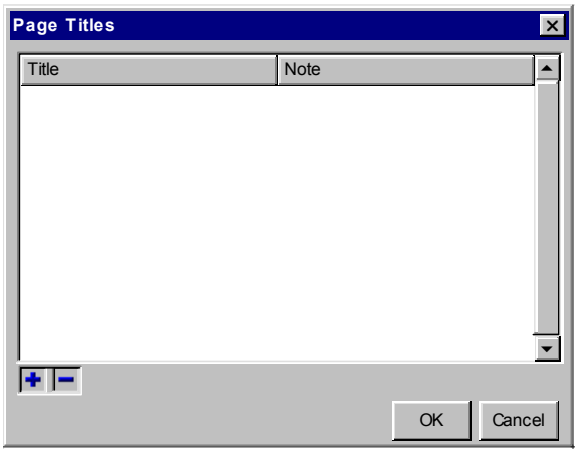
Default fonts in this area are the ones you defined in CodePainter's Front End in the 'Project' menu in the 'Project Options' option.



Picture 217 - Font Definition

5.8.5 Page Titles

Picture 218 - Page Titles: Page 1



Titles

Page titles. Entities can have dialog windows having more pages. The default title is 'Pag.' followed by the page number. This list allows changing pages' title.

5.8.6 Autonumber

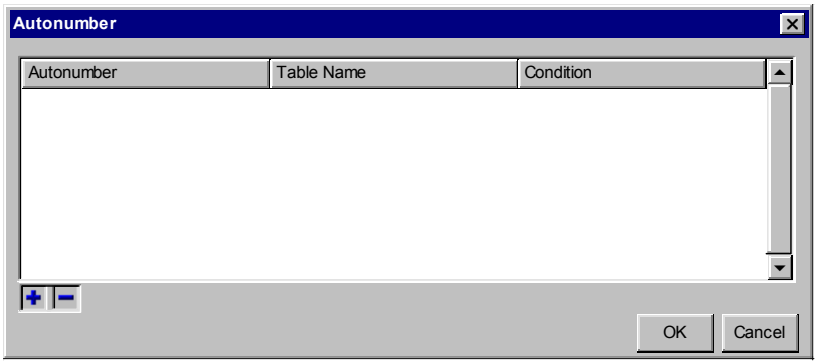
Manages progressive numbering automatically.

You can define fields that are numbered automatically in each entity. Once the starting number is defined the field is increased every time a new record is input.

When the 'autonumber' option for a field is defined, the field is initialized with the next value in the numbering sequence when a new record is input. If the user does not change the value and the record is saved the table containing progressive numbering is read. The system checks whether any other user has taken the 'booked' number. If the 'booked' number is still available the record is saved using it. If not the next available number is identified, the record saved with this new number and a message displayed so that the user is notified of the change. This method makes sure that in multiuser environments all available numbers will be used and that no numbering gaps are created.

When the user edits the 'booked' value the field value is overridden. The 'custom' number could be greater or less than the one given by the system. When the number is greater than the one given by the system the record is saved with that number and a numbering gap is created. When the number is less than the one given by the system the record is saved without changing the 'booked' field value nor the autonumbering table.

This second method allows to postpone data entry for known quantities. For example Invoices are input by the Head office. Unit B has issued 10 invoices but has not yet send the data. Head office can leave a gap of ten units overriding the next invoice number. When the data from Unit B arrives it can be input using the 10 numbers left empty.



Picture 219 - Autonumber

Autonumber

List of progressive numbers.

This list contains the fields building the progressive number, the reference table, and if required the expression that defines whether the progressive system must be used or not.

Autonumber

Name of the field that will be linked to a progressive table. You can define more fields simply dividing them with commas. In this latter case the last field is the one taking on progressive numbering, whereas the other fields are the 'variations'.

Example

You need to number a specific document and you also need to differentiate in-coming from out-going documents. You need to define two fields:TIPDOC and NUMDOC. The system will create progressive numbers for each TIPDOC value and inputting the progressive number in NUMDOC. Fields receiving progressive numbers can be either numeric or alphabetical. In order to match the alphabetical order with the sequence character types '0' are added to in front of the letter until the field is filled in completely.

Table name

Name of the table containing the progressive value. Progressive values are stored in a dedicated table. This field contains a symbolic name required to identify which numbering must be used. You can create separate numberings, e.g. 'cli' for the customer key, 'art' for the item code) or numberings that can be used by more entities, e.g. different documents such as orders and invoices, using the same numbering.

Condition

Sometimes you may be required to use progressive numbering only under certain conditions. For example you may need to save in-coming and out-going in one entity only. You define the field NUMDOC that must be automatically numbered for out-going documents, but in which you want to manually input numbers for in-coming documents. If out-going documents have the value 'O' in the field TIPDOC you need to define the condition TIPDOC='O' so that progressive numbers are used only for these documents.

Autonumber

You want automatic progressive numbering on invoices.

The field TIPODOC contains the document type ('I' for invoices, 'R' for receipts). The field NUMDOC must contain the progressive number for invoices and the manually inputted numbers for other documents.

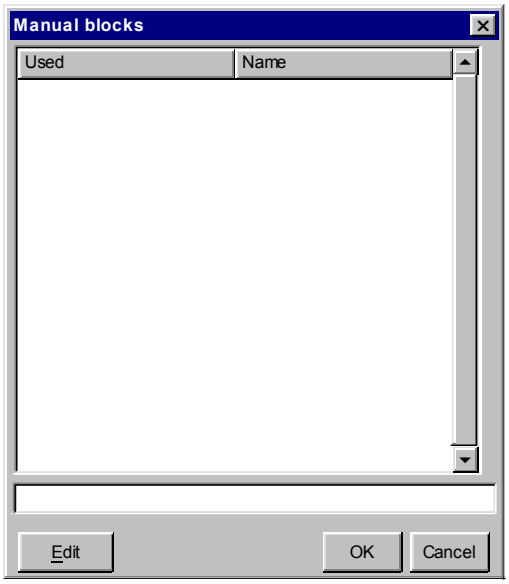
The list will contain:

'w_NUMDOC' in the 'Autonumber' column.

'doc' in the 'Table Name' column.

'TIPDOC'="F" in the 'Condition' column.

5.8.7 Manual Blocks



Picture 220 - Manual Blocks: Page 1

Used - Name (List of manual areas)

List of manual areas contained in the template.

Manual areas containing some code are highlighted in the left column.

Manual Areas

Name of the manual area that must be edited.

You can select manual area either from the list or defining different name in case the desired manual area is not included in the list.

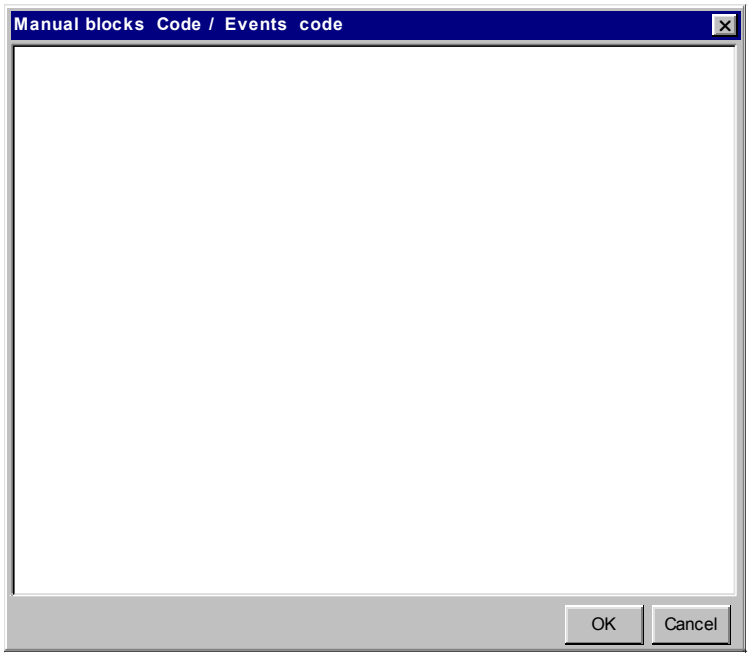
Edit

Access the code defined in the selected manual area.

Double clicking the manual area a dialog window is opened where you can define the code :

5.8.8 Manual Blocks Code

Editing dialog window for manual areas.



Picture 221 - Manual Blocks Code

Code

Source code contained in the manual area.

5.9 Selection Lists

CodePainter has a set of lists based on the project's data dictionary.

These list are generally activated in the Codify tool using the '?' or '...' buttons. These lists make information required for development available.

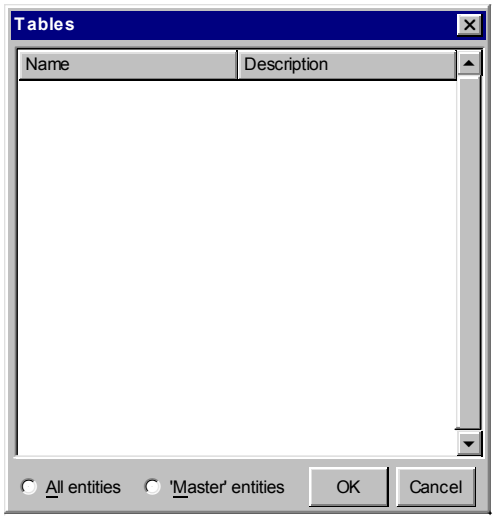
These lists can be sorted double clicking the column title. Selected values are input in the desired window section.

The sequence of selected elements can be moving them up or down in the list. To move elements you need to position the mouse on the first column on the left and wait until the mouse changes into a hand.

This section details the use of selection lists.

5.9.1 Tables

Picture 222 - Tables



Tables List

List of available tables.

Entity Type

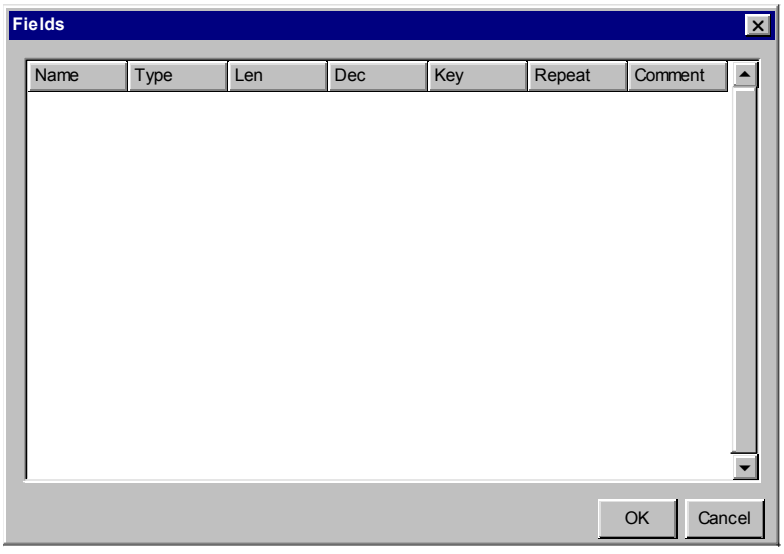
All entities

All tables are displayed.

'Master' Entities

Displays only tables belonging to predefined entity classes.

5.9.2 Fields



Picture 223 - Fields

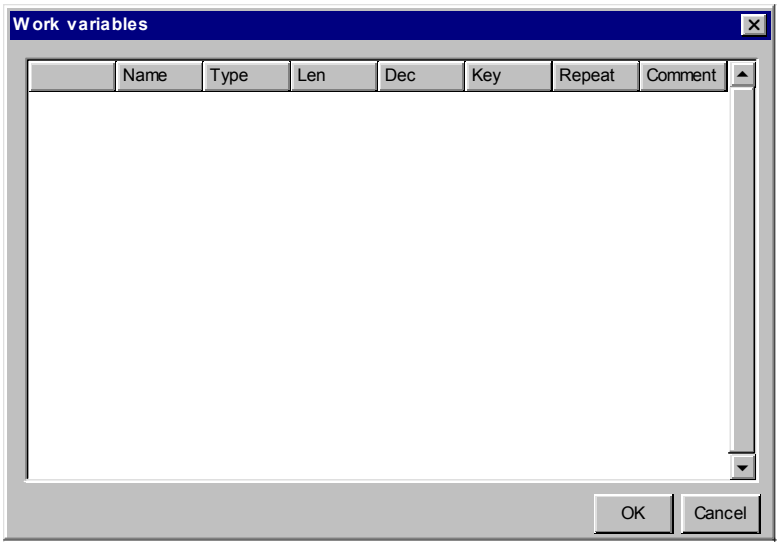
Fields List

List of fields. The list contains all fields defined.

5.9.3 Work Variables

List of variables that have not been implemented so far.

Picture 224 - Work Variables



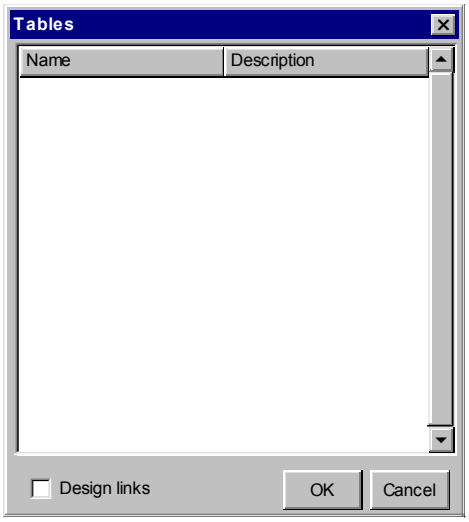
Variables

List of declared variables that have not been implemented so far.

Very often when you define links, variables are defined before they are displayed on the screen. This list analyzes all link definitions and details all available variables. The main advantage is that data is read directly from the data dictionary. Therefore it also reads data type and length, so that the new element can be added to the screen with the correct settings.

5.9.4 Tables

List of files that can be used to define links.



Picture 225 - Tables

Files

List of all tables in the design plan.

Design Links

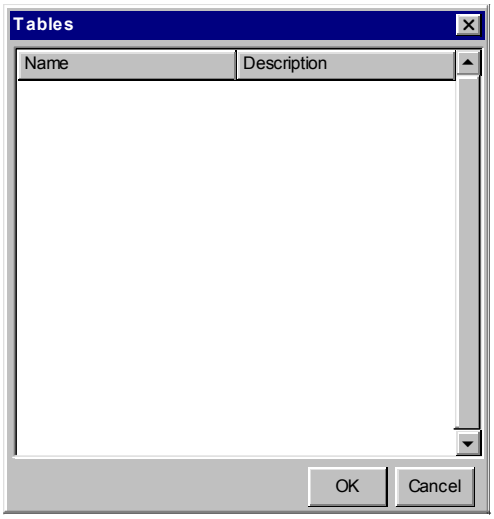
Defines whether to list all tables in the design plan or only those for which a link has been defined during the Design phase.

In the design plan each entity has a set of declared links. During the Codify phase you can create new links for which the source code is generated but not the database referential integrity. It is recommended to use declared links only.

When this flag is set the list above shows declared links. When the flag is not set all tables are displayed.

5.9.5 Tables

Picture 226 - Tables



Tables list

List of available tables.

Entity Type

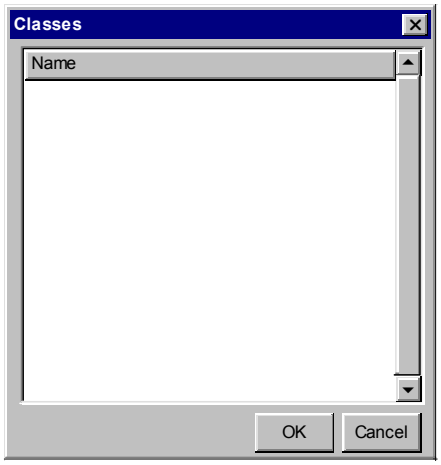
All entities

Displays all entities.

'Master' entities

Displays only entities belonging to the Master class.

5.9.6 Classes



Picture 227 - Classes

Classes List

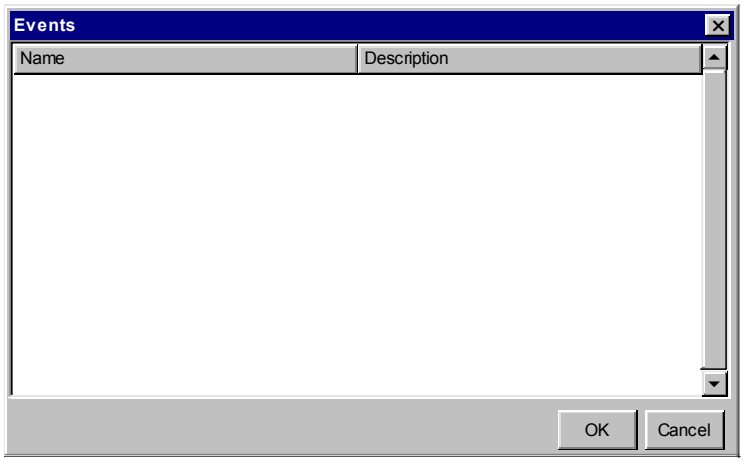
List of available classes.

This list contains the name of object classes defined in CodePainter.

All defined classes are stored in the file CLASSES.CPL under the Painter's Classes directory. To add new classes you simply need to define them in this file.

5.9.7 Events

Picture 228 - Events



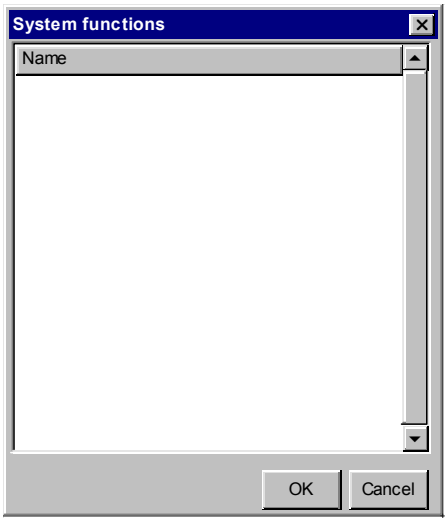
Events list

List of Events.

At run-time CodePainter notifies events to the objects in the dialog windows. Objects will therefore be able to react to given situations, such as data loading, change of an element or saving a record.

This list contains the name and brief description of all events that can be notified.

5.9.8 System Functions



Picture 229 -
System Functions

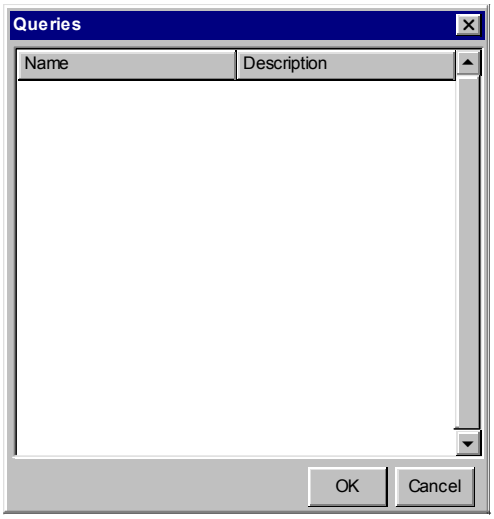
System Functions

List of system functions.

System functions are identified by strings. This list contains all system functions implemented by CodePainter's run-time system.

5.9.9 Queries

Picture 230 - Queries



Name - Description (Tables list)

List of available tables.

All entities - Master entities

All entities

Displays all tables.

'Master' entities

Displays only tables that belong to a to an entity of this class.

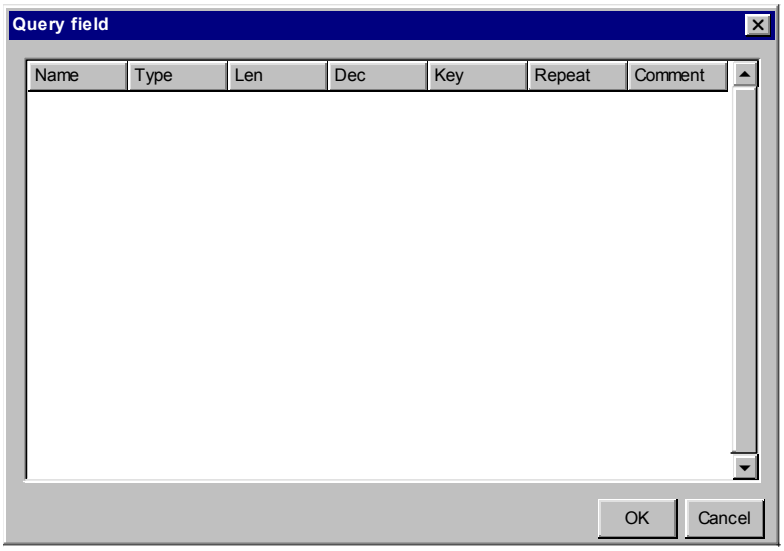
Design links

Defines whether to display all tables or only those associated to a table for which a link has been defined in the design plan.

When the flag is set the list displays only linked tables having referential integrity. When the flag is not set the list displays all tables.

You can create links in the Codify phase without going back to the design plan. The code to manage the link is created but the referential integrity to the database is omitted.

5.9.10 Query Field



Picture 231 - Query Field: Page 1

Fields list

Field list.

This list contains all fields of the selected query.

Fields list

Field list.

This list contains all files of the selected entity.

5.9.11 Design Notes

Picture 232 - Design Notes

Design Notes

Displays the notes added to the elements during the Design phase.

5.10 Product Information

The Help menu option 'About' displays information on the product.

5.10.1 About

The About dialog window contains information on the procedure in use.

The logo CODE**PAINTER REVOLUTION** is displayed and information on the version, series number and build number are given.

The build number helps you identifying CodePainter's distribution, no matter if the product is released totally or partially or through patches.

Picture 233 - About: Page 1

Chapter 6

Master/Detail

6.1 Master/Detail Painter

Master/Detail entities are very similar to Detail File entities. They look alike and their definition and use are very similar. The main difference is that the Master/Detail entity is made of two objects, a **Master** and a **Detail**.

Master/Detail entities manage two physical tables. The first (Master) contains Header and Footer data. The second (Detail) contains the Master Key and repeated rows of information. The two tables are linked through the primary key.

These two tables are displayed in one window only. Header and Footer data are stored in the main window and repeated data in a grid that can be scrolled up and down.

Choosing Detail File entities rather than Master/Detail entities is bound to the kind of problem you are required to manage.

Working with Detail Files there is the advantage that you use a single table only. On the other hand if you have many 'general data' fields in the header and footer the Detail table will contain redundancy data. Indeed general data would be repeated as many times as the number of rows in the document. Master/Detail entities store 'general data' only once, thus avoiding unnecessary redundancies.

Obviously if the header must contain a few fields only creating a Master Table would be an unnecessary waste of memory space. In this latter case you should use Detail File entities.

The Master/Detail Painter is used to improve the prototype's layout and functionalities, or to create new Master/Detail File entities that will be included in the design plan.

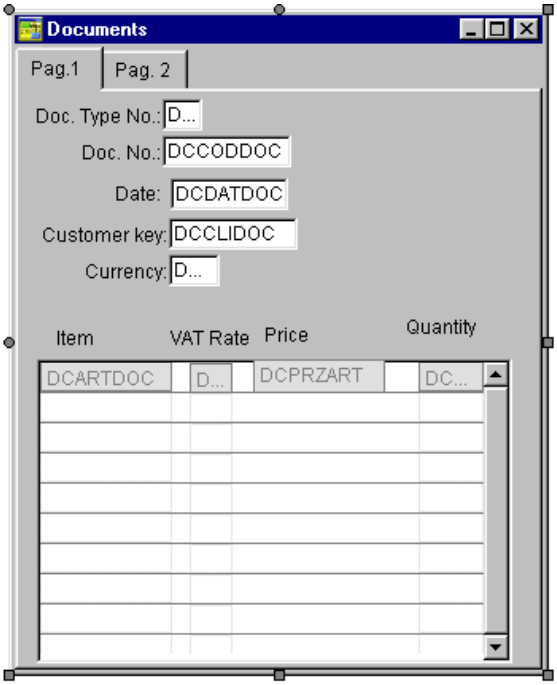
Using the Master/Detail File Painter you can improve the window's layout adjusting elements' size and position and you can also add strings, variables, buttons etc. using the WYSIWYG (What You See Is What You Get) methodology. You can further define initialization properties, calculations, validation checks, etc.

When you require to make changes to the Design plan you can maintain the changes made and generate the prototype only for changed elements in the entity. This is achieved using comparison functions that will be explained later on.

6.2 The Working Logic

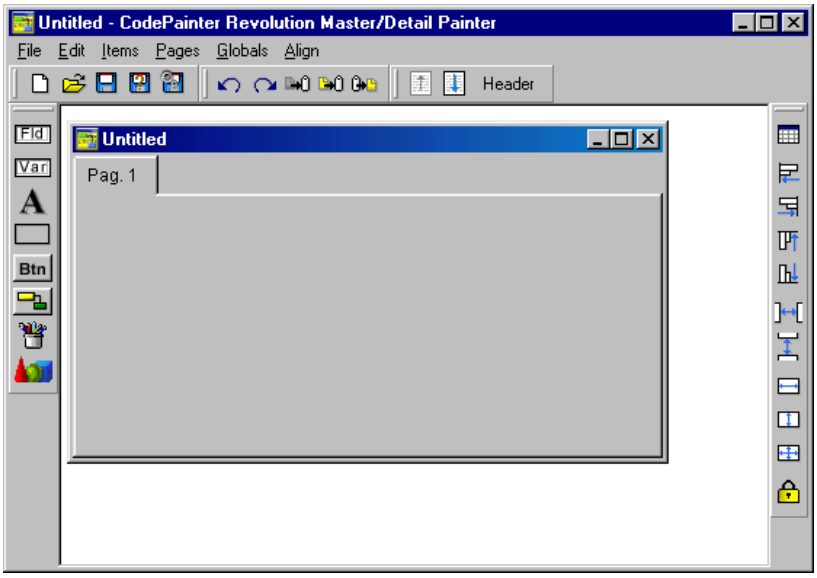
In the Master/Detail Painter you can open prototyped entities. A set of toolbars help you interacting with the prototyped entities. You can either open entities defined in the design plan or create new ones basing on the application's data dictionary. Opened Master/Detail entities have a variable number of rows.

To open defined entities open the 'File' menu and select the 'Open' option. All existing Master/Detail entities are listed. Fields defined for the Master/Detail entity in the design plan have been prototyped. These fields and descriptions defined in the linked table are included in the Detail File prototype and are sorted basing on the data dictionary.



Picture 234 - Master/Detail Painter: Prototype Window

To create new entities open the 'File' menu and select the 'New' option. You can also use blank window which is defaulted when you open the Detail File Painter.

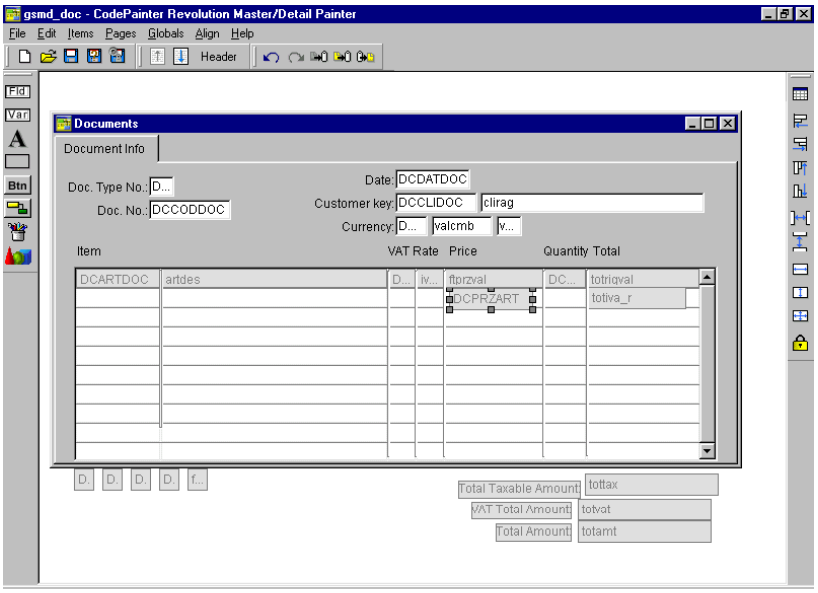


Picture 235 - Master/Detail Painter: New Window

Using the Master/Detail Painter you can '*design*' the interface and build in validations. Using the 'Code Generation' option you obtain a working procedure without the need to regenerate the entire design.

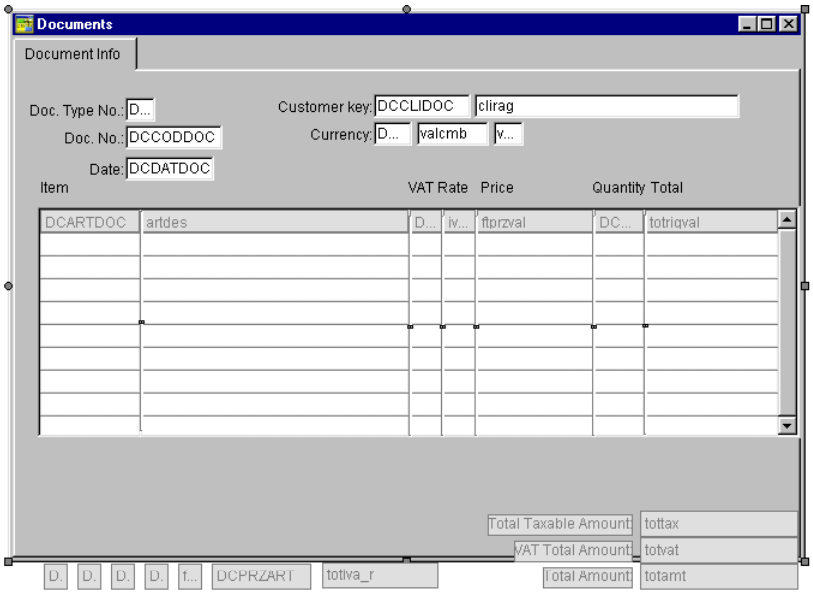
In the opened entity you can select, move and resize elements as you would do in any other MS Window application.

To move the entity window click the title (Caption Bar) and drag it in the desired position. You can notice that while you move the window the mouse changes into a cross.



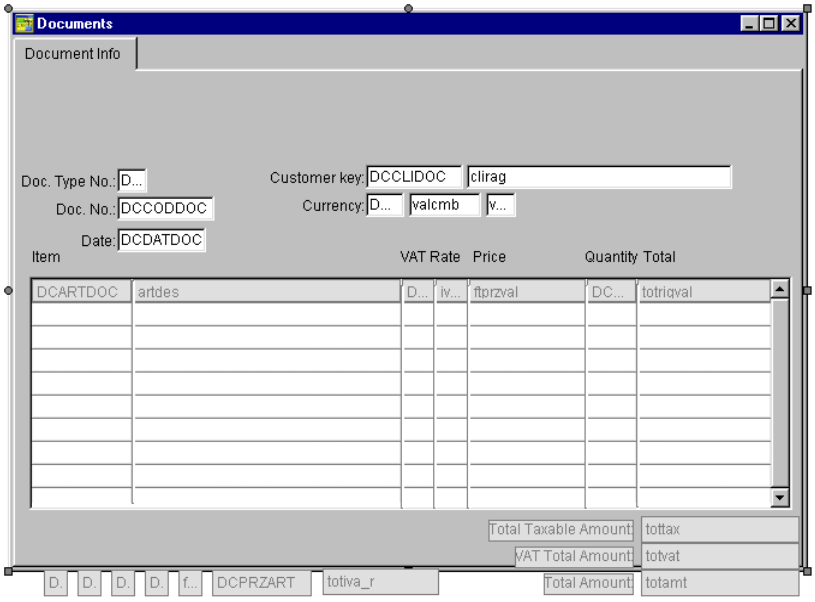
Picture 236 - Window Selection

To resize the window you need to select the entity and drag the displayed anchors. The size is always changed according to the direction in which you are dragging when the mouse becomes a 'two ways' arrow. The elements in the window remain fixed compared to the to left corner.



Picture 237 - Re-sizing By Dragging

When you right click and drag one of the circles the elements remain fixed compared to the bottom right corner. You can notice that while you right click and drag the circle an anchor is displayed under the 'two ways' arrow.

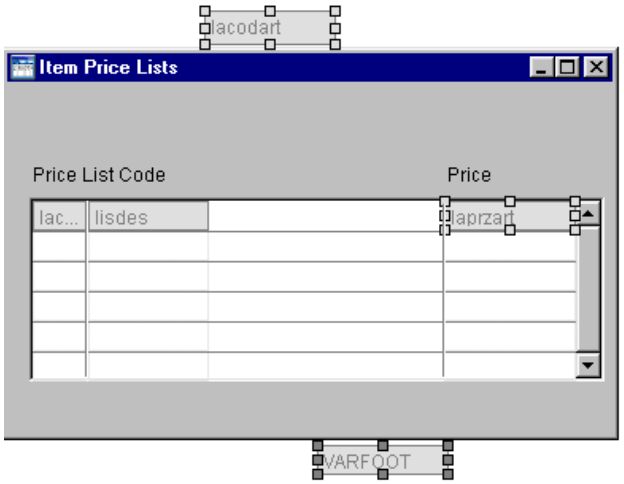


Picture 238 - Re-sizing with anchors, little circle and right mouse

Similarly you can move and re-size any element in the entity window.

6.3 Selecting And Aligning Elements

In Detail File and Master/Detail entities you can select elements of the header, footer and body simultaneously and use all 'Align' menu functionalities.



Picture 239 - Aligning Elements OF Header, Body And Footer

Double clicking elements you can edit the definition dialog window even if the selected element does not belong to the current block (header, body and footer).

Warning
Header, body and footer must not be intended as separate pages, but as a logical organization of elements.

Groups of elements can be selected in different ways:

Left clicking the mouse you can drow a rectangle around the elements you wish to select.

Picture 240 - Selected Group Of Elements

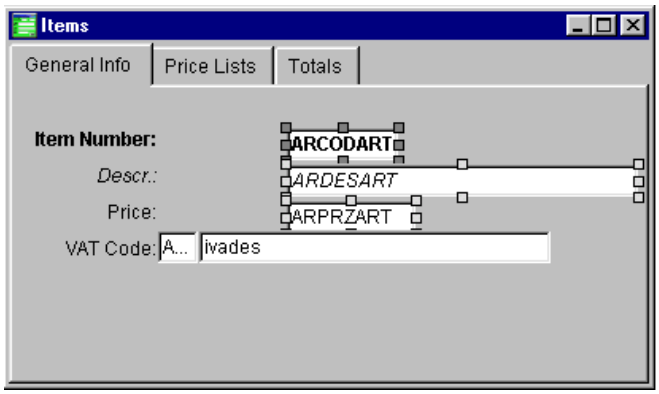
You can also keep the **<Shift>** key pressed and click the elements you wish to select. Single elements are deselected still keeping the **<Shift>** key pressed and clicking again the desired element. All elements are deselected clicking on the working form.

Selected elements are highlighted by handles, little *squares* around the element. Most elements with handles are light grey and one only dark grey. The latter one is the *Master Item* and is used as reference for resizing and aligning the other elements

Warning
When resizing and realigning elements the result changes depending on the Master Item selected.

Aligning the Master Item

Picture 241 - Align to the Left Of The Master Item



Picture 242 - Align to the Left Of The Master Item

The two pictures above show you the different results you will get selecting two different master items. The Master Item can be changed also once the group of elements has been selected simply clicking the desired item again.

You can align and re-size group of elements using both the 'Align' menu or toolbar.

To open the definition dialog window double click the desired element.

Right clicking elements a menu is opened that allows you to edit the element's properties, delete the element or change the editing sequence.

Pressing **<Del>** you can delete one or more selected elements.

Pressing **<Enter>** you can open the definition dialog window of the selected item or in case of multiple selection of the Master Item.

Pressing **<Cursor Keys>** you can move one or more selected elements.

Pressing **<Shift>** and **<Cursor Keys>** you can resize the selected element or in case of multiple selection the Master Item keeping the top right corner fixed.

Pressing **<Ctrl>** and **<Shift>** and **<Cursor Keys>**you can resize the selected element or in case of multiple selection the Master Item keeping the bottom left corner fixed.

Using keyboard keys you can also scroll elements.

Pressing **<Tab>** you can scroll forward.

Pressing **<Shift>** and **<Tab>** you can scroll backwards.

Pressing **<Ctrl>** and **<Tab>** you can scroll a selected group of elements forward.

Pressing **<Ctrl>** and **<Shift>** and **<Tab>** you can scroll a selected group of elements backwards.

To scroll the window elements standard Windows rules apply:

Pressing **<Tab>** you can scroll forward.

Pressing **<Shift>** and **<Tab>** you can scroll backwards.

When you are positioned on tabstrips you can press the **<Cursor Keys>** to scroll the option pages.

Pressing **<Alt>** and the <UnderlinedLetter> you can edit the corresponding option.

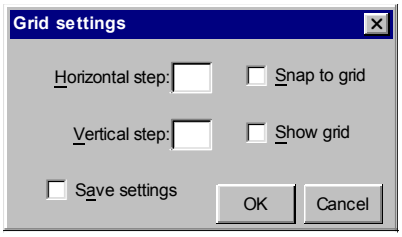
In selection lists you can sort columns. You can also add and delete elements using either the mouse or the **<Ins>** and **<Del>** keys.

When you move elements using the **<Cursor Keys>** changes are madein pixels. When you align elements you can define whether to use a positioning grid or not.

6.3.1 Grid Settings

Design grid definition.

Picture 243 - Grid Settings: Page 1



dx

Horizontal path. The grid has anchors. The distance between anchors is given by the number of pixel defined starting from the left border of the design window.

dy

Vertical path. The grid has anchors. The number of anchors is given by the number of pixel defined starting from the top border of the design window. When the dialog window has more pages the anchor is placed on the top border of the tab-strip.

Snap to grid

Defines whether the elements must be within the grid. When the flag is not selected the elements can be placed anywhere in the window. When the flag is set elements must be in fixed places.

Show grid

Defines whether the grid must be displayed or not. When elements in the dialog window are many and close to each other the grid makes the window unreadable. When this flag is not selected the grid is still active, but is not displayed.

Save settings

Defines whether grid settings must be saved or not. Grid settings can be defined and saved for each dialog window. When this flag is set settings are maintained.

6.4 Master/Detail Painter Toolbars

Let us now analyze the various Detail File Painter toolbars. These toolbars help you interacting with Detail Files.

6.4.1 Painter Tools Toolbar

The Painter Tools toolbar allows you to add new elements to the current entity. The toolbar has the same functionalities as the 'Items' menu.

figura 9 - Painter Tools Toolbar

Here to follow you will find a brief description of the files and their meanings.

Button	Meaning
	Adds database table 'Field' objects.
	Adds 'Memory Variable' objects to support validations and calculations/totals.
	Adds descriptive 'String' objects.
	Adds 'Box' objects that allow you to improve the entity's layout adding lines or rectangles that help subdividing the window.
	Adds 'Button' objects. These objects run queries, procedures, system functions and/or dialog windows.
	Adds 'Parent/Child Link' objects that allow you to create buttons that are integrated in the window or that open 'Child' entities.
	Adds Bitmap objects.
	Adds 'Object' objects that allow you to integrate external objects to your project. External objects belong to predefined classes. Examples are graphs, calendars, zooms, zooms with selection, etc.

6.4.2 File Toolbar

The 'File' toolbar has a set of button that help you interacting with the tool. This toolbar has the same functionalities as the 'File' menu.

Picture 244 - File Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Creates new entities.
	Opens existing entities.
	Saves the changes made to the current entity.
	Saves the current entity with a different name.
	Saves and generates the source code for the current entity.

6.4.3 Align Toolbar

The *Align* menu allows you to position and resize groups of selected elements.

figura 10 - Align Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Opens the grid management
	Aligns elements to the left in comparison to the Master Item.
	Aligns elements to the right in comparison to the Master Item.
	Aligns elements to the top in comparison to the Master Item.
	Aligns elements to the bottom in comparison to the Master Item.
	Positions elements with the same horizontal distance in comparison to Master Item and the selected elements.
	Positions elements with the same vertical distance in comparison to Master Item and the selected elements.
	Re-sizes selected elements with the same height as the Master Item.
	Re-sizes selected elements with the same width as the Master Item.
	Re-sizes all elements with the same height and width as the Master Item.
	Blocks the possibility to move and re-size elements (the functionalities are not inhibited in the 'Align' menu.

6.4.4 Clipboard Toolbar

The 'Clipboard' toolbar allows you to execute typical Window's commands such as Undo, Redo, Cut, Copy and Paste.

Picture 245 - Clipboard Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).
	The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).
	The 'Copy' option copies one or more selected elements.
	The 'Cut' option deletes selected elements.
	The 'Paste' option pastes copied or cutted elements in the selected position.

6.4.5 Pages Toolbar

Using the 'Pages' toolbar you can browse the entity's pages. The toolbar shows two buttons and the number of the page in which you are positioned.

Picture 246 - Pages Toolbar

Here to follow you will find a brief description of the buttons and their meanings

Button	Meaning
	Moves the edit functionality from the previous page to the current page. This button does not work if you are on the first page.
	Moves the edit functionality from the following page to the current page. This button does not work if you are on the last page.

6.5 Main Menu

6.5.1 File

Picture 247 - File Menu

The 'File' menu has a set of options to:

- Create new entities.
- Load existing entities.
- Save changes to the current entity.
- Save the current entity with a different name.
- Save and generate the current entity.
- Read information about CODE**PAINTER REVOLUTION**.
- Exit the tool.

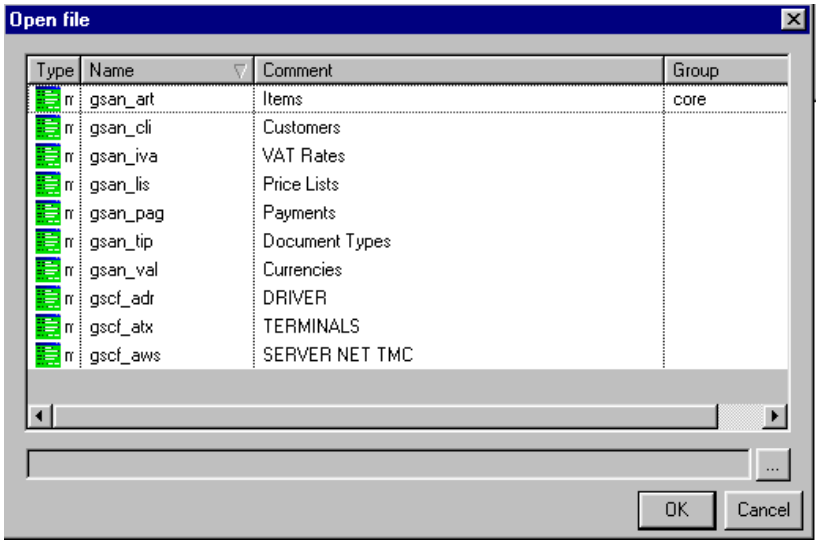
New

The 'New' option closes the current entity asking whether you want to save the changes or not and opens a new entity.

Open...

The 'Open' option loads definition files belonging to the current project.

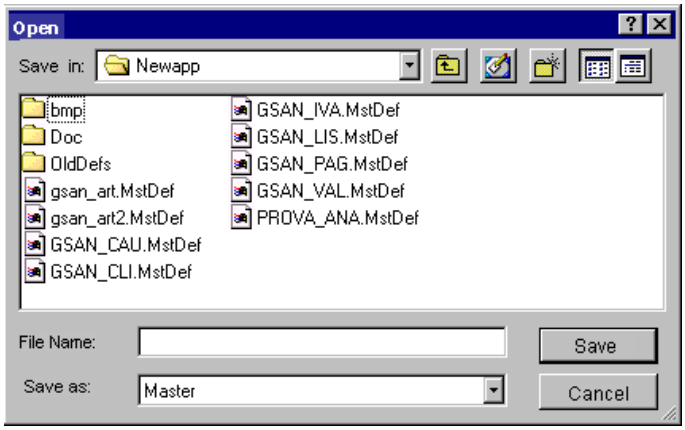
Picture 248 - Open File



The dialog window lists all entities in the project. You can order files by Type, Name, Comment or Group clicking the corresponding column. You can identify the sorting column by the arrow next to the column description.

Picture 249 - Sort Columns

Clicking the '...' button you can browse to search entities of the same type belonging to other project modules.



Picture 250 - Open Dialog Window To Select Design Plans

Save

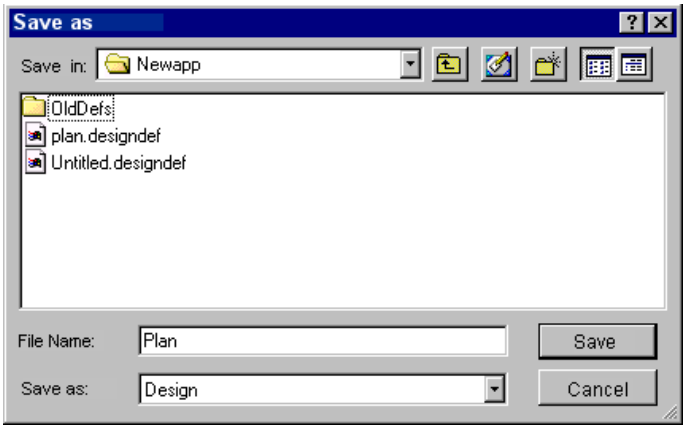
The 'Save' option saves the entity in use.

When you save the entity back-up file (.BAK) is created to store the entity without including the latest changes.

Save As...

The 'Save As' option saves the entity with a different name.

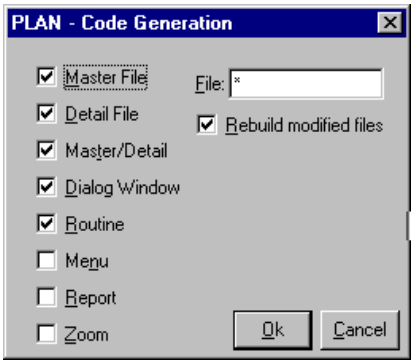
Picture 251 - Save As Dialog Window



Save and generate

The 'Save And Generate' option saves the current entity and runs the generation. The same is done opening the 'Generation' menu and selecting the 'Design'.

Picture 252 - Code Generation



About

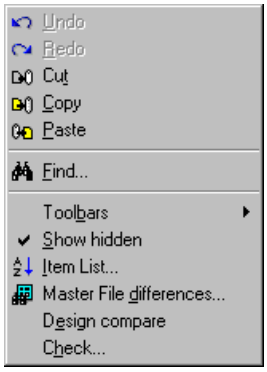
The 'About' option displays information about the product.

For more information please refer to section 'Product Information'.

Exit

The 'Exit' option makes you exit the tool asking you whether you want to save the changes or not.

6.5.2 Edit



Picture 253 - Edit Menu

Opens a menu from where you can:

- Undo/redo commands.
- Delete, copy and add elements.
- Search, display, replace elements.
- Show/ hide toolbars.
- Show/ hide elements with the 'Editing' flag set to 'Hide'.
- Display the 'Item List'.
- Identify the discrepancies between Codify and the Design.
- Identify errors in the source code.

Undo

The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).

Redo

The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).

Copy

The 'Copy' option copies one or more selected elements. More elements can be either selected or grouped in a selection frame you can draw with the mouse.

Cut

The 'Cut' option deletes selected elements from the plan.

Paste

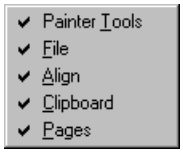
The 'Paste' option pastes copied or cutted elements in the selected position.

Find...

The 'Find' option opens the Find and Replace dialog window to find and/or replace elements.

Toolbar

The Toolbars option has displays a submenu to enable or disenable toolbars on the Design Painter.



Picture 254 - Menu Toolbar

Show hidden

The 'Show Hidden' option allows you to show or hide variables for which the 'Editing' option has been set to 'Hide'.

Item List...

The 'Item List' option displays the a list of all elements in the window.

File Painter Differences...

Using this option you can compare the entity with the design plan. For more information please refer to section 'Edit Menu Advanced Options'

Design Compare

Using this option you can compare the file definition with the entity defined in the Design phase. For more information please refer to section 'Advanced Edit Menu Options -Design Compare'.

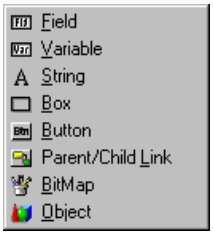
Check...

The 'Check' option checks the congruence of expressions and relations defined during the Codify phase. For more information please refer to section 'Edit Menu Advanced Options'.

6.5.3 Items

The 'Items' menu allows adding elements to the entity in use. It has the same functionality as the 'Painter Tools' toolbar.

Picture 255 - menu' Items



Here to follow you will find a brief description of the various elements:

Field

Using this option you can add 'Field' objects corresponding to a field of the associated table to your entity. When this option is selected the 'Fields of Tables ...' window is opened containing the list of the entity's fields. The list is read from the data dictionary.

Variable

Using this option you can add Variable objects to your entity. These objects are memory variables used to support validations and calculations/totals.

String

Using this option you can add String objects to your entity. These objects are descriptive strings.

Box

Using this option you can add Box objects to your entity. These objects allow you to improve the entity's layout adding lines or rectangles that help subdividing the window.

Lines are defined '*collapsing*' box objects using **<Shift>** + **<arrows>** keys, or the mouse.

Button

Using this option you can add Button objects to your entity. These objects are typically defined to launch queries, process procedures, system functions and/or Dialog windows.

N. B.
Using Button objects to integrate the entity with system functions ('Help', 'Query', 'Edit', 'Load', 'Save', 'Delete', 'Quit', 'PgUp', 'PgDn', 'ZoomPrev', 'ZoomNext') does inhibit standard toolbar functionalities in the generated application.

Parent/Child Link

Using this option you can add 'Parent/Child Link' objects to your entity. At Design level 'Parent/Child' relationships mean hierarchical links between two entities. Used as objects they allow you to create buttons that are integrated in the window or that open 'Child' entities.

Right clicking these objects you can access to the 'Open Codify...' option and the other standard options ('Edit', 'Delete', 'Sequence'). The 'Open Codify...' option allows you to open the codify tool of the integrated Child entity.

Bitmap

Using this option you can add Bitmaps to your entity.

Object

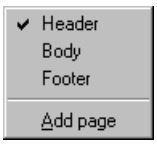
Using this option you can add 'Object' objects to your entity. These objects allow you to integrate external objects to your project. External objects belong to predefined classes. Examples are graphs, calendars, zooms selecting zooms, etc.

For more information on External Object Classes and/or on how to define new classes please refer to the COMPONENT GUIDE manual.

6.5.4 Pages

Using the 'Pages' menu you can browse the three entity's pages, namely Header, Body and Footer, as well as other additional pages. This menu has the same functionalities as the 'Pages' toolbar. New pages can be added only using the 'Pages' menu.

Picture 256 - Pages Menu



The menu displays the names of the available pages. The current page has the 'check' symbol next to the name.

Add Page

Adds a page to the entity. To delete a page you need to delete all contained elements. When you open the entity the next time, CodePainter identifies that the page is empty and deletes it automatically.

6.5.5 Global

The 'Global' menu allows you to add general information on the current entity, such as pages title, default fonts, templates, information on the author, etc. For more information please refer to 'Globals menu'.

Picture 257 - Global Menu

Globals...

Opens the 'Global Definition' window in which you can define the generation template, the program that must be launched when the dialog window is confirmed, general information on the author, etc.

Tables...

Allows to define which working tables must be opened.

Page Structure...

Allows defining pages characteristics, such as number of rows, activation of the scroll bar, line spacing, etc.

Font...

Allows changing the default font, size, style, etc. for the current entity.

Pages Title...

Allows adding titles to defined pages.

Autonumber...

Opens the window that manages autonumbering on fields of the entity.

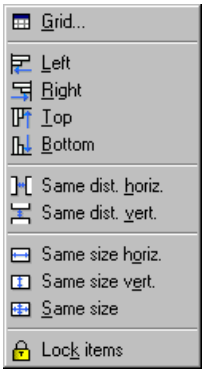
Manual Blocks...

Allows adding manual areas. In the manual areas you can add lines of code to integrate SW application functionalities.

6.5.6 Align

The *Align* menu allows you to position and resize groups of selected elements.

Alignment and re-sizing functionalities are made in relation to a master item. For more information please refer to section 'Selecting And Aligning Elements'.



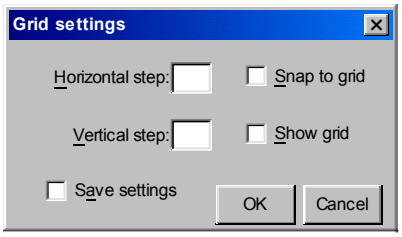
Picture 258 - Align Menu

The Align menu allows you to:

- Open the grid management
- Align to the right/left/top/bottom
- Position with the same horizontal/vertical distance.
- Re-size with the same height and width.
- Block changes for manual positioning and resizing.

Grid...

Opens the Grid Setting window to manage the grid:



Picture 259 - Grid Setting

Horizontal Step

Defines the horizontal size of the cell in pixels.

Vertical Step

Defines the vertical size of the cell in pixels.

Snap to grid

Activates the grid to position elements.

Show grid

Displays the grid.

Save setting

Saves the current setting of the grid.

Left

Aligns the left side of selected elements with the left side of the master item.

Right

Aligns the right side of selected elements with the right side of the master item.

Top

Aligns the top of selected elements with the top of the master item.

Bottom

Aligns the bottom of selected elements with the bottom of the master item.

Same dist. horiz.

Selected elements are positioned with the same horizontal distance. The distance is measured between the first and second element starting from the left.

Same dist. vert.

Selected elements are positioned with the same vertical distance. The distance is measured between the first and second element starting from the top.

Same size horiz.

Resizes the width of selected elements like the master item.

Same size vert.

Resizes the height of selected elements like the master item.

Same size

Resizes selected elements like the master item.

Lock items

Inhibits selection and change commands for selected items.

Warning

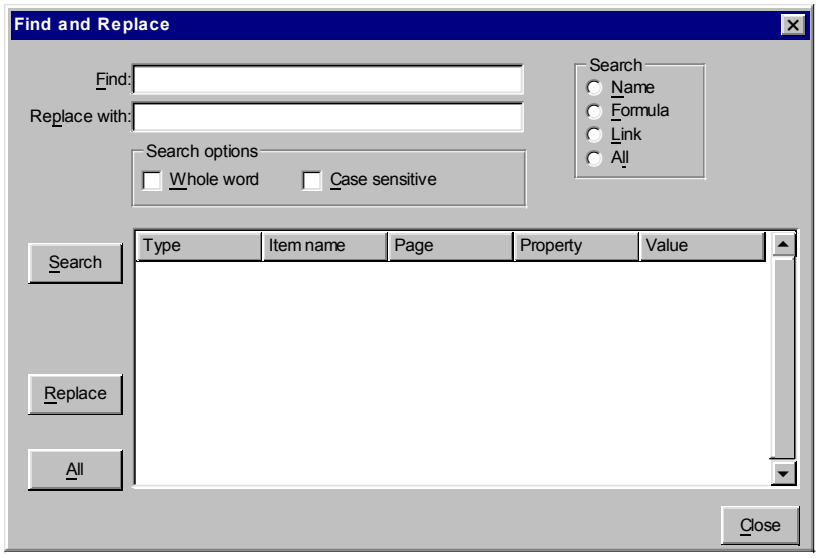
The commands are inhibited for the mouse and the keyboard while the Align toolbar remains active.

6.6 Edit menu Advanced Options

This section details the Edit Menu options.

6.6.1 Find And Replace

Picture 260 - Find and Replace



Find

String that must be found.

Replace With

String that must replace the found string.

Whole Word

When this field is flagged the search activity must be performed on whole words only. Otherwise search results could be also substrings of long words.

Case Sensitive

The search activity matches upper and lower Case.

Search Button

Starts the search activity:

Name:

the search activity is perfomed on element names

Formula:

the search activity is perfomed on defined formulas.

Link:

the search activity is perfomed on defined links.

All:

the search activity is perfomed on names, formulas and links.

List Of Found Items

Search result listing elements found.

Replace Button

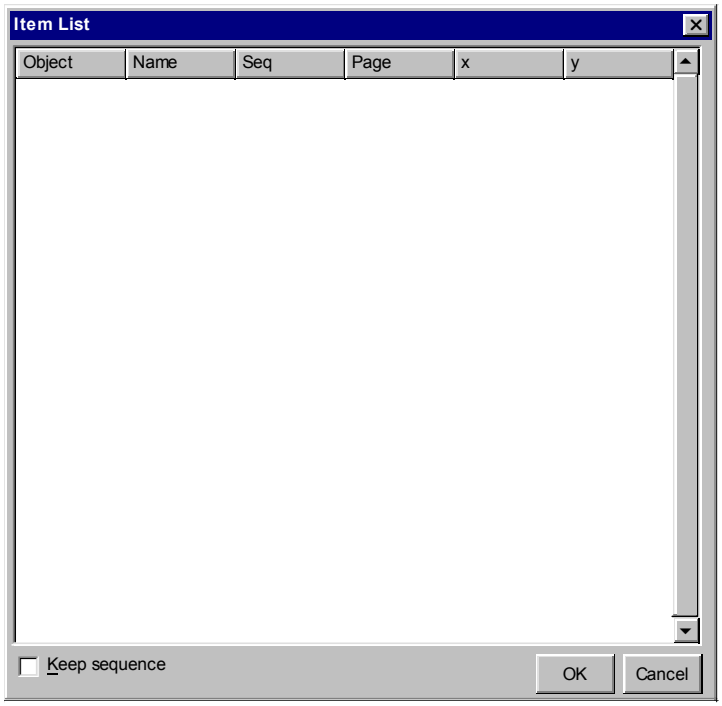
Replaces strings found with the string defined in the textbox 'Replace with' for all elements selected in the list (List of found items).

All Button

Replaces strings found with the string defined in the textbox 'Replace with' for all elements in the list (List of found items) no matter if elements are selected or not.

6.6.2 Item List

Picture 261 - Item List



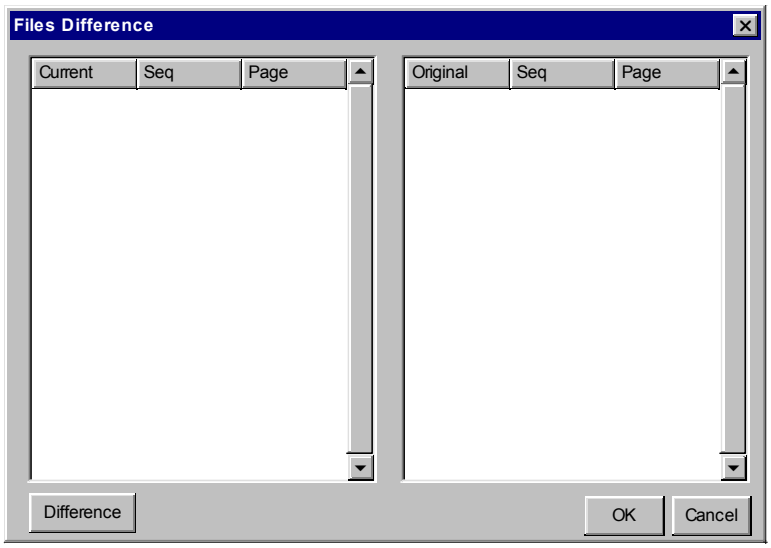
Elements List

List of elements in the dialog window. This list allows to access elements defined for this entity quickly.

Keep sequence

Flag to maintain the elements' sequence. When the flag is set and the dialog window saved elements are ordered to reflect the sequence defined in the elements' list. To order single elements right click the desired element and select the 'Sequence' option.

6.6.3 Files Difference



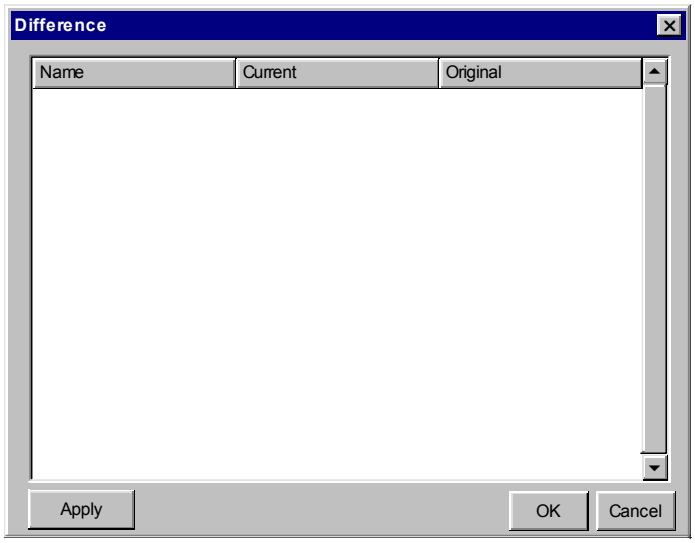
Picture 262 - Files Difference

Difference Button

You can compare two entities using the 'Detail File Differences' option.

6.6.4 Difference

Picture 263 - Difference



Difference List

List of differences between two definition files.

Apply Button

The selected difference is applied to the current definition file.

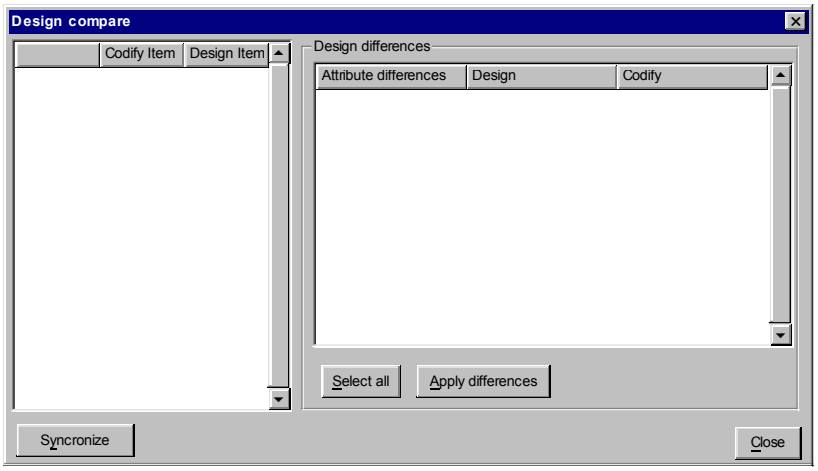
6.6.5 Design Compare

Comparing the Codify phase with the design plan allows you to synchronize entity definitions. It can be done only when the 'Bind to design entity' option in the 'Tables' or 'File and Keys' menu is set.

Once opened the dialog window displays all differences between the properties of design and codify entities, e.g. discrepancies between comments, or fields in a link, or missing fields, i.e. those fields defined in the design plan and not taken over to the Codify phase.

The comparison starts from the design entity. This involves that 'non-design' properties, which have been added during the Codify phase are not listed.

Using the 'Synchronize' or 'Apply differences' button you can synchronize some or all differences identified in the Codify phase.



Picture 264 - Design Compare

Elements List

List of elements for which the comparison has been run. The icon in the first column shows whether differences have been detected or not.

Design differences

List of differences for the element selected from the elements list. The first column briefly describes the attribute for which the difference has been identified. The second column displays the attribute value defined in the design plan. The third column displays the value defined in the Codify phase.

Select all

Selects all elements in the differences list.

Apply differences

Synchronizes codify attributes for selected elements basing on the design value.

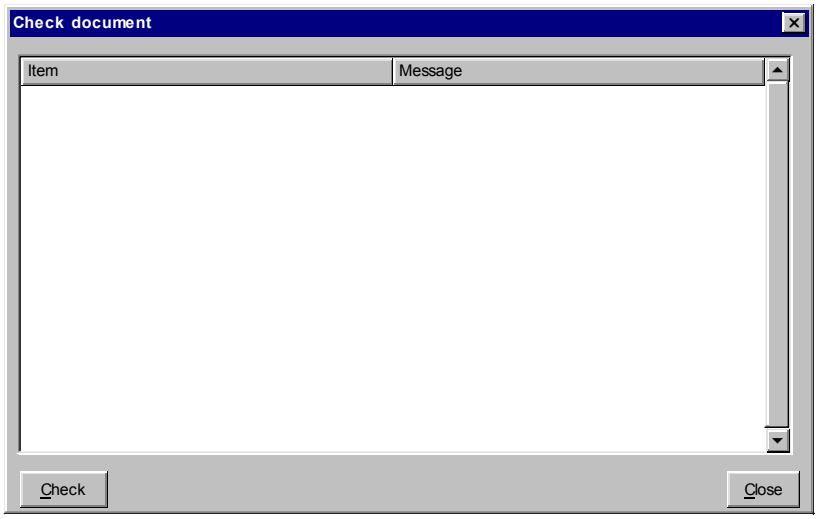
Synchronize

Synchronizes codify attributes for all elements basing on the design value.

6.6.6 Check Document

The Check dialog window executes congruence validations on definitions added using the Codify tool. CodePainter allows you to override values that usually are considered wrong, but may be required to manage specific issues. The Check dialog windows checks that no contrasting situations have been defined that could lead to malfunctions at run-time.

Picture 265 - Check Document



Errors List

Error list.

Check

Starts the validation of added definitions.

6.7 Items Menu Advanced Options

The Items menu is accessed opening the Edit - Toolbar menu or right clicking the working area and is used to add new elements to the entity in use. Let's see the various options in detail.

N. B.

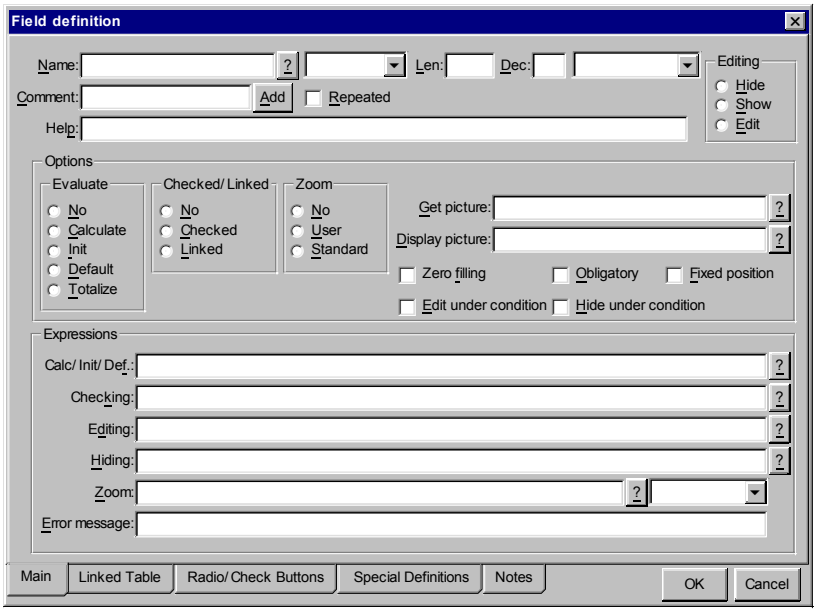
Options for variables and fields are the same. For each field a working variable is created. The working variable is saved in the field only when the Master is closed.Variables and fields are treated in the same way as they are both variables.

6.7.1 Field definition

The definition window allows to define properties for fields or variables. Here you can define how elements are displayed, or define control and calculation formulas, or mandatory fields, or whether to associate a checkbox, radio or combobox, etc.

Main Page

The first page defines the main element's properties. The window has three sections: the first groups the element's characteristics, the second activates controls or calculations and in the third expressions to execute controls are defined.



Picture 266 - Field Definition: Page 1

Name

Name of the field or variable.

This name must follow the rules for the construction of variable identifiers for the target environment. Typically it must start with a letter and cannot have blanks, commas or points. Some environments also have length limitations.

CodePainter creates for each element (fields and variables) a working variable that takes on the prefix 'w_'. Fields are initializated when the record is read. The user edits the record using the values in the working variables. When the record is saved the contents of 'w_' variables is saved in the corresponding database fields.

Variables that do not have values stored in the database are initializated when the record is loaded. The working variable either does not contain any value (i.e. blank for strings, or 0 for numbers), or it contains the value defined in the 'Init' expression. During the editing phase they behave as fields as defined above.

When the selected element is a field you can list all fields of the associated table clicking the '?' button. In the same way when the element is a variable you can list all variables that CodePainter identifies as useable, but have not yet been placed on the screen.

Field Type

Kind of element.

This combobox defines the kind of element you are editing. Implemented element types are those typically used by business/ commercial applications, i.e. character, numeric, date, logical, memo. The element type here must match with the associated database field. You can define different types only if the type is compatible with what is read by the files: shorter strings, numbers having less digits, etc.

Len

Element's length.

Here you define the length of the variable associated to the element. The length is expressed in characters for strings and in digits for numbers. For logical elements, dates and memos the default length is used.

Dec

Decimal digits for the element.

For numeric elements you can define the number of decimals required. This number is included in the total length.

Key/Index

Defines whether an elements is part of a key or of a search index.

Elements belonging to the primary key can be edited only when a new record is created. They cannot be edited in the change mode.

In the query mode all elements belonging to primary keys or indexes are enabled. When the user enters a value the search activity is quick, because it is executed on the index and not systematically scanning the entire file.

Normally this option is defined for fields and not for values.

Comment

Brief comment on the element.

This comment is displayed in the element list of the edited entity. Fields are automatically initializated using the description defined in the design plan. For variables this must be set by the programmer.

Using the 'Add' button you can add a string next to the element and define the text and comment. This allows you to obtain a prototype for working variables quickly, which is in line with the dialog window.

Add

Using the 'Add' button you can add a string next to the element and define the text and comment. This allows you to obtain a prototype for working variables quickly, which is in line with the dialog window.

Repeated

Defines whether elements are repeated or not.

Repeated elements are stored in a temporary file. Each body row has its value. When you move to a new row you can edit the value associated to the selected element.

Repeated elements must not necessarily be within the grid. You can place them outside the grid setting the 'Fixed Position' option. These elements are displayed only once but as you move the cursor from one row to the other the fixed element displays the value associated to the highlighted row.

Editing

Editing status of the element.

Elements have three editing statuses:

Hide

Hidden elements are not displayed. They are used for calculations or as supporting variables.

Show

Showed elements are displayed on the screen but cannot be edited. They are used in links as return variables, to store calculation results or as support variable that must be checked by the user.

Edit

Editable elements can be changed by the user.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Evaluate

Defines whether the element is calculated.

Elements can be edited by the user or calculated by the procedure. When you define options involving calculations you need to define the expression in the 'Calc/Init/Def.' option in the 'Expressions' area. There are many calculation options:

'No'

When it is set the element is not calculated. Variables are initialized (blank string for characters, 0 for numbers, etc.).

'Calculate':

The element is calculated. The defined expression is checked every time a value affecting the calculation result is entered in the dialog window. When the calculation is simple this strategy works smoothly. When the calculation implies a slow routine you need to limit re-calculations. In the 'Calc/Link depends on' option you need to define which elements trigger the re-calculation. The calculation will be executed only when one of these elements changes. Calculated elements are normally 'shown' and not 'editable'. To make elements editable the calculation must not be re-executed at each input, but only when trigger elements are changed. Otherwise the calculation is re-executed also when a value is entered, thus overriding the input.

'Init':

The calculation is executed only once when the element is edited. This option is used to define a different default value then the one given automatically by the system.

'Default':

In some cases fields cannot be initializated together with the dialog window. This may happen because the field initialization depends upon some values that the user must input first.When the 'Default' option is set the calculation is executed the first time the cursor enters the element. In the initialization expression you can therefore define all elements that the user must enter first.

'Totalize':

When this option is set the elements sums up all values in the body rows. In the calculation expression you need to define the name of the element that must be totalized.

Checked/Linked

Defines whether the element is controlled or linked to a file.

To control the data that the user has input you need to define one of the options in the table below. If you select an option that requires a control you need to define the validation formula in the 'Checking' line in the 'Expressions' area.

'No':

The element is not controlled. Any value is accepted.

'Checked':

Any user input is controlled executing the 'Checking' formula. When the result is TRUE the value is accepted, otherwise an error message is displayed, the element is emptied and the cursor positioned on the element. The standard error message can be changed defining the 'Error message' expression.

'Linked':

The control activity involves a database table. The input value is searched in the database and is accepted only if the linked table contains a record that has the value as primary key. Detail on how this search activity is performed are defined in the 'Linked Table' page. 'Linked' elements can be also controlled by a 'Checking' formula. The value is first searched on the linked table. If found and returned the 'Checking' formula is executed.

Zoom

Defines whether there is an active zoom on the element.

Zooms are procedures run when the user presses the function key F9 when the cursor is on the related element.

'No'

No action is performed.

'User'

The expression defined in the 'Zoom' formula in the 'Expressions' area is executed.

'Standard'

When an element is liked to a table CodePainter creates a standard zoom to display and search data in the liked table. Parameters defined in the 'Linked Table' page influence the standard zoom. The 'Zoom' expression is used and a specific zoom configuration executed.

Get picture

Format of the input element.

In this option you can define the format that must be applied when the element is inserted.

You need to define an expression that is checked at run-time. Constant values must be defined between apexes.

Typical formats are:

"999.99": for numeric values.

"XXXXX": for characters.

Repl("!", 20): forces upper case in 20 characters strings.

When this option is not defined and you define a display format the input uses the latter. The '?' button creates a standard format for the element type.

Display picture

Format of the displayed element.

In this option you can define a format to be applied when the value is displayed. You need to define an expression that is checked at run-time. Constant values must be defined between apexes.

Typical formats are:

"999.99": for numeric values.

"XXXXX": for characters.

Repl("!", 20): forces upper case in 20 characters strings.

When no input format is defined the format defined for display is used also when data is input.

Zero filling

The value takes on zeros on its left until the value is filled up.

When the 'Autonumber' option is used on character keys entries are created as '00001', '00002' etc. The Zero filling option makes it easier for the user who is not required to manually fill the field with zeros. The system automatically adds zeros until the field length is reached.

This option works only is the entered value is numeric and does not start with '0'.

Obligatory

Mandatory element.

An editing phase cannot be terminated as long as no value has been defined for this element. When the user presses the function key F10 and no value has been entered an error message 'Obligatory field' is displayed. The cursor is positioned on the element and the editing mode is kept so that the user can enter a value.

Fixed position

Repeated elements in a fixed position.

In complex programs the grid associated to the body may be too small to contain all required values. You can fix the position of body's elements. The element is displayed only once in the dialog window. The element takes on different values depending on the active row.

During data input you will first input all fields in the grid, pass on to fixed elements and then go on to the next row.

Edit under condition

Defines whether the element can be edited only under specific conditions.

Elements may be editable only when specific conditions occur. To define conditions set the 'Edit' option to edit and define the 'Editing' formula in the 'Expressions' area that must be checked before the cursor is positioned on the element. When the logical expression returns a TRUE value the element is edited. Otherwise the cursor is passed on to the next element.

Hide under condition

Hides elements.

Elements may be hidden under specific conditions. To define these conditions set the 'Edit' or 'Show' option and define the 'Hiding' formula in the 'Expressions' area that must be checked. When the logical expression returns a TRUE value the element is hidden and therefore not displayed on the dialog window.

Calc/Init/Def.

Formula used for calculations.

When calculations formula are defined in the 'Evaluate' option the expression is used to calculate the element's value.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Simple Calculation

Define three elements: PRZART, QTAART and TOTART. The total (TOTART) must be calculated multiplying price and quantity.

The calculation formula must be defined as follows:

```
w_PRZART * w_QTAART
```

Editable Calculation

Consider the same three fields: PRZART, QTAART and TOTART. The total (TOTART) must be calculated and the element must be editable so that the user can adjust roundings manually.

The calculation formula must be defined as follows:

```
w_PRZART * w_QTAART
```

Further, in the 'Calc/Link depends on' option you need to define:

```
w_PRZART
```

```
w_QTAART
```

The calculation is executed only when the value for one of these fields changes.

Checking

Control formula for the element.

When elements are controlled or linked this expression is used to check whether values are acceptable or not. When the logical expression returns a TRUE value the value is accepted. When the returned value is FALSE the value is refused.

When the value is refused the default value is input in the element depending on the element type, an error message is displayed and the cursor passes on to the next element. The default message is 'Value not accepted', but you can change it defining the desired formula in 'Error message'.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Controlled Field

The field PRZART accepts only values greater or equal to zero:

```
w_PRZART>=0
```

Editing

Formula to activate the conditional editing.

This formula defines whether the field is editable or not. To define conditional editing you need to define the 'Edit' option in the 'Edit' area, set the 'Edit under condition' checkbox, and define the 'Editing' formula in the 'Expressions' area. When the logical expression returns a TRUE value the element is edited. When the returned value is FALSE the fields is disenable.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Hiding

Formula to hide the element.

This formula defines whether the field must be hidden or not. To define these conditions set the 'Edit' or 'Show' option, set the 'Edit under condition' checkbox and define the 'Hiding' formula in the 'Expressions' area. When the logical expression returns a TRUE value the field is hidden. When the returned value is FALSE the field is displayed.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Zoom

Zoom formula.

When elements have an active zoom this formula is used to define what must be executed when the user presses the function key F9.

When the 'Zoom' option is set to 'User' in this formula you need to define what must be executed. The combobox next the option define the type of zoom. Zoom types are:

'User program': the formula is a program line.

'Mask': the formula defines the dialog window that must be executed as zoom.

'Batch': the formula defines a routine created using the Routine Painter.

'UTK Object': the formula contains the output configuration name that must be displayed as zoom.

When the 'Zoom' option defines a 'Standard' zoom the formula can contain the name of a specific configuration that must be loaded in the zoom window.

Error message

Text for the error message.

When the element is controlled the user could enter values that are refused. In these cases the error message 'Value is not correct' is displayed. In this option you can define custom error messages for each element.

Checked/Linked

Defines whether the element is controlled or linked to a file.

To control the data that the user has input you need to define one of the options in the table below. If you select an option that requires a control you need to define the validation formula in the 'Checking' line in the 'Expressions' area.

'No':

The element is not controlled. Any value is accepted.

'Checked':

Any user input is controlled executing the 'Checking' formula. When the result is TRUE the value is accepted, otherwise an error message is displayed, the element is emptied and the cursor positioned on the element. The standard error message can be changed defining the 'Error message' expression.

'Linked'

The control involves a database file. The input value is searched in the database and is accepted only if the linked table contains a record that has the value as primary key. Detail on how this search activity is performed are defined in the 'Linked Table' page. 'Linked' elements can be also controlled by a 'Checking' formula. The value is first searched on the linked table. If found

and returned the 'Checking' formula is executed.

Evaluate

Defines whether the element is calculated or not.

Elements can be edited by the user or calculated by the program. When the calculation option is selected you need to define the expression that produces the result in the 'Calc/Init/Def.' option in the 'Expressions' area. There are many calculation options:

'No'

When 'No' is set the element is not calculated. For variables the blank value is initializated (blank string for characters, 0 for numbers, etc.).

'Calculate':

The element is calculated. The defined expression is checked every time a value affecting the calculation result is entered in the dialog window. When the calculation is simple this strategy works smoothly. When the calculation implies a slow routine you need to limit re-calculations. In the 'Calc/Link depends on' option you need to define which elements trigger the re-calculation. The calculation will be executed only when one of these elements changes. Calculated elements are normally 'shown' and not 'editable'. To make elements editable the calculation must not be re-executed at each input, but only when trigger elements are changed. Otherwise the calculation is re-executed also when a value is entered, thus overriding the input.

'Init':

The calculation is executed only once when the element is edited. This option is used to define a different default value then the one given automatically by the system.

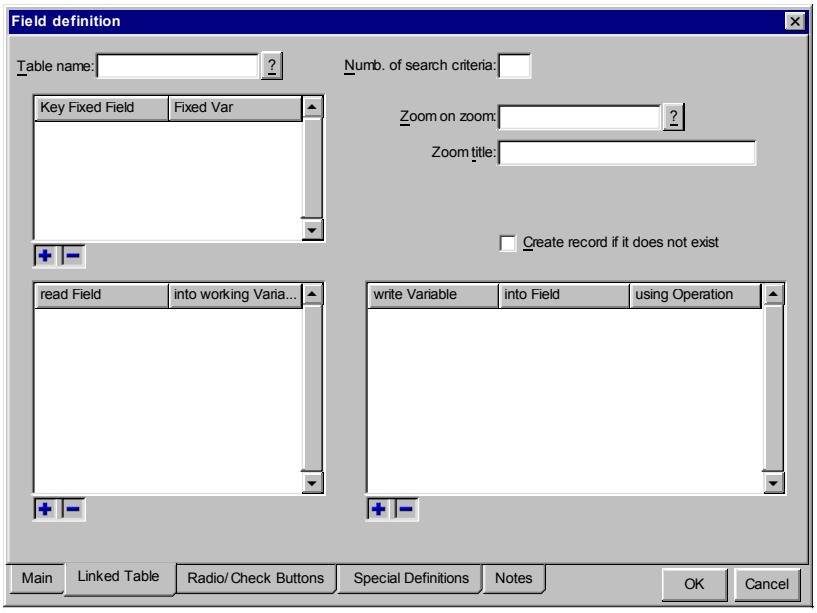
'Default':

In some cases fields cannot be initializated together with the dialog window. This may happen because the field initialization depends upon some values that the user must input first.When the 'Default' option is set the calculation is executed the first time the cursor enters the element. In the initialization expression you can therefore define all elements that the user must enter first.

Linked Table Page

In the second page you need to define the rules for the relationship between an element and a file record. The prototype defaults automatically what you defined in the Design phase.

When you define links in the prototype without going back to the design plan the relationship is not programmed in the database. This means that referential integrity for this relationship is not established. Values are checked only when they are entered in the dialog window.



Picture 267 - Field Definition: Page 2

Table name

Name of the related table.

Define the name of the related table. The value is searched in the related table basing on the parameters defined in this dialog window.

The table name is added to the workfiles list so that when the program is run the system checks the existence of the related table. Once you enter the table name, all lists helping the programmer to select fields' names are activated. Moreover, the procedure name that must be activated for the 'zoom on zoom' option is initialized. This procedure is a zoom that can be executed pressing F9 within an active zoom.

The '?' button displays the list of tables available in the database.

Numb. of search criteria

Defines the number of search criteria.

Using this option you can change the way records are searched in the related table. SQL sentences use fields defined in the 'Read Field ... into working Variables' area as search criteria. If the first search goes wrong the SQL sentence uses the second field in the list and so forth as long as there are search criteria.

Define an element 'ARTORD' which is related to the 'Items' file. Your search criteria is CODART and you need to return the field DESART.The standard search criteria is:

```
CODART=w_ARTORD
```

If you define the 'Read Field' list as follows:

```
CODART, w_ARTORD


DESART, e_DA_ORD
```

and the first search criteria does not lead to any result, no error message is displayed and the search activity is executed:

```
DESART=w_ARTORD
```

Only if the second search goes wrong as well the defined value is refused.

The example highlights how the current fields is searched as code first and as description afterwards. If the result is given by the second search criteria values are returned in the defined sequence.

This kind of alternative search is performed for the number of defined criteria, following the list of fields that must be returned. Fields must always be of the same type, you cannot mix e.g. numbers and characters.

Fixed fields belonging to the key are used also used for alternative searches.

Zoom on zoom

Procedure which is executed pressing F9 on an active zoom.

When the user presses F9 a zoom window on the related table is opened. If searched values are not found pressing F9 again (on the active zoom) a new window is opened that allows managing the linked file.

This option allows to define the procedure that must be executed to zoom on an active zoom. Normally the standard procedure to manage the related table is defined. This means that when you select the file reading design definitions, this field is initializated by CodePainter.

Zoom title

Contains the title of the zoom window opened.

Create record if it does not exist

Defines whether a record must be created on the related table.

Linked fields must find a related value in the linked table. This is true for all elements input by the user, because they are validated and not found values are refused.

When linked elements are hidden values are never entered and thus never checked. For example an element may be calculated basing on another element on the screen but linked to another table. It is therefore possible to write totals in a record that does not exits, i.e. the related record is missing.

Using this option the search activity can go on. A new record is created and filled in the related table using the list of fixed keys of read fields and totals. Returned values are used to attribute values to fields.

This happens especially when you have Parent/Child relationships whereby totals are required to update the Child entity.

Key Fixed Field

Fields building the primary key, excluding the current field.

This list is used when the primary key of the related table is composite.

The search activity is started when the user enters a value in the current field, which in turn is placed in line with the first element in the list right under the value. Primary key fields are placed next to the corresponding values in the dialog window according to the logic defined in this list. For each field there is value, which has been read by a working variable.

Composite Key

The primary key of the linked table is made of the fields: CODMAG and CODART. In the Dialog Window there are two elements, MAGORD and ARTORD asking the user to input the warehouse code and the item code on which he/she wants to work.

In the ARTORD element you need to define:

```
CODMAG, w_MAGORD
```

In 'read Fields' define:

```
CODART, w_ARTORD
```

The search activity in the related table will have matches of the following kind:

```
MAGART=w_MAGORD and CODART=w_ARTORD
```

The standard zoom uses these fields as filter on the table so that only a selection of the table is displayed and not the entire related table. In the example above after that the user enters a value for the warehouse the zoom window will display only the items in that warehouse.

read Field

List of fields that must be read from the related table.

This list drives the relationship with the related table. When the user enters a value in the current element the search activity is started. The selection criteria is given by the first field in this list and the fixed fields defined in the 'Key Fixed Fields' area (optional).

Example

If the related file has the key CODART and the current element is ARTORD the search activity is as follows:

```
CODART=w_ARTORD
```

If the entered value is found the link is successful and all fields defined in the list are read and values are returned to the corresponding working variables.

If the value is not in the file the error message 'Value is not correct' is displayed and the element is reset.

The search activity can also find partial values, e.g. the user in enters 'Smith' and the value in the file is 'Smith John'. In these cases the value entered in the field is completed with the value found in the table.

When the search activity finds more values (e.g. the user enters 'Smith' and the values in the table are 'Smith John' and 'Smith Bob') a zoom window containing the list of matches found is opened and the user can select the desired record.

When the search activity goes wrong but more search criteria are defined, alternative searches are executed as defined in 'Numb. of search criteria'.

Write Variable

This list defines totalization transactions towards the related table.

This list is made of three columns: the first column defines the variables that will be summed up, the second the fields that will be increased in value and the third the operations that must be performed.

In the third column you can define the constants '+', '-', and '=' to get totals, reversals or a data entry. You can also define fields with one character that contain the transaction that must be executed. When this value is blank no transaction is performed. Transaction types must be stored in a field and not in a variable because CodePainter requires this information to perform reversals in case of changs or deletions.

Example

Your application has two tables, namely 'Items' with its fields CODART, DESART and QTAART, and 'Orders' with the fields ARTORD and QTAORD.

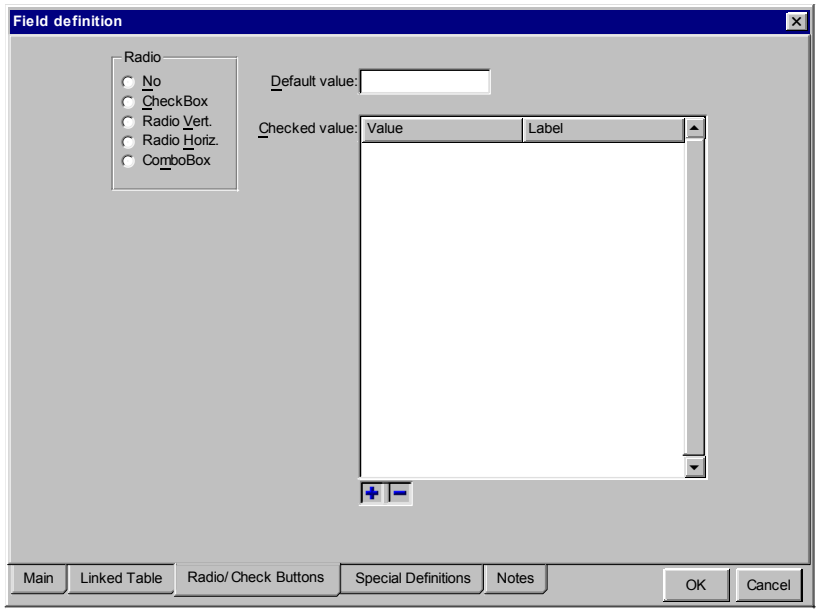
You want to sum ordered quantities in QTAART. Define the 'Write Variable' list for the element ARTORD as follows:

```
w_QTAORD, ARTORD, +
```

Radio/Check Buttons Page

The third page contains parameters to change the standard input/ output textbox in other input/ output structures.

Picture 268 - Field Definition: Page 3



Radio

To display the element as checkbox, radio or combobox.

Fields and variables are created as textboxes. This option allows you to change the method in which data is entered and displayed.

Default value

Defaulted value when the user does not select any value.

Value

List of variables and labels.

This list is made of two columns: the first defines the value which is given to the element, the second the label that must be displayed.

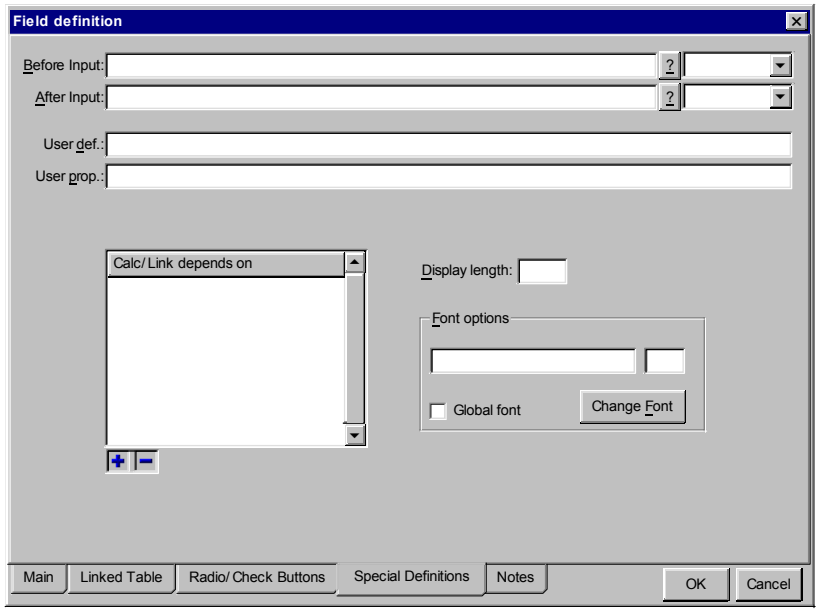
Values must be defined as constants in the target language. Strings must therefore be written between apexes. Labels are always text constants therefore they do not need to be written between apexes.

Special Definitions Page

The fourth page defines properties, which are used seldom.

Rather then using 'Before' and 'After' it is advisable to use the structured options in the first page. 'User Def' require customized templates. 'Depends on' are used for optimizations.

Picture 269 - Field Definition: Page 4



Before input

Program that must be executed before the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low-level and cannot be defined so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

After input

Program that must be executed after the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low-level and cannot be defines so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

User def.

Free definition.

Could be used by a custom template.

User prop.

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to add definitions to the ones generated by CodePainter.

Example

```
forecolor=RGB(255, 0, 0)
```

Calc/Link depends on

List defining the elements on which the calculation execution or the link execution depends on.

Calculations and links are normally executed every time a value is entered. This strategy works smoothly if the calling routine is quick. If calling routines take long to be terminated you need to limit the number of calls.

Defining working variables in this list, calculations and associated links are executed only when the value for one of these variables changes.

This list is also used for calculated elements, which can be edited. Elements will remain editable, but they will be recalculated only when the value for one of these variables changes. Here you should define all calculation parameters associated to the element.

Using this method you avoid executing many line codes. Nevertheless checking the change of the value is costly and therefore this option should be activated only when calculations are complex.

The same applies to link executions.

Example

A field is linked to the 'Items' file. The field is returned to the linked 'VAT' file. When item's data is entered the VAT value must be defaulted. The field must be editable so that the VAT value can be returned by the first link.

You need to define the 'item number' field in the 'Depends on' list of the 'VAT' field. When the user selects an item the link with the VAT file is re-executed. The VAT value can be freely changed. If the user goes back to the 'item number' field and enters a new value, the link to the VAT file is re-executed.

Display length

Length used to display the element.

This option is used so that CodePainter calculates a length, which is different from the one given by the element parameters.

Change Font

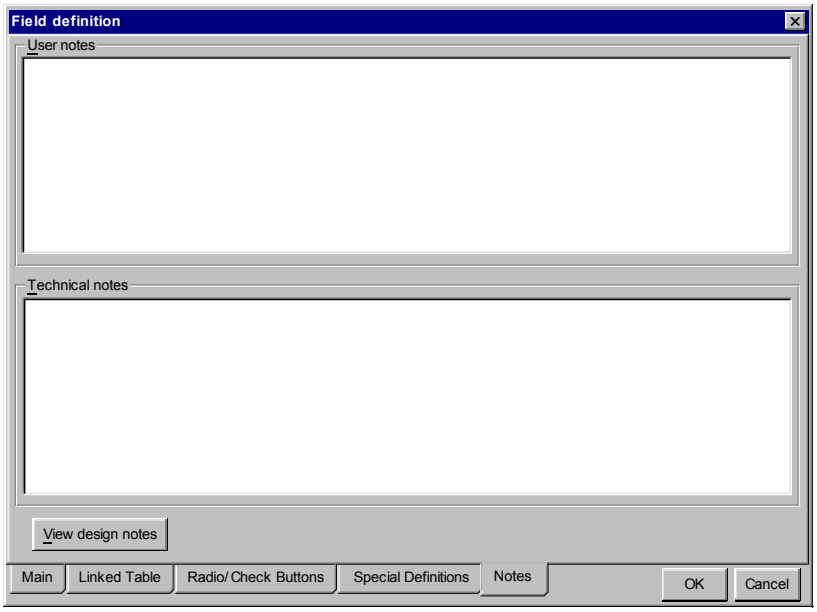
Defines the elements font. If no font is defined the dialog widow's default font is used.

Global font

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.



Picture 270 - Field Definition: Page 5

User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

Technical notes on the element.

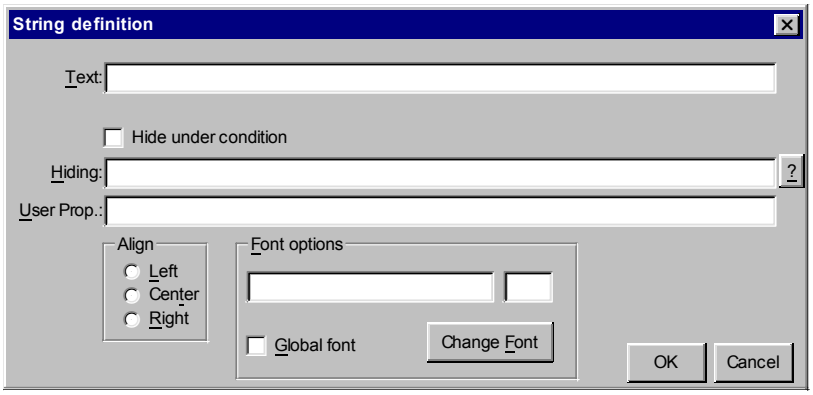
These notes are used to create technical documentation.

View design notes

6.7.2 String Definition

Dialog window to define the characteristics of comment strings.

Picture 271 - String Definition



Text

String text.

Hide under condition

Hides the string.

To display strings only when specific conditions are met you need to define the hiding formula in the 'Hiding' textbox. When the returned value is TRUE the string is hidden, and is not displayed.

Hiding

Formula defining the conditions to hide the string.

This formula defines whether the string must be hidden or not. To hide the string under given conditions you need to set the checkbox 'Hide under condition' and to define a logical expression in this field. When the result is TRUE the string is hidden, when the result is FALSE the string is displayed.

Clicking the '?' button the list of elements in the dialog window is displayed. These elements can be used to define expressions. Please note that while in the Editing mode CodePainter stores values in working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

User prop

Control properties defined by the programmer.

Elements on the screen are generated as a 'control'. This option allows you to add other specific controls.

Example

```
forecolor=RGB(255,0,0)
```

Align

String alignment.

Defines how the string must be positioned within the string box.

'Left'

Aligned to the left.

'Center'

Centered.

'Right'

Aligned to the right.

To set the window layout you should align strings to the right. You should not align to the left and create boxes that exactly fit the strings. The end-user may use a different font from the one defined and therefore strings may be bigger or smaller. In these cases strings would be no longer aligned.

When you need to translate the application you need to align strings to the right as words in other languages may be longer or shorter. Using CodePainter you can automatically translate dialog windows and set a different language for each user.

Prototypes are created using these principles, i.e. alignment to the right and field space longer than required.

Change Font

Defines the strings' font. If no font is defined the dialog widow's default font is used.

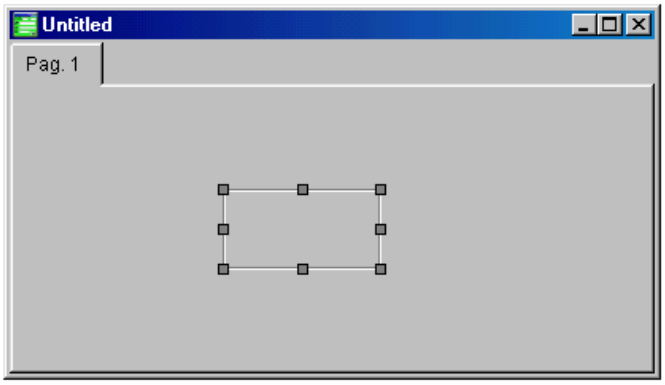
Global font

When selected the font defined in the 'Global Font' menu is applied to the current element.

6.7.3 Box

This option allows you to add a Box element to the entity in use.

Picture 272 - Box



Boxes are used to graphically divide the form. Boxes can be used as vertical or horizontal lines or rectangles.

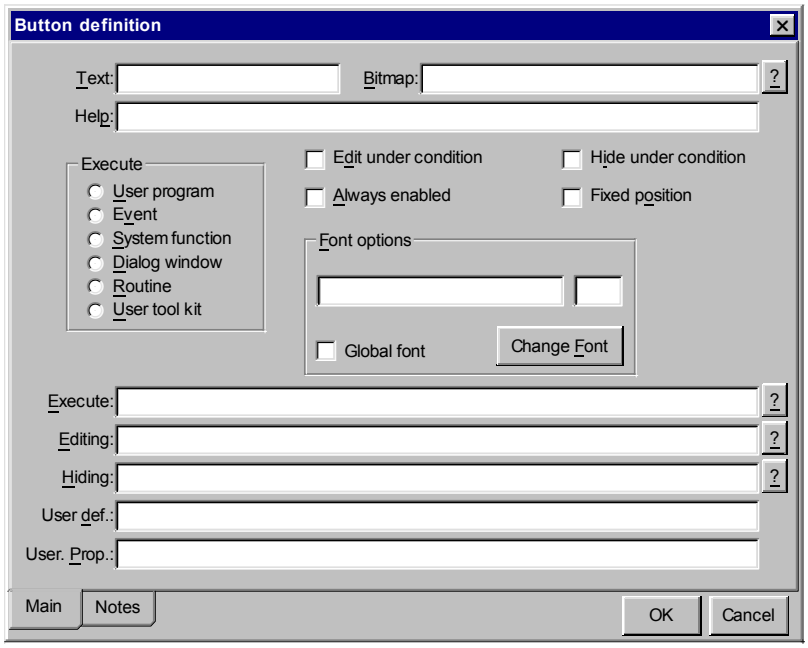
Boxes are changed into lines '*collapsing*' them using the mouse or **<Shift>** + **<Cursor Keys>**.

6.7.4 Button Definition

Dialog window defining button options. Buttons are added into dialog windows to execute procedures such as reports, additional dialog windows, etc.

Main Page

In the first page you can define how the button is displayed, activated and its functions.



Picture 273 - Button Definition: Page 1

Text

Text that must be displayed inside the button.

Bitmap

Bitmap displayed inside the button.

Help

Short 'Help' text.

The text defined in this option is used to associate a tooltip to the element. When the user positions that mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Execute

Defines how CodePainter must interpret the execution command line when the button is clicked.

'User program'

Executes a program defined by the developer.

'Event':

Notifies an event.

'System function':

Executes a system function.

'Dialog window':

Opens a dialog window created using the Dialog Window Painter.

'Routine':

Executes a routine developed using the Routine Painter.

'User Tool kit':

Executes an output created using the run-time framework.

Depending on the command type defined in 'Execute' different values are required. 'User Program' is copied in the source. 'Event' requires the name of the event that must be notified. 'System function' requires the name of the function that must be activated. 'User tool kit' requires the name of the query followed by the output format name. In all other cases the name of the file that must be executed is required

Edit under condition

Defines whether the button is enabled only under specific conditions.

To enable buttons only when specific conditions are met you need to define a formula that must be validated before the cursor is positioned on the element.

Setting this option the 'Editing' formula in the textbox is activated. The formula must be a logical expression. When the returned value is TRUE the button will be enabled, otherwise the cursor is passed on to the next element.

Hide under condition

Hides the button.

To hide buttons under specific conditions you need to define the 'Hiding' formula in the textbox. The formula will determine whether the button must be displayed or hidden. If the logical value TRUE is returned the element is hidden otherwise the button is displayed.

Always enabled

The button is enabled in 'Editing' or 'Query' modes.Buttons are used to access procedures that are bound to the main dialog window. To use these procedures may make sense in the 'Editing' and/or in the 'Query' mode only. E.g. a button executing a report should always be enabled.

Setting this flag the button will be enabled in the 'Editing' as well ad in the 'Query' mode. When the flag is not set the button is enabled in the 'Editing' mode only.

Fixed position

The button is repeated in a fixed position.

In complex programs the grid associated to the body may be too small to contain all values. You can define that a specific button in the body is in a fixed position outside the grid and that it is displayed only once in the window. The button is still linked to the row but is displayed only once.

During data input you will first input all fields in the grid, pass on to fixed elements and then go on to the next row.

Execute

Defines what must be executed when the button is pressed.

The meaning of this line depends on what has been defined for the radio button 'Execute'. The button on the left allows quick access to the following definitions:

'User program':

The code line is copied into the source. Using the '?' button you can access the list of variables contained in the dialog window.

'Event':

Defines the name of the Event that must be notified.

'System function':

Executes a CodePainter function. Clicking the '?' button the list of available functions is displayed.

'Dialog window':

Opens a dialog window created using the Dialog Window Painter. The '?' button lists all Dialog Windows available in the project.

'Routine':

Executes a routine developed using the Routine Painter. Beside the routine name you can also define in brackets the parameters that must be passed on to the routine.

'User Tool Kit':

Executes an output created using the run-time framework, typically a query created using the Query Painter. A dialog window is opened in which you can select the kind of output required: preview, report, word, excel or graph. To execute the output you need to add a comma after the query name and digit the output name.

Editing

Formula to set conditions to allow the 'Edit' mode.

This formula defines whether the button is active or not. To define conditions you need to set the 'Edit under condition' option and define a logical expression. When the expression result is TRUE the button is enabled. When the result is FALSE the button is disenabled.

Clicking the '?' button the list of all elements in the dialog window is displayed. These elements can be used to define the expression. Please note that during the Editing mode CodePainter stores the values of working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

Hiding

Formula to hide the button.

This formula defines whether the button must be hidden or not. To hide the button you need to set the checkbox 'Hide under condition' and to define a logical expression in the formula. When the result is TRUE the button is hidden, when the result is FALSE the button is FALSE.

Clicking the '?' button the list elements in the dialog window is displayed. These elements can be used to define the expression. Please note that while in the Editing mode CodePainter stores the values of working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

User def

Free definition.

This option can be used by custom templates.

User prop

Control properties defined by the programmer.

Elements on the screen are generated as a 'control'. This option allows you to add other specific controls.

Example

```
forecolor=RGB(255, 0, 0)
```

Change Font

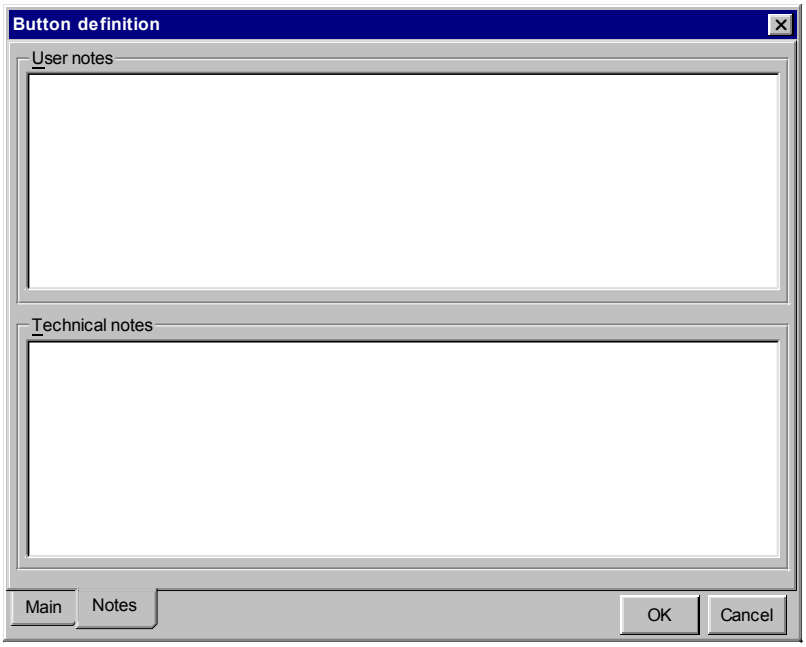
Select the font to be applied to this element. When no font is defined the default font used by the dialog window is applied.

Global font

Notes Page

Notes that are used to create the documentation.

Picture 274 - Button definition: Page 2



User Notes

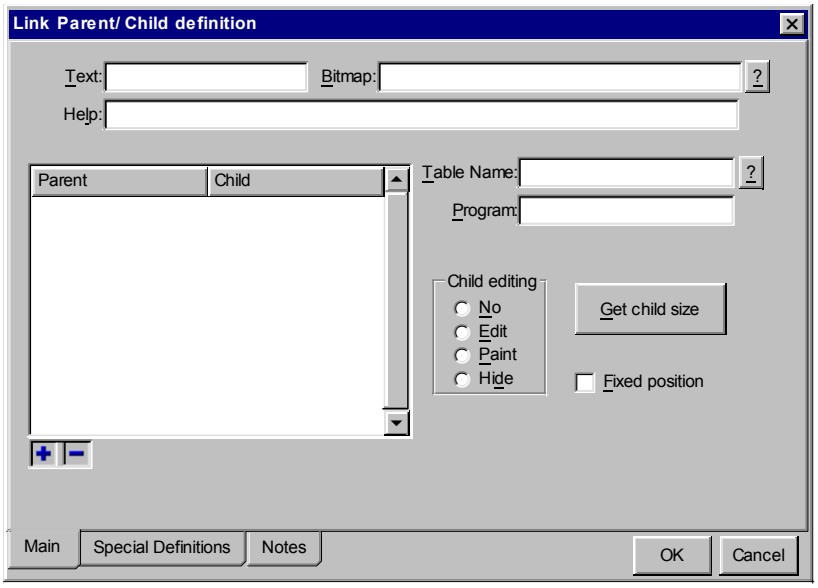
Notes on the element.These notes are used to create user documentation.

Technical Notes

Technical notes on the element.These notes are used to create technical documentation.

6.7.5 Link Parent/Child Definition

Main Page



Picture 275 - Link Parent/Child Definition: Page 1

Text

Text displayed within the button.

Bitmap

Bitmap displayed within the button.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Parent/Child

List of key fields that bind the Parent to the Child.

This list defines all fields that relate the Parent object to the Child object. CodePainter will pass on these fields when a search activity is required or in Load mode when the Child requires the Parent's key.

Table Name

Name of the Child file.

Program

Name of the Child file's managing procedure.

Child editing

Child's editing mode.

'No'

The Child is activated when the button is clicked.

'Edit'

The Child's managing procedure is integrated in the Parent's dialog window.

'Paint'

The Child is in a different window, but is activated with the Parent window.

'Hide'

The Child is hidden. Data in the Child is canceled when it is canceled in the Parent. Data in the Child cannot be changed.

Get child size

Measures the Child's editing window and adjusts the Parent's window so that the Child fits in.

This button is used when the 'Child editing' option is set to 'Edit'.

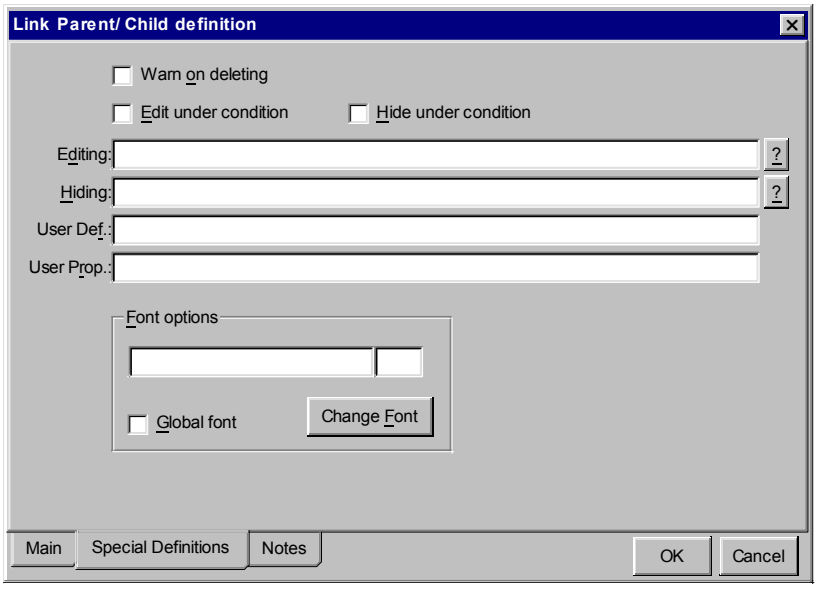
Fixed position

Repeated button in a fixed position.

In complex programs the grid associated to the body may be too small to contain all required values. You can fix the position of the button in the body. The button is displayed only once in the dialog window, and is still bound to the row taking up only reduced space.

During data input you will first input all fields in the grid, pass on to fixed elements and then go on to the next row.

Special Definitions Page



Picture 276 - Link Parent/Child Definition: Page 2

Warn on deleting

When you are about to delete records of the Parent entity this option warns you that the Child entity data will be deleted as well.

Behind small windows there may be a number of fields hidden in a Parent/Child link button. To avoid deleting this hidden data by mistake you can set this checkbox. Before deleting the system checks the Child's contents. If data is found a message asking you to confirm the delete command is displayed.

Edit under condition

Defines whether the button is enabled only under specific conditions.

Buttons may be enabled only when specific conditions occur. To define conditions set the 'Edit' option to edit and define the 'Editing' formula in the textbox that must be checked before the cursor is positioned on the button. When the logical expression returns a TRUE value the button is enabled. Otherwise the cursor is passed on to the next element.

Hide under condition

Hides buttons

Buttons may be hidden under specific conditions. To define these conditions set the 'Edit' or 'Show' option and define the 'Hiding' formula in the textbox that must be checked. When the logical expression returns a TRUE value the button is hidden and therefore not displayed on the dialog window.

Editing

Formula to activate the conditional enabling.

This formula defines whether the button is enabled or not. To define conditional enabling you need to define the 'Edit' option in the 'Edit' area, set the 'Edit under condition' checkbox, and define the 'Editing' formula in the textbox. When the logical expression returns a TRUE value the button is enabled. When the returned value is FALSE the button is disenable.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Hiding

Formula to hide the button.

This formula defines whether the button must be hidden or not. To define these conditions set the set the 'Edit under condition' checkbox and define the 'Hiding' formula in the textbox. When the logical expression returns a TRUE value the button is hidden. When the returned value is FALSE the button is displayed.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

User def

Free definition.

Could be used by a custom template.

User prop

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to add definitions to the ones generated by CodePainter.

Example

```
forecolor=RGB(255,0,0)
```

Change Font

Defines the elements font. If no font is defined the dialog widow's default font is used.

Global font

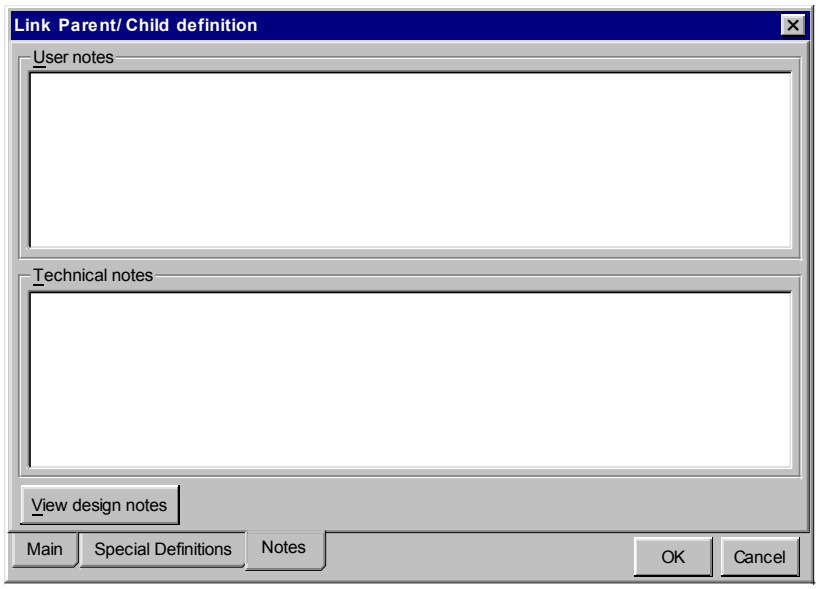
When selected the font defined in the 'Global Font' menu is applied to the current element.

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.

Picture 277 - Link Parent/Child Definition: Page 3



User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

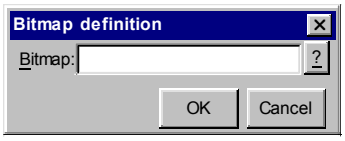
Technical notes on the element.

These notes are used to create technical documentation.

View design notes

Quick access to the Design notes of the current element.

6.7.6 Bitmap Definition



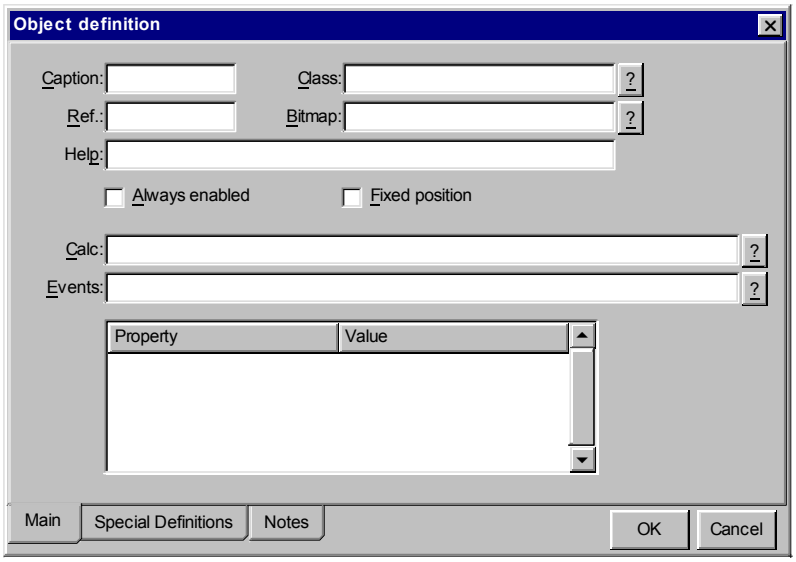
Picture 278 - Bitmap Definition: Page 1

Bitmap

6.7.7 Object definition

Main Page

Picture 279 - Object Definition: Page 1



Caption

Object text.

This text is used as title for those objects that need a string to be displayed.

Class

Object class.

Objects that can be added to dialog windows must belong to predefined classes. CodePainter uses a run-time class library or dedicated templates to generate the relevant source code. The class identifies which tool must be used.

Clicking the '?' button you can access all classes installed in your development environment. For more information on classes and their use, please refer to the COMPONENT GUIDE.

Ref.

Name of the variable associated to the object.

Objects are created as 'controls' in the current form. Associated working variables are not created, because objects are not read by CodePainter's standard procedures.

When you require to access the object from a manual area or a routine, you first need to define the variable name in this property. The working variable created works like the ones associated to fields or variables and is initializated with the pointer to the object.

Bitmap

Bitmap associated to the object.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Always enabled

The object is enables in 'Editing' and 'Query' mode.

Objects are used to access procedures that are bound to the main dialog window. Using these procedures may make sense in the 'Editing' and/or in the 'Query' mode only. E.g. an object executing a graph should always be enabled.

Setting this flag the object will be enabled in the 'Editing' as well as in the 'Query' mode. When the flag is not set the object is enabled in the 'Editing' mode only.

Fixed position

The repeated object is in a fixed position.

In complex programs the grid associated to the body may be too small to contain all required values. You can fix the position of body's elements. The element is displayed only once in the dialog window. The element takes on different values depending on the active row, but takes up only reduced space.

During data input you will first input all fields in the grid, pass on to fixed elements and then go on to the next row.

Calc

Passes on values to the object every time the dialog window is re-calculated.

The input dialog window is recalculated when specific actions are performed: when a new record is input, or when the user inputs specific fields, etc. Each time the dialog window is re-calculated it notifies the object passing on as parameter the value defined in this expression. The object will react according to the class to which it belongs.

For example the 'Dash Board' ('cruscotto') changes the position of the arrow, the 'Calendar' ('calendario') selects the current date, etc.

For more information on how each class exploits recalculations please refer to the COMPONENT GUIDE.

When calculations are complex you can limit re-calculations defining the 'Depends on' property as for any field or variable.

Events

List of events to which the object is hooked.

Objects can communicate with the dialog window in which they are in either through recalculations or events.

Events are notified by the dialog window. The object will react according to the class's specifications to which it belongs. Clicking the '?' button next to properties you access the list of available events. For more information on how each element answers to a given event, please refer to the COMPONENT GUIDE.

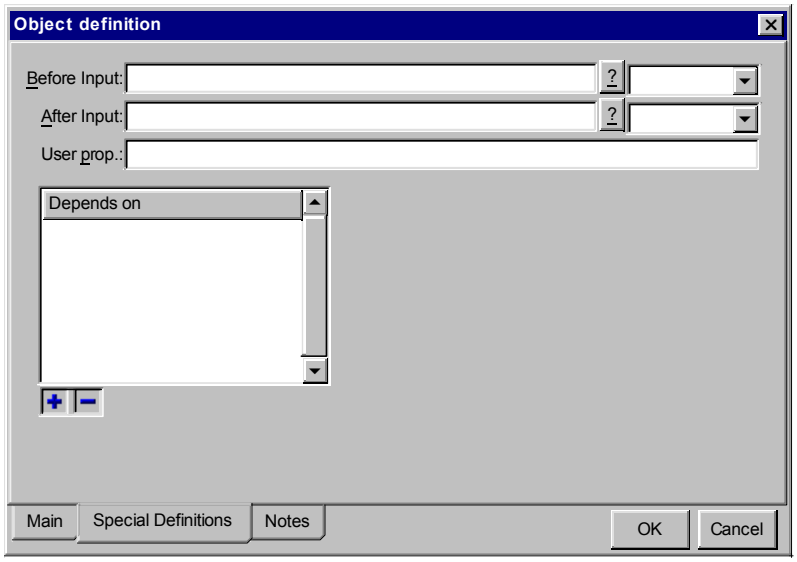
Properties

List of object properties.

Objects have a set of properties that differ according to the class to which they belong.

This list displays the property name in the first column and the default value in the second. These values can be changed in order to configure the object. For more information on object properties, please refer to the COMPONENT GUIDE.

Special Definitions Page



Picture 280 - Object Definition: Page 2

Before input

Program that must be executed before the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low level and cannot be defined so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

After input

Program that must be executed after the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low level and cannot be defines so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

User prop

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to adddefinitions to the ones generated by CodePainter.

Depends on

List defining the elements on which the calculation execution for associated objects depends.

Calculations and links are normally executed every time a value is entered. This strategy works smoothly if the calling routine is quick. If calling routines take long to be terminated you need to limit the number of calls.

When defining working variables in this list, calculations and associated links are executed only when the value for one of these variables changes.

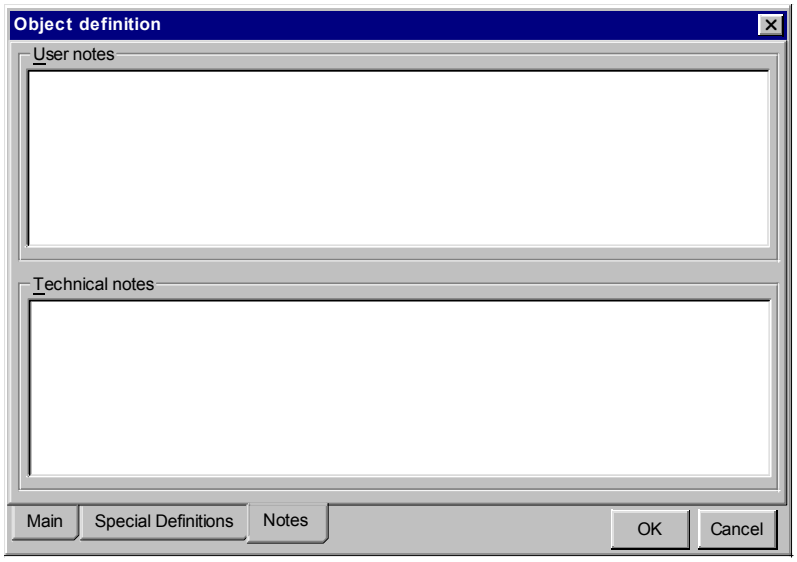
In this list you should define all calculation parameters associated to the element.

Using this method you avoid executing many code lines. Nevertheless checking the change of the value is costly and therefore this option should be activated only when calculations are complex.

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.



Picture 281 - Object Definition: Page 3

User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

Technical notes on the element.

These notes are used to create technical documentation.

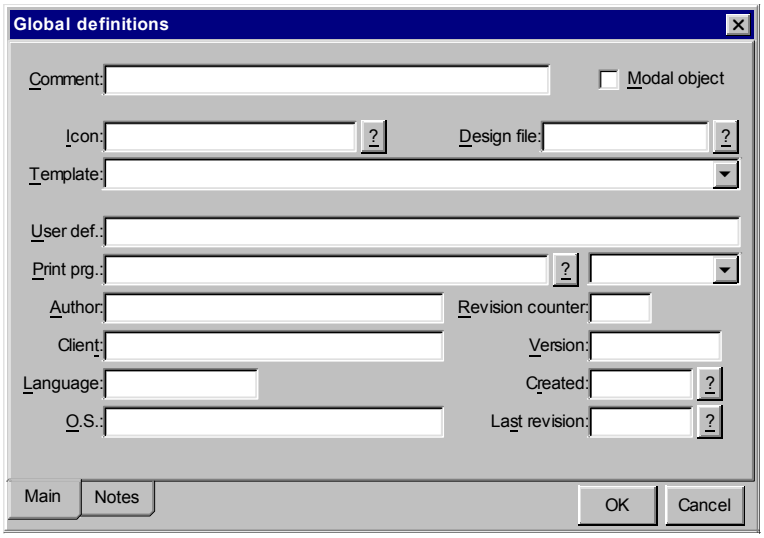
6.8 'Globals' Menu Advanced Options

This section details the 'Globals' menu functions.

6.8.1 Global Definitions

Main Page

Picture 282 - Global Definitions: Page 1



Comment

Brief comment on the entity. This comment is displayed in the project file so that you can search entities by file name (i.e. the associated procedure), or by brief description.

Modal object

Defines whether the entity is a modal dialog window or not. Modal dialog windows ask to terminate the input before you can go on to another dialog window. This means that the user cannot use the mouse to go to other windows until the current window is saved (F10) or exit (Esc).

Icon

Icon associated to this entity.

Template

Name of the template used for code generation. In this property you need to define the name of the template used to generate the source code. Templates are procedures' skeletons, finalized by the programmer to generate the required source code. Change the template and the generated source code changes as well.

User def

Free string. Can be used by custom templates.

Print prg.

Program activated during queries pressing F2.

In this property you can define the program that must be called when the user clicks the 'Print' button during queries. The combobox next to the property defines how the command line must be interpreted.

'User program': the command is copied in the source code. 'Mask': a dialog window created with the Dialog Window Painter is opened. 'Batch': a routine created with the Routine Painter is executed. You can also pass on parameters defining them ion brackets. 'UTK Object': defines an output configuration created with user tools such as the Query Painter.

The '?' button next to the property displays a list of possible choices depending on the defined activity.

Author

Author of the program. Short text defining the developer in charge of the entity.

Client

The commissioner. Short text to define the customer who commissioned the entity.

Language

Development language. Short text describing the programming language used to develop this entity.

O.S.

Operating System. Short text defining the limits given to the entity by the operating system.

Revision counter

Revision counter. This read only property shows you how often the entity has been changed. The number is increased automatically every time the entity is saved. This property can be used to check whether copied objects have been changed by the original author.

Version

Entity version number.

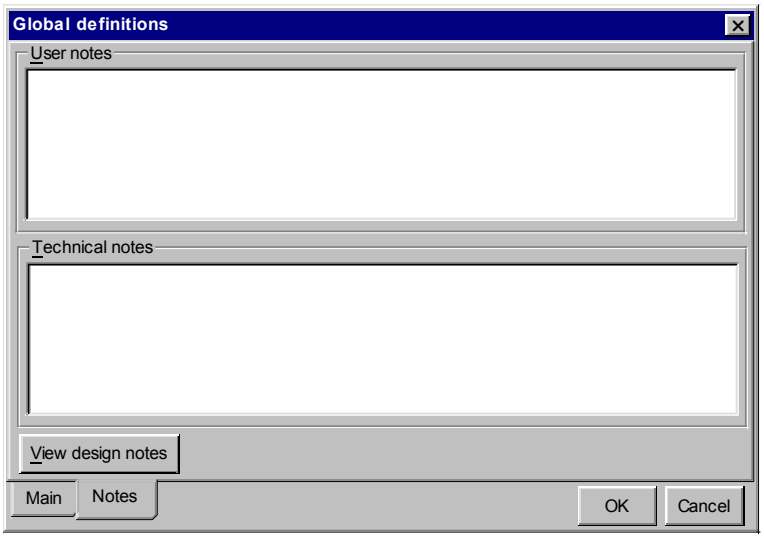
Created

Date in which the entity was created.

Last revision

Date in which the last revision was made.

Notes Page



Picture 283 - Global Definitions: Page 2

User Note

Notes on the element. These notes are used to create user documentation.

Technical Notes

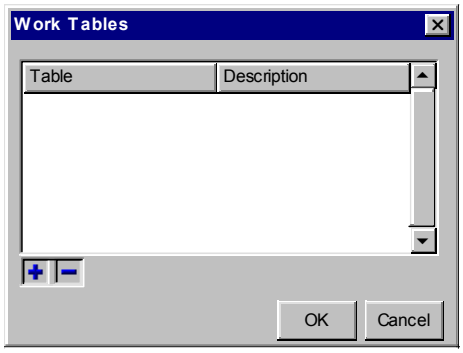
Technical notes on the element. These notes are used to create technical documentation.

View design notes

Quick access to the Design notes of the current element.

6.8.2 Tables

Picture 284 - Tables



Tables

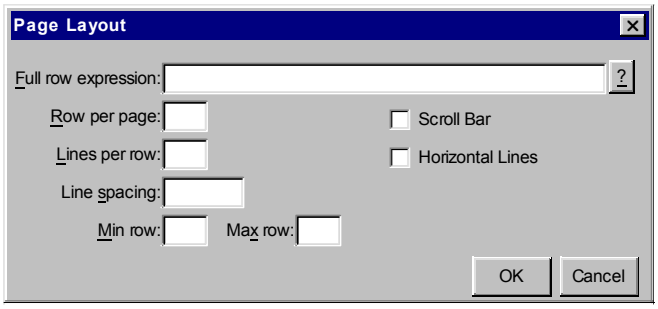
List of tables.

The generated procedure controls that the database contains all tables displayed in this list. When one or more tables are missing an error message is displayed and the procedure is not executed. This is done to avoid errors.

The procedure can also use tables which are not included in the list. Programs should not be executed if they can generate errors.

Tables used in links are automatically added to the list.

6.8.3 Page Layout



Picture 285 - Page Layout: Page 1

Full Row Expression

Full row expression.

Rows are considered completed only when the expression is TRUE. When the user tries to go from one row to another this expression is checked. When the row is considered full and the cursor is on the last row of the body a new row is added, otherwise the cursor goes to the first field of the footer.

Row per page

Number of rows in the body.

In this property you need to define the number of rows in the entity's body. By increasing this number the grid is enlarged so that it can display more rows.

Line per row

Number of lines per row.

The high of each row is a slightly higher than input fields basing on default fonts. Changing this property you can increase rows' size and have more space for data in the body.

Increasing rows' space the overall space used by the body increases as well. The dialog window margins may be overridden. To stay within margins you need to decrease the number of rows per page.

Line spacing

Space between single lines in the row.

Each body row is made of one or more lines. In this property you can define the lines' heigh.

Min. row

Minimum number of rows required for the entity.

When the data input is saved the number of rows in the body is checked. When the number is lower than the one defined an error message is displayed and the body is edited again.

Max rows

Maximum number of rows allowed for the entity.

When the data input is saved the number of rows in the body is checked. When the number is higher than the one defined an error message is displayed and the body is edited again.

Scroll Bar

When selected the vertical scroll bar is enabled.

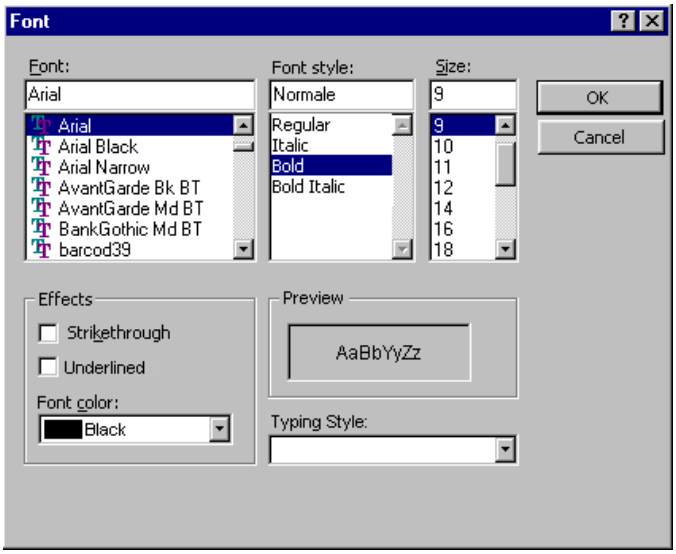
Horizontal Lines

When selected the horizontal lines dividing page's elements are enabled.

6.8.4 Font

Allows changing default fonts for the entity in use.

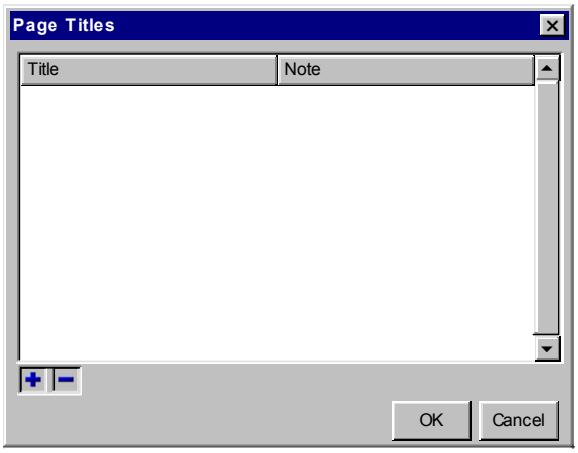
Default fonts in this area are the ones you defined in CodePainter's Front End in the 'Project' menu in the 'Project Options' option.



Picture 286 - Font Definition

6.8.5 Page Titles

Picture 287 - Page Titles: Page 1



Titles

Page titles. Entities can have dialog windows having more pages. The default title is 'Pag.' followed by the page number. This list allows changing pages' title.

6.8.6 Autonumber

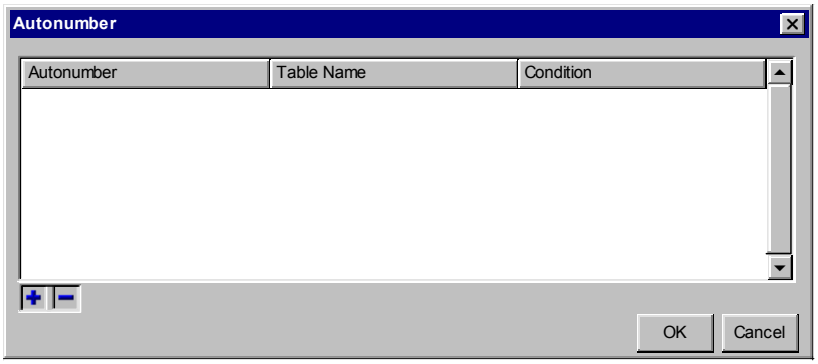
Manages progressive numbering automatically.

You can define fields that are numbered automatically in each entity. Once the starting number is defined the field is increased every time a new record is input.

When the 'autonumber' option for a field is defined, the field is initialized with the next value in the numbering sequence when a new record is input. If the user does not change the value and the record is saved the table containing progressive numbering is read. The system checks whether any other user has taken the 'booked' number. If the 'booked' number is still available the record is saved using it. If not the next available number is identified, the record saved with this new number and a message displayed so that the user is notified of the change. This method makes sure that in multiuser environments all available numbers will be used and that no numbering gaps are created.

When the user edits the 'booked' value the field value is overridden. The 'custom' number could be greater or less than the one given by the system. When the number is greater than the one given by the system the record is saved with that number and a numbering gap is created. When the number is less than the one given by the system the record is saved without changing the 'booked' field value nor the autonumbering table.

This second method allows to postpone data entry for known quantities. For example Invoices are input by the Head office. Unit B has issued 10 invoices but has not yet send the data. Head office can leave a gap of ten units overriding the next invoice number. When the data from Unit B arrives it can be input using the 10 numbers left empty.



Picture 288 - Autonumber

Autonumber

List of progressive numbers.

This list contains the fields building the progressive number, the reference table, and if required the expression that defines whether the progressive system must be used or not.

Autonumber

Name of the field that will be linked to a progressive table. You can define more fields simply dividing them with commas. In this latter case the last field is the one taking on progressive numbering, whereas the other fields are the 'variations'.

Example

You need to number a specific document and you also need to differentiate in-coming from out-going documents. You need to define two fields:TIPDOC and NUMDOC. The system will create progressive numbers for each TIPDOC value and inputting the progressive number in NUMDOC. Fields receiving progressive numbers can be either numeric or alphabetical. In order to match the alphabetical order with the sequence character types '0' are added to in front of the letter until the field is filled in completely.

Table name

Name of the table containing the progressive value. Progressive values are stored in a dedicated table. This field contains a symbolic name required to identify which numbering must be used. You can create separate numberings, e.g. 'cli' for the customer key, 'art' for the item code) or numberings that can be used by more entities, e.g. different documents such as orders and invoices, using the same numbering.

Condition

Sometimes you may be required to use progressive numbering only under certain conditions. For example you may need to save in-coming and out-going in one entity only. You define the field NUMDOC that must be automatically numbered for out-going documents, but in which you want to manually input numbers for in-coming documents. If out-going documents have the value 'O' in the field TIPDOC you need to define the condition TIPDOC='O' so that progressive numbers are used only for these documents.

Autonumber

You want automatic progressive numbering on invoices.

The field TIPODOC contains the document type ('I' for invoices, 'R' for receipts). The field NUMDOC must contain the progressive number for invoices and the manually inputted numbers for other documents.

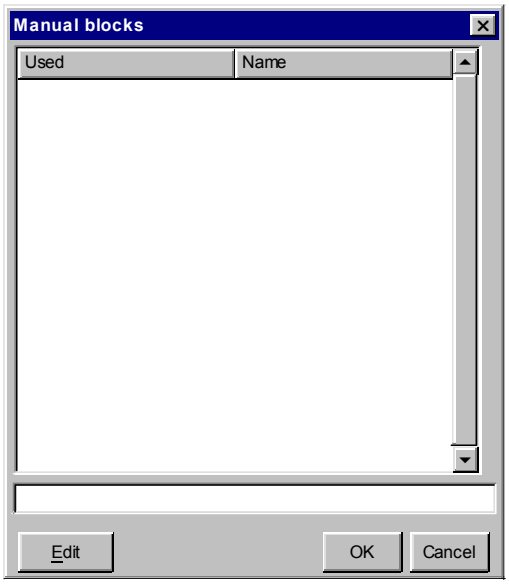
The list will contain:

'w_NUMDOC' in the 'Autonumber' column.

'doc' in the 'Table Name' column.

'TIPDOC'="F" in the 'Condition' column.

6.8.7 Manual Blocks



Picture 289 - Manual Blocks: Page 1

Used - Name (List of manual areas)

List of manual areas contained in the template.

Manual areas containing some code are highlighted in the left column.

Manual Areas

Name of the manual area that must be edited.

You can select manual area either from the list or defining different name in case the desired manual area is not included in the list.

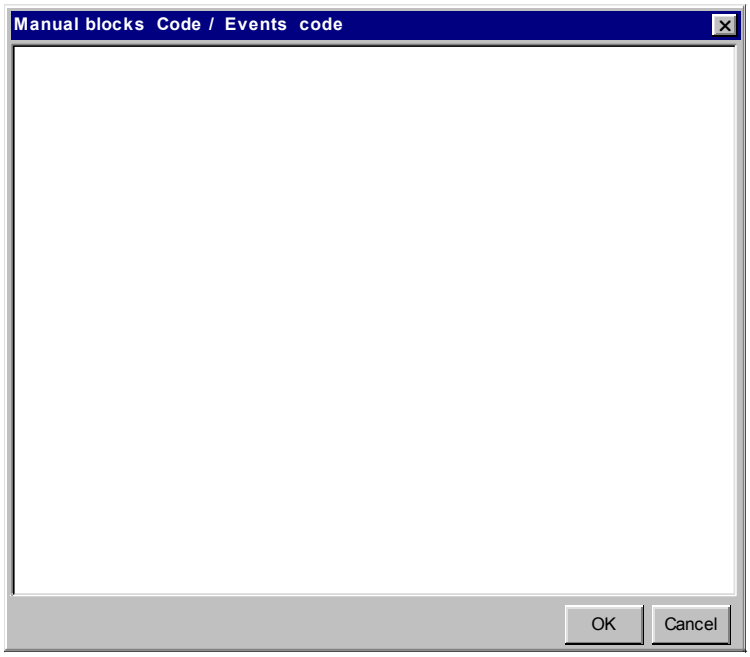
Edit

Access the code defined in the selected manual area.

Double clicking the manual area a dialog window is opened where you can define the code :

6.8.8 Manual Blocks Code

Editing dialog window for manual areas.



Picture 290 - Manual Blocks Code

Code

Source code contained in the manual area.

6.9 Selection Lists

CodePainter has a set of lists based on the project's data dictionary.

These list are generally activated in the Codify tool using the '?' or '...' buttons. These lists make information required for development available.

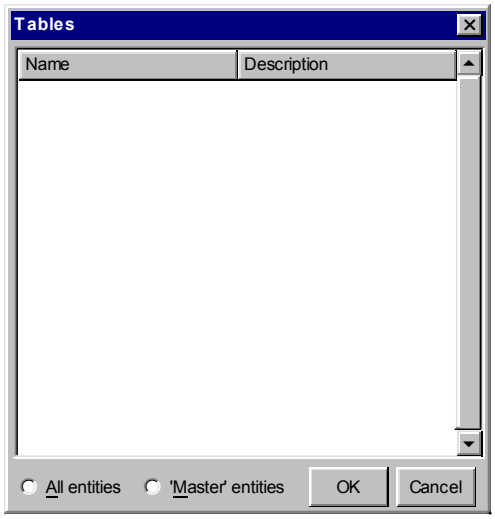
These lists can be sorted double clicking the column title. Selected values are input in the desired window section.

The sequence of selected elements can be moving them up or down in the list. To move elements you need to position the mouse on the first column on the left and wait until the mouse changes into a hand.

This section details the use of selection lists.

6.9.1 Tables

Picture 291 - Tables



Tables List

List of available tables.

Entity Type

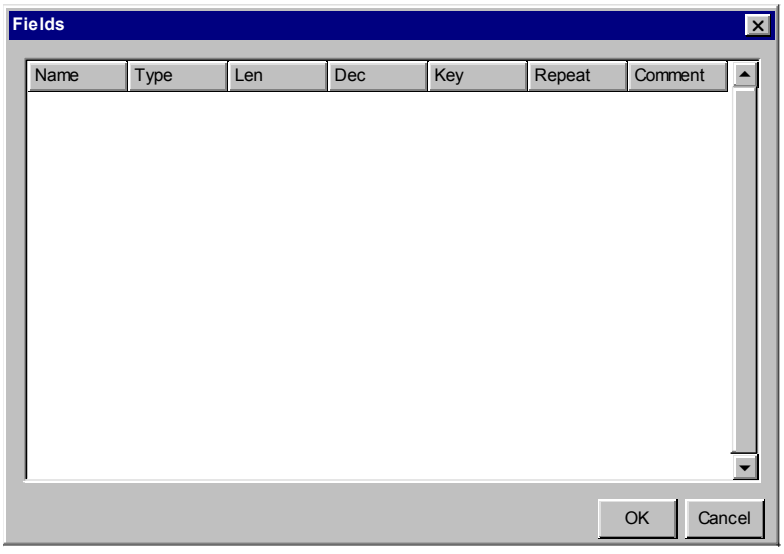
All entities

All tables are displayed.

'Master' Entities

Displays only tables belonging to predefined entity classes.

6.9.2 Fields



Picture 292 - Fields

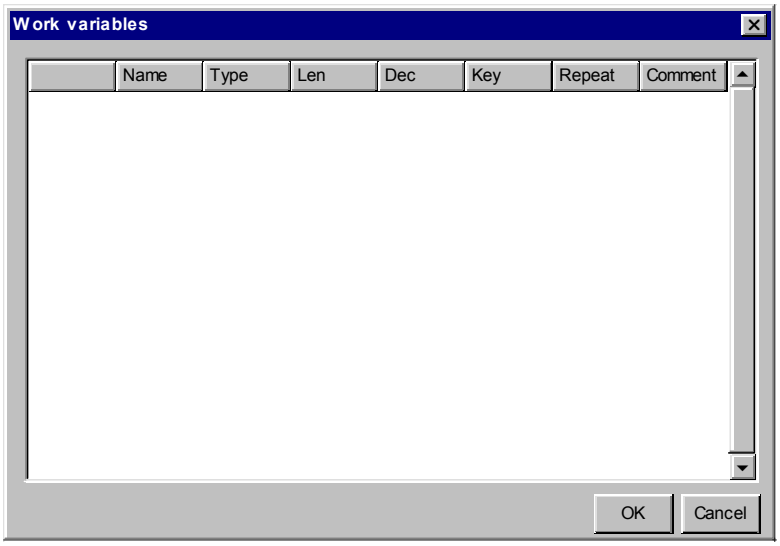
Fields List

List of fields. The list contains all fields defined.

6.9.3 Work Variables

List of variables that have not been implemented so far.

Picture 293 - Work Variables



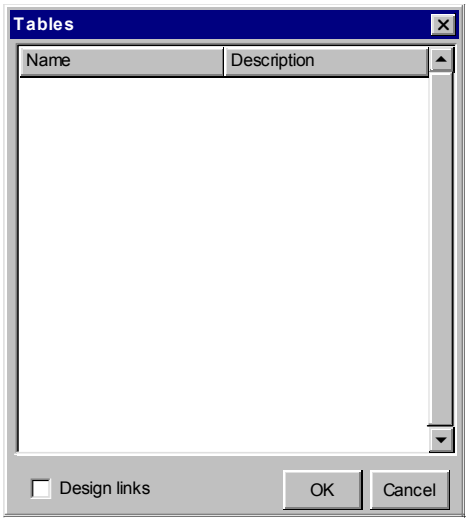
Variables

List of declared variables that have not been implemented so far.

Very often when you define links, variables are defined before they are displayed on the screen. This list analyzes all link definitions and details all available variables. The main advantage is that data is read directly from the data dictionary. Therefore it also reads data type and length, so that the new element can be added to the screen with the correct settings.

6.9.4 Tables

List of files that can be used to define links.



Picture 294 - Tables

Files

List of all tables in the design plan.

Design Links

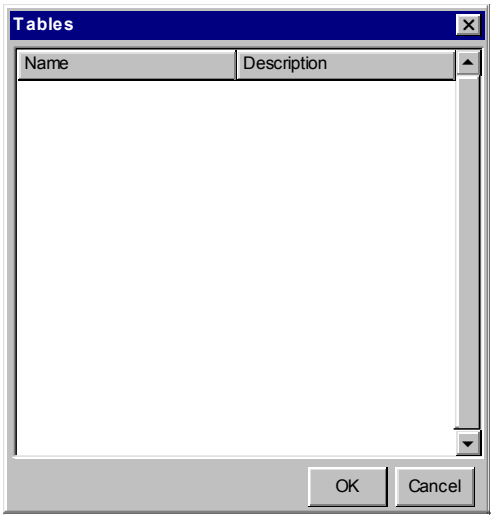
Defines whether to list all tables in the design plan or only those for which a link has been defined during the Design phase.

In the design plan each entity has a set of declared links. During the Codify phase you can create new links for which the source code is generated but not the database referential integrity. It is recommended to use declared links only.

When this flag is set the list above shows declared links. When the flag is not set all tables are displayed.

6.9.5 Tables

Picture 295 - Tables



Tables list

List of available tables.

Entity Type

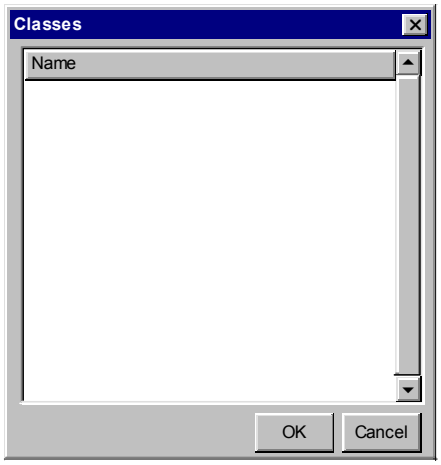
All entities

Displays all entities.

'Master' entities

Displays only entities belonging to the Master class.

6.9.6 Classes



Picture 296 - Classes

Classes List

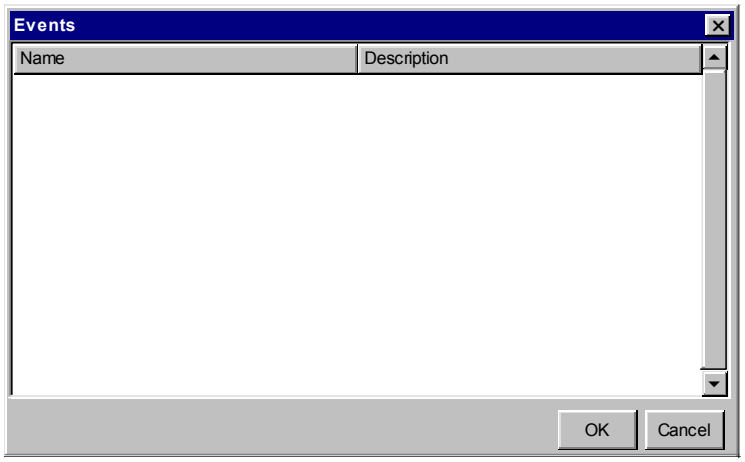
List of available classes.

This list contains the name of object classes defined in CodePainter.

All defined classes are stored in the file CLASSES.CPL under the Painter's Classes directory. To add new classes you simply need to define them in this file.

6.9.7 Events

Picture 297 - Events



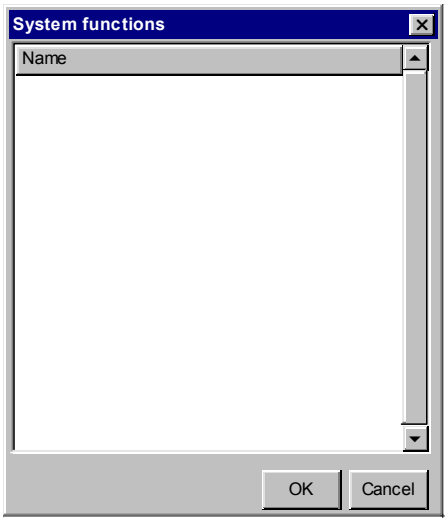
Events list

List of Events.

At run-time CodePainter notifies events to the objects in the dialog windows. Objects will therefore be able to react to given situations, such as data loading, change of an element or saving a record.

This list contains the name and brief description of all events that can be notified.

6.9.8 System Functions



Picture 298 -
System Functions

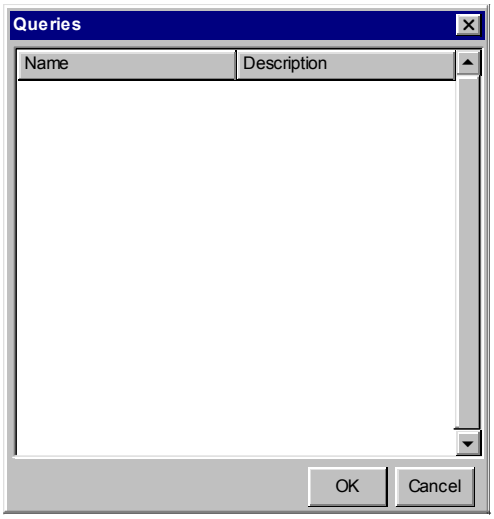
System Functions

List of system functions.

System functions are identified by strings. This list contains all system functions implemented by CodePainter's run-time system.

6.9.9 Queries

Picture 299 - Queries



Name - Description (Tables list)

List of available tables.

All entities - Master entities

All entities

Displays all tables.

'Master' entities

Displays only tables that belong to a to an entity of this class.

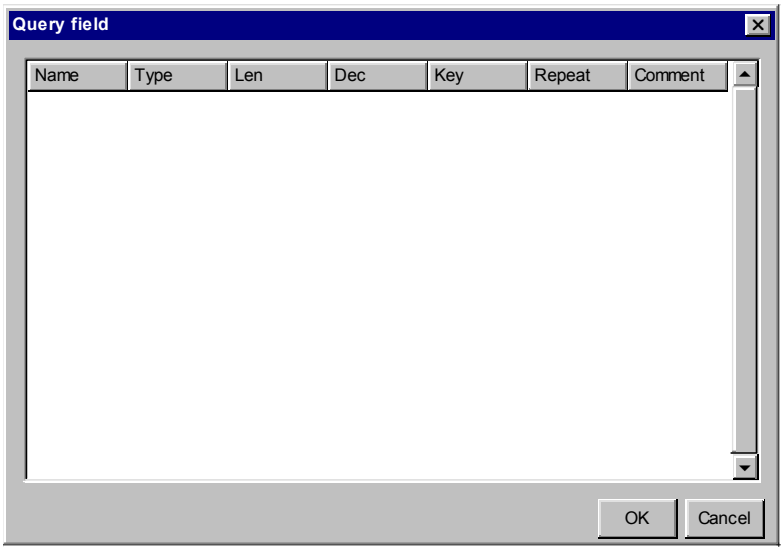
Design links

Defines whether to display all tables or only those associated to a table for which a link has been defined in the design plan.

When the flag is set the list displays only linked tables having referential integrity. When the flag is not set the list displays all tables.

You can create links in the Codify phase without going back to the design plan. The code to manage the link is created but the referential integrity to the database is omitted.

6.9.10 Query Field



Picture 300 - Query Field: Page 1

Fields list

Field list.

This list contains all fields of the selected query.

Fields list

Field list.

This list contains all files of the selected entity.

6.9.11 Design Notes

Picture 301 - Design Notes

Design Notes

Displays the notes added to the elements during the Design phase.

6.10 Product Information

The Help menu option 'About' displays information on the product.

6.10.1 About

The About dialog window contains information on the procedure in use.

The logo CODE**PAINTER REVOLUTION** is displayed and information on the version, series number and build number are given.

The build number helps you identifying CodePainter's distribution, no matter if the product is released totally or partially or through patches.

Picture 302 - About: Page 1

Chapter 7

Dialog Window

7.1 Dialog Window Painter

Dialog Window entities are not linked to a database table. Dialog windows correspond to procedures that manage the user interface.

The Dialog Window Painter has the same functions as the Master Painter, but any reference to databases is omitted. Further you cannot add 'Field' variables or define tables and/or keys.

These entities are used for general data requests, e.g to run queries that read filter parameters or the options of a linked report. Dialog windows are often executed by clicking a button.

The Dialog Window Painter is used to improve the prototype's layout and functionalities, or to create new Dialog Window entities that will be included in the design plan.

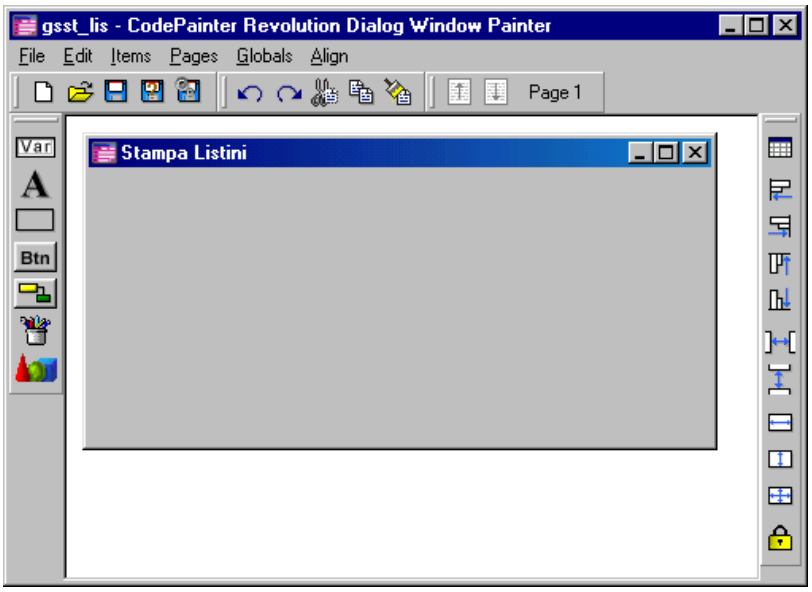
Using the Dialog Window Painter you can improve the window's layout adjusting elements' size and position. You can also add strings, variables, buttons etc. using the WYSIWYG (What You See Is What You Get) methodology. You can further define initialization properties, calculations, validation checks, etc.

When you need to change the Design plan you can maintain the changes made in the Codify phase and generate the prototype only for changed elements in the entity. This can be achieved using comparison functions that will be explained later on.

7.2 The Working Logic

In the Dialog Window Painter you can open prototyped entities. A set of toolbars help you interacting with the prototyped entities. You can either open entities defined in the design plan or create new ones basing on the application's data dictionary. Opened Dialog Window entities have a variable number of rows.

To open defined entities open the 'File' menu and select the 'Open' option. No fields defined at Design level have been prototyped, because Dialog Windows are not linked to tables. All you can see in the Dialog Window Painter is the template and the notes you defined in the design plan.



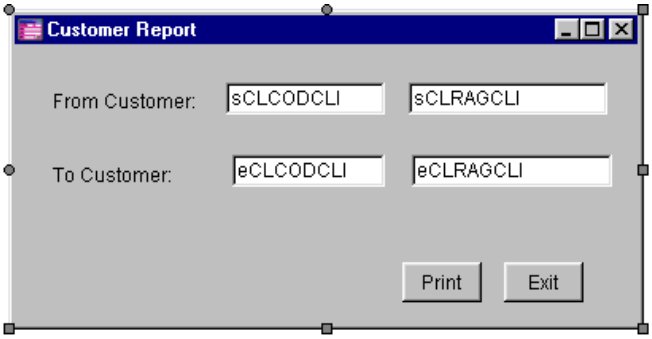
Picture 303 - Dialog Window Painter: New Or Prototyped Window

Using the Dialog Window Painter you can '*design*' the interface and build in validations. Using the 'Code Generation' option you get a working procedure without the need to regenerate the entire design.

In the new or opened entity you can select, move and resize elements as you would do in any other MS Window application.

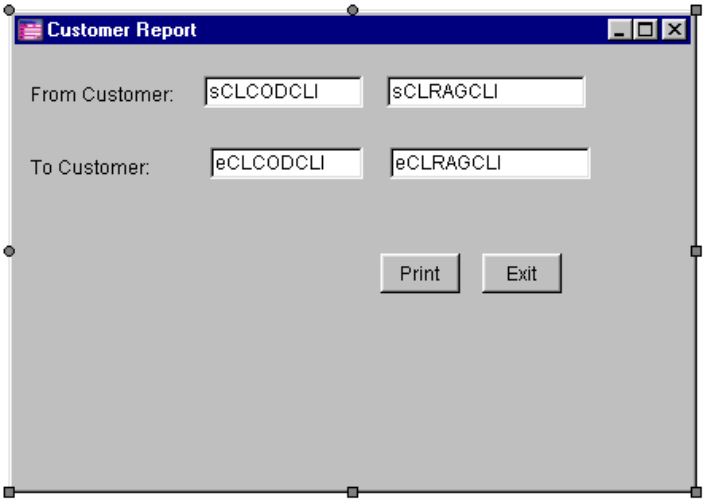
To move the entity window click the title (Caption Bar) and drag it in the desired position. You can notice that while you move the window the mouse changes into a cross.

Picture 304 - Selecting The Window

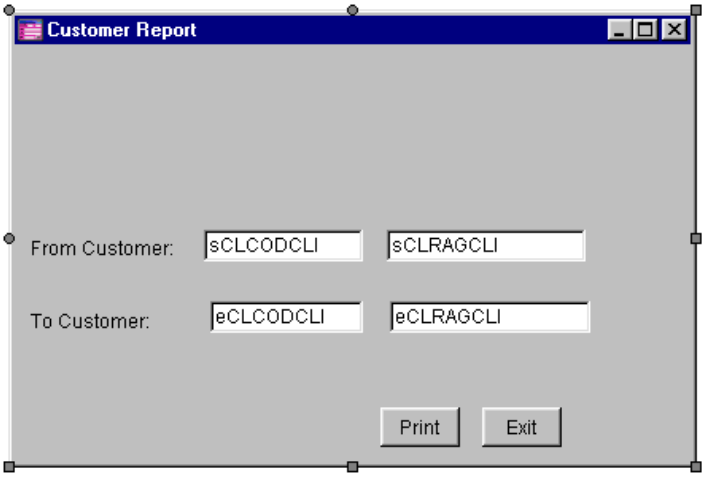


To resize the window you need to select the entity and drag the displayed anchors. The size is always changed according to the direction in which you are dragging when the mouse becomes a 'two ways' arrow. The elements in the window remain fixed compared to the left corner.

Picture 305 - Re-sizing By Dragging



When you right click and drag one of the circled window handles elements remain fixed compared to the bottom right corner. You can notice that while you right click and drag the circle an anchor is displayed under the 'two ways' arrow.



Picture 306 - Re-sizing with Anchors

Similarly you can move and re-size any element in the entity window.

7.3 Selecting And Aligning Elements

Groups of elements can be selected in different ways:

Left clicking the mouse you can drow a rectangle around the elements you wish to select.

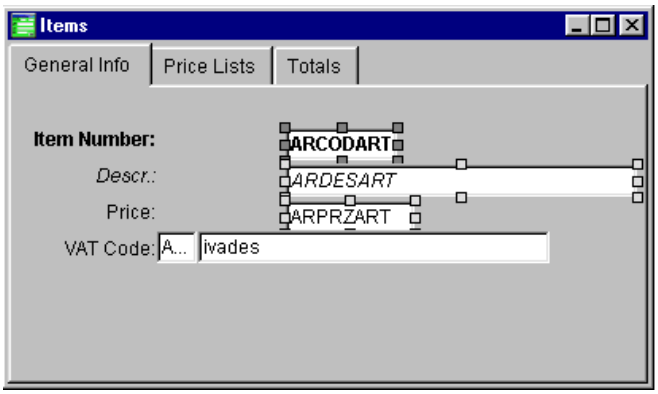
Picture 307 - Selected Group Of Elements

You can also keep the **<Shift>** key pressed and click the elements you wish to select. Single elements are deselected still keeping the **<Shift>** key pressed and clicking again the desired element. All elements are deselected clicking on the working form.

Selected elements are highlighted by handles, little *squares* around the element. Most elements with handles are light grey and one only dark grey. The latter one is the *Master Item* and is used as reference for resizing and aligning the other elements

Warning
When resizing and realigning elements the result changes depending on the Master Item selected.

Aligning the Master Item



Picture 308 - Align to the Left Of The Master Item

Picture 309 - Align to the Left Of The Master Item

The two pictures above show you the different results you will get selecting two different master items. The Master Item can be changed also once the group of elements has been selected simply clicking the desired item again.

You can align and re-size group of elements using both the 'Align' menu or toolbar.

To open the definition dialog window double click the desired element.

Right clicking elements a menu is opened that allows you to edit the element's properties, delete the element or change the editing sequence.

Pressing **<Del>** you can delete one or more selected elements.

Pressing **<Enter>** you can open the definition dialog window of the selected item or in case of multiple selection of the Master Item.

Pressing **<Cursor Keys>** you can move one or more selected elements.

Pressing **<Shift>** and **<Cursor Keys>** you can resize the selected element or in case of multiple selection the Master Item keeping the top right corner fixed.

Pressing **<Ctrl>** and **<Shift>** and **<Cursor Keys>**you can resize the selected element or in case of multiple selection the Master Item keeping the bottom left corner fixed.

Using keyboard keys you can also scroll elements.

Pressing **<Tab>** you can scroll forward.

Pressing **<Shift>** and **<Tab>** you can scroll backwards.

Pressing **<Ctrl>** and **<Tab>** you can scroll a selected group of elements forward.

Pressing **<Ctrl>** and **<Shift>** and **<Tab>** you can scroll a selected group of elements backwards.

To scroll the window elements standard Windows rules apply:

Pressing **<Tab>** you can scroll forward.

Pressing **<Shift>** and **<Tab>** you can scroll backwards.

When you are positioned on tabstrips you can press the **<Cursor Keys>** to scroll the option pages.

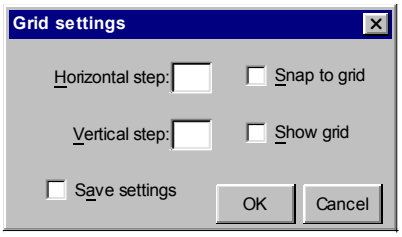
Pressing **<Alt>** and the <UnderlinedLetter> you can edit the corresponding option.

In selection lists you can sort columns. You can also add and delete elements using either the mouse or the **<Ins>** and **<Del>** keys.

When you move elements using the **<Cursor Keys>** changes are madein pixels. When you align elements you can define whether to use a positioning grid or not.

7.3.1 Grid Settings

Design grid definition.



Picture 310 - Grid Settings: Page 1

dx

Horizontal path. The grid has anchors. The distance between anchors is given by the number of pixel defined starting from the left border of the design window.

dy

Vertical path. The grid has anchors. The number of anchors is given by the number of pixel defined starting from the top border of the design window. When the dialog window has more pages the anchor is placed on the top border of the tab-strip.

Snap to grid

Defines whether the elements must be within the grid. When the flag is not selected the elements can be placed anywhere in the window. When the flag is set elements must be in fixed places.

Show grid

Defines whether the grid must be displayed or not. When elements in the dialog window are many and close to each other the grid makes the window unreadable. When this flag is not selected the grid is still active, but is not displayed.

Save settings

Defines whether grid settings must be saved or not. Grid settings can be defined and saved for each dialog window. When this flag is set settings are maintained.

7.4 Dialog Window Painter Toolbars

Let us now analyze the various Dialog Window Painter toolbars. These toolbars help you interacting with Dialog Windows.

7.4.1 Painter Tools Toolbar

The Painter Tools Toolbar allows you to add objects to the entity in use. It has the same functionalities as the Items menu.

figura 11 - Painter Tools Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Adds 'Memory Variable' objects to support validations and calculations/totals.
	Adds descriptive 'String' objects.
	Adds 'Box' objects that allow you to improve the entity's layout adding lines or rectangles that help subdividing the window.
	Adds 'Button' objects These objects to launch queries, process procedures, system functions and/or Dialog windows.
	Adds 'Parent/Child Link' objects that allow you to create buttons that are integrated in the window or that open 'Child' entities.
	Adds Bitmap objects.
	Adds 'Object' objects that allow you to integrate external objects to your project. External objects belong to predefined classes. Examples are graphs, calendars, zooms selecting zooms, etc.

7.4.2 File Toolbar

The 'File' toolbar has a set of button that help you interacting with the tool. This toolbar has the same functionalities as the 'File' menu.

Picture 311 - File Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Creates new entities.
	Opens existing entities.
	Saves the changes made to the current entity.
	Saves the current entity with a different name.
	Saves and generates the source code for the current entity.

7.4.3 Align Toolbar

The *Align* menu allows you to position and resize groups of selected elements.

figura 12 - Align Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Opens the grid management
	Aligns elements to the left in comparison to the Master Item.
	Aligns elements to the right in comparison to the Master Item.
	Aligns elements to the top in comparison to the Master Item.
	Aligns elements to the bottom in comparison to the Master Item.
	Positions elements with the same horizontal distance in comparison to Master Item and the selected elements.
	Positions elements with the same vertical distance in comparison to Master Item and the selected elements.
	Re-sizes selected elements with the same height as the Master Item.
	Re-sizes selected elements with the same width as the Master Item.
	Re-sizes all elements with the same height and width as the Master Item.
	Blocks the possibility to move and re-size elements (the functionalities are not inhibited in the 'Align' menu.

7.4.4 Clipboard Toolbar

The 'Clipboard' toolbar allows you to execute typical Window's commands such as Undo, Redo, Cut, Copy and Paste.

Picture 312 - Clipboard Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).
	The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).
	The 'Copy' option copies one or more selected elements.
	The 'Cut' option deletes selected elements.
	The 'Paste' option pastes copied or cutted elements in the selected position.

7.4.5 Pages Toolbar

Using the 'Pages' toolbar you can browse the entity's pages. The toolbar shows two buttons and the number of the page in which you are positioned.

Picture 313 - Pages Toolbar

Here to follow you will find a brief description of the buttons and their meanings

Button	Meaning
	Moves the edit functionality from the previous page to the current page. This button does not work if you are on the first page.
	Moves the edit functionality from the following page to the current page. This button does not work if you are on the last page.

7.5 Main Menu

7.5.1 File

Picture 314 - File Menu

The 'File' menu has a set of options to:

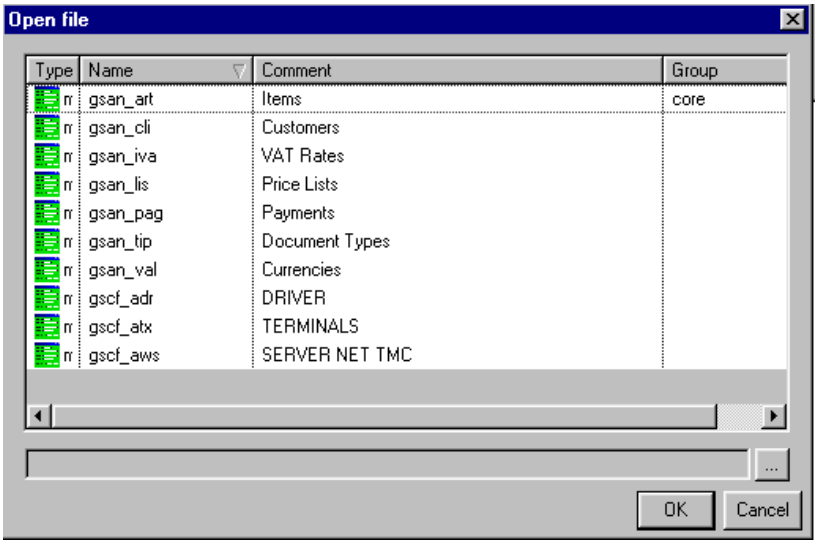
- Create new entities.
- Load existing entities.
- Save changes to the current entity.
- Save the current entity with a different name.
- Save and generate the current entity.
- Read information about CODE**PAINTER REVOLUTION**.
- Exit the tool.

New

The 'New' option closes the current entity asking whether you want to save the changes or not and opens a new entity.

Open...

The 'Open' option loads definition files belonging to the current project.



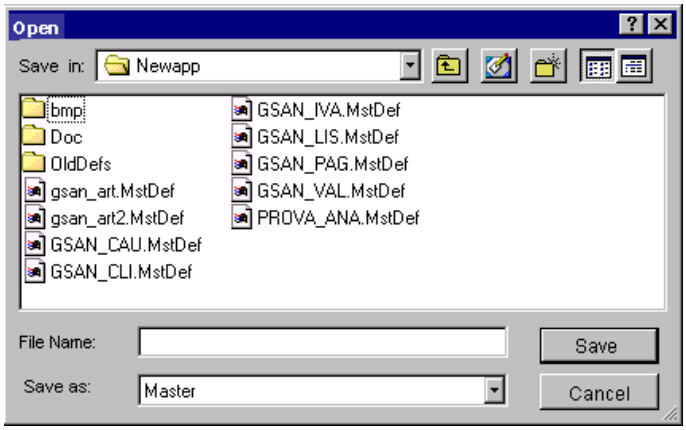
Picture 315 - Open File

The dialog window lists all entities in the project. You can order files by Type, Name, Comment or Group clicking the corresponding column. You can identify the sorting column by the arrow next to the column description.

Picture 316 - Sort Columns

Clicking the '...' button you can browse to search entities of the same type belonging to other project modules.

Picture 317 - Open Dialog Window To Select Design Plans



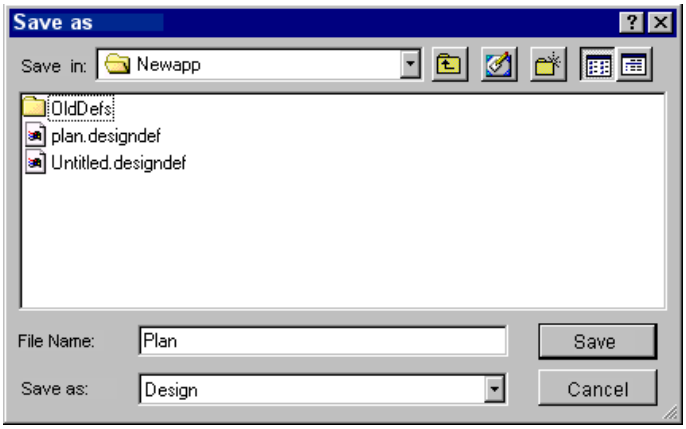
Save

The 'Save' option saves the entity in use.

When you save the entity back-up file (.BAK) is created to store the entity without including the latest changes.

Save As...

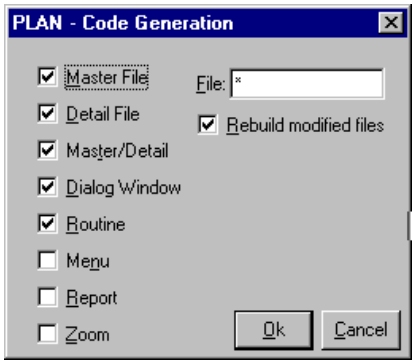
The 'Save As' option saves the entity with a different name.



Picture 318 - Save As Dialog Window

Save and generate

The 'Save And Generate' option saves the current entity and runs the generation. The same is done opening the 'Generation' menu and selecting the 'Design'.



Picture 319 - Code Generation

About

The 'About' option displays information about the product.

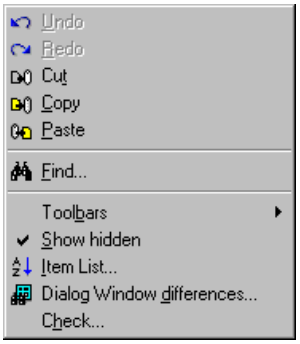
For more information please refer to section 'Product Information'.

Exit

The 'Exit' option makes you exit the tool asking you whether you want to save the changes or not.

7.5.2 Edit Menu

Picture 320 - Edit Menu



The 'Edit' menu has a set of options to:

- Undo/ redo commands.
- Copy, delete and add elements.
- Search, display and replace elements.
- Display and hide toolbars.
- Display and hide elements.
- Display a list of defined elements ('Item List').
- Identify discrepancies with the Design definition.
- Identify 'Codify' errors.

Undo

The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).

Redo

The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).

Copy

The 'Copy' option copies one or more selected elements. More elements can be either selected or grouped in a selection frame you can draw with the mouse.

Cut

The 'Cut' option deletes selected elements from the plan.

Paste

The 'Paste' option pastes copied or cutted elements in the selected position.

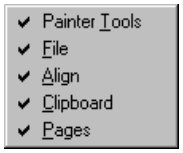
Find...

The 'Find' option opens the Find and Replace dialog window to find and/or replace elements.

Toolbar

The Toolbars option has displays a submenu to enable or disenable toolbars on the Design Painter.

Picture 321 - Toolbar Menu



Show Hidden

The 'Show Hidden' option allows you to show or hide variables for which the 'Editing' option has been set to 'Hide'.

Item List...

The 'Item List' option displays the a list of all elements in the window.

File Painter Differences...

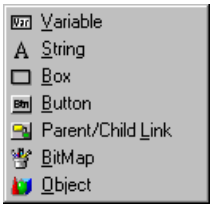
Using this option you can compare the entity with the design plan. For more information please refer to section 'Edit Menu Advanced Options'

Check...

The 'Check' option checks the congruence of expressions and relations defined during the Codify phase. For more information please refer to section 'Edit Menu Advanced Options'.

7.5.3 Items Menu

The 'Items' menu allows adding elements to the entity in use. It has the same functionality as the 'Painter Tools' toolbar.



Picture 322 - Items Menu

Here to follow you will find a brief description of the various elements:

Variable

Using this option you can add Variable objects to your entity. These objects are memory variables used to support validations and calculations/totals.

String

Using this option you can add String objects to your entity. These objects are descriptive strings.

Box

Using this option you can add Box objects to your entity. These objects allow you to improve the entity's layout adding lines or rectangles that help subdividing the window.

Lines are defined '*collapsing*' box objects using **<Shift>** + **<arrows>** keys, or the mouse.

Button

Using this option you can add Button objects to your entity. These objects are typically defined to launch queries, process procedures, system functions and/or Dialog windows.

N.B.
Using Button objects to integrate the entity with system functions ('Help', 'Query', 'Edit', 'Load', 'Save', 'Delete', 'Quit', 'PgUp', 'PgDn', 'ZoomPrev', 'ZoomNext')does inhibit standard toolbar functionalities in the generated application.

Parent/Child Link

Using this option you can add 'Parent/Child Link' objects to your entity. At Design level 'Parent/Child' relationships mean hierarchical links between two entities. Used as objects they allow you to create buttons that are integrated in the window or that open 'Child' entities.

Right clicking these objects you can access to the 'Open Codify...' option and the other standard options ('Edit', 'Delete', 'Sequence'). The 'Open Codify...' option allows you to open the codify tool of the integrated Child entity.

Bitmap

Using this option you can add Bitmaps to your entity.

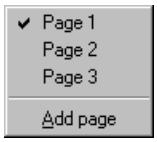
Object

Using this option you can add 'Object' objects to your entity. These objects allow you to integrate external objects to your project. External objects belong to predefined classes. Examples are graphs, calendars, zooms selecting zooms, etc.

For more information on External Object Classes and/or on how to define new classes please refer to the COMPONENT GUIDE manual.

7.5.4 Pages

Using the 'Pages' menu you can browse the entity's pages. This menu has the same functionalities as the 'Pages' toolbar. New pages can be added only using the 'Pages' menu.



Picture 323 - Pages Menu

The menu displays the names of the available pages. The current page has the 'check' symbol next to the name.

Add Page

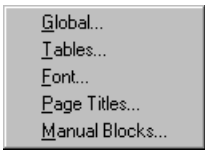
Adds a page to the entity. To delete a page you need to delete all contained elements. When you open the entity the next time, CodePainter identifies that the page is empty and deletes it automatically.

N. B.
In order to be able to access the pages in the generated application, at least one field needs to be editable. If your project requires that all fields in a page cannot be edited, you need to add two editable buttons with the system functions 'Page Up' and 'Page Down'.

7.5.5 Globals

In the 'Globals' menu you can define general information on the Dialog Window entity in use.

Picture 324 - Globals Menu



Globals...

This menu opens the 'Global Definition' window where you can define the generation template, the icon that must be associated to the entity, the author, the user, the developing language, the object that must be launched when the dialog window is saved, etc.

Tables...

In this option you can define the design plan in which the entity is included, the master table used and the work tables that it must open.

Font...

It allows to modify the default character and related characteristics, as length, style and effects for the entity in use.

Pages Title...

This option allows changing default fonts for the entity in use.

Default fonts are defined in the Front End under 'Project/Project Options'.

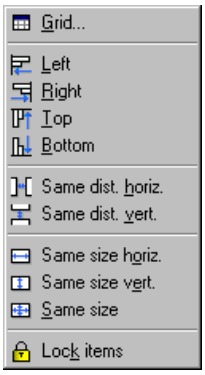
Manual Blocks...

This option displays the list of manual areas defined for the entity in use.

7.5.6 Align

The *Align* menu allows you to position and resize groups of selected elements.

Alignment and re-sizing functionalities are made in relation to a master item. For more information please refer to section 'Selecting And Aligning Elements'.



Picture 325 - Align Menu

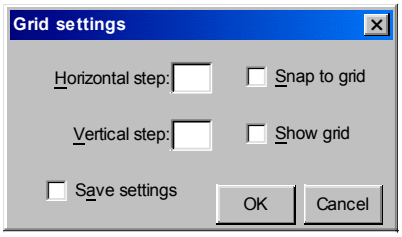
The Align menu allows you to:

- Open the grid management
- Align to the right/left/top/bottom
- Position with the same horizontal/vertical distance.
- Re-size with the same height and width.
- Block changes for manual positioning and resizing.

Grid...

Opens the Grid Setting window to manage the grid:

Picture 326 - Grid Setting



Horizontal Step

Defines the horizontal size of the cell in pixels.

Vertical Step

Defines the vertical size of the cell in pixels.

Snap to grid

Activates the grid to position elements.

Show grid

Displays the grid.

Save setting

Saves the current setting of the grid.

Left

Aligns the left side of selected elements with the left side of the master item.

Right

Aligns the right side of selected elements with the right side of the master item.

Top

Aligns the top of selected elements with the top of the master item.

Bottom

Aligns the bottom of selected elements with the bottom of the master item.

Same dist. horiz.

Selected elements are positioned with the same horizontal distance. The distance is measured between the first and second element starting from the left.

Same dist. vert.

Selected elements are positioned with the same vertical distance. The distance is measured between the first and second element starting from the top.

Same size horiz.

Resizes the width of selected elements like the master item.

Same size vert.

Resizes the height of selected elements like the master item.

Same size

Resizes selected elements like the master item.

Lock items

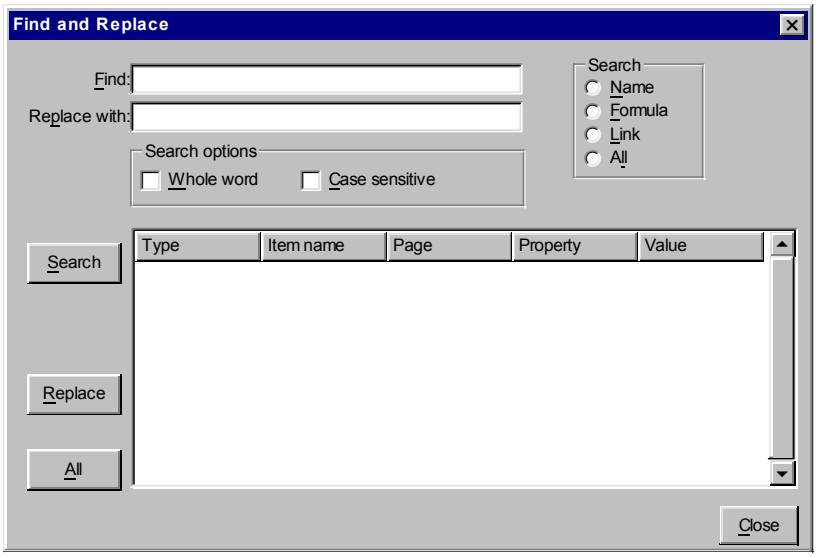
Inhibits selection and change commands for selected items.

Warning
The commands are inhibited for the mouse and the keyboard while the Align toolbar remains active.

7.6 Edit Menu Advanced Options

This section details the Edit Menu options.

7.6.1 Find And Replace



Picture 327 - Find and Replace

Find

String that must be found.

Replace With

String that must replace the found string.

Whole Word

When this field is flagged the search activity must be performed on whole words only. Otherwise search results could be also substrings of long words.

Case Sensitive

The search activity matches upper and lower Case.

Search Button

Starts the search activity:

Name:

the search activity is perfomed on element names

Formula:

the search activity is perfomed on defined formulas.

Link:

the search activity is perfomed on defined links.

All:

the search activity is perfomed on names, formulas and links.

List Of Found Items

Search result listing elements found.

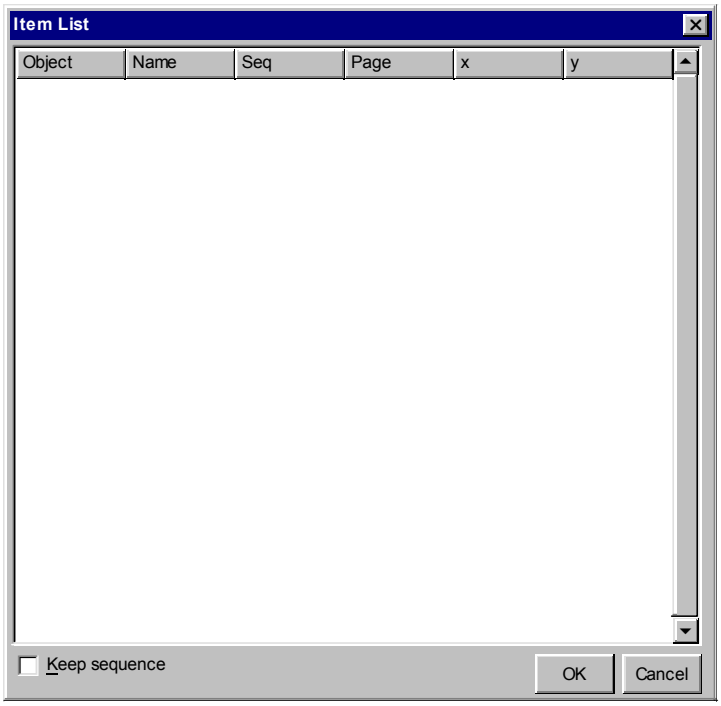
Replace Button

Replaces strings found with the string defined in the textbox 'Replace with' for all elements selected in the list (List of found items).

All Button

Replaces strings found with the string defined in the textbox 'Replace with' for all elements in the list (List of found items) no matter if elements are selected or not.

7.6.2 Item List



Picture 328 - Item List

Elements List

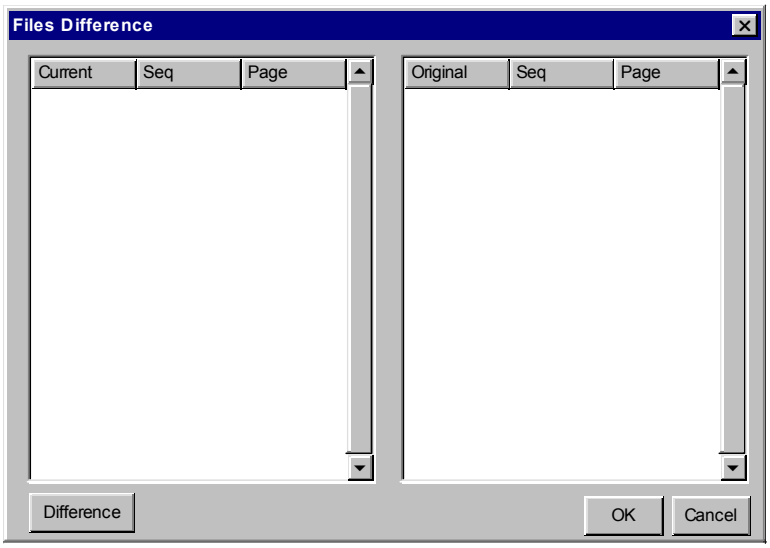
List of elements in the dialog window. This list allows to access elements defined for this entity quickly.

Keep sequence

Flag to maintain the elements' sequence. When the flag is set and the dialog window saved elements are ordered to reflect the sequence defined in the elements' list. To order single elements right click the desired element and select the 'Sequence' option.

7.6.3 Files Difference

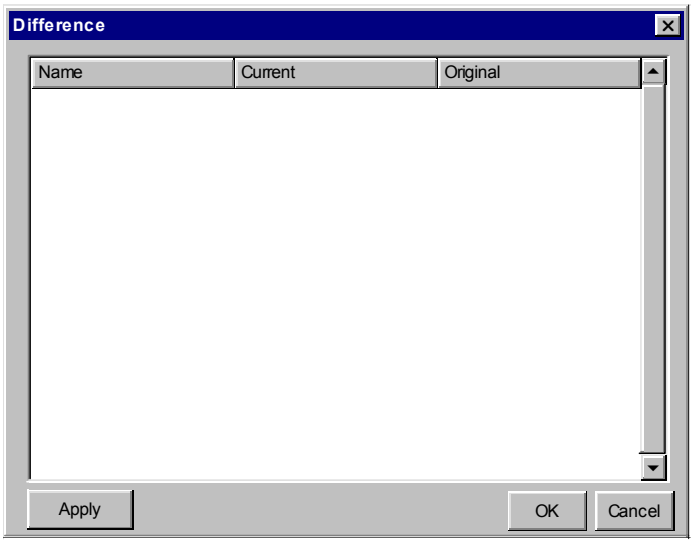
Picture 329 - Files Difference



Difference Button

You can compare two entities using the 'Detail File Differences' option.

7.6.4 Difference



Picture 330 - Difference

Difference List

List of differences between two definition files.

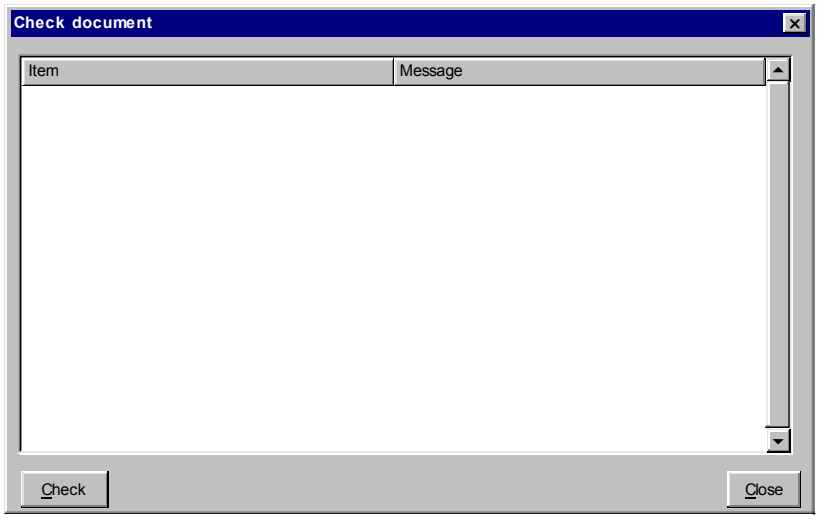
Apply Button

The selected difference is applied to the current definition file.

7.6.5 Check Document

The Check dialog window executes congruence validations on definitions added using the Codify tool. CodePainter allows you to override values that usually are considered wrong, but may be required to manage specific issues. The Check dialog windows checks that no contrasting situations have been defined that could lead to malfunctions at run-time.

Picture 331 - Check Document



Errors List

Error list.

Check

Starts the validation of added definitions.

7.7 Items Menu Advanced Options

The Items menu is accessed opening the Edit - Toolbar menu or right clicking the working area and is used to add new elements to the entity in use. Let's see the various options in detail.

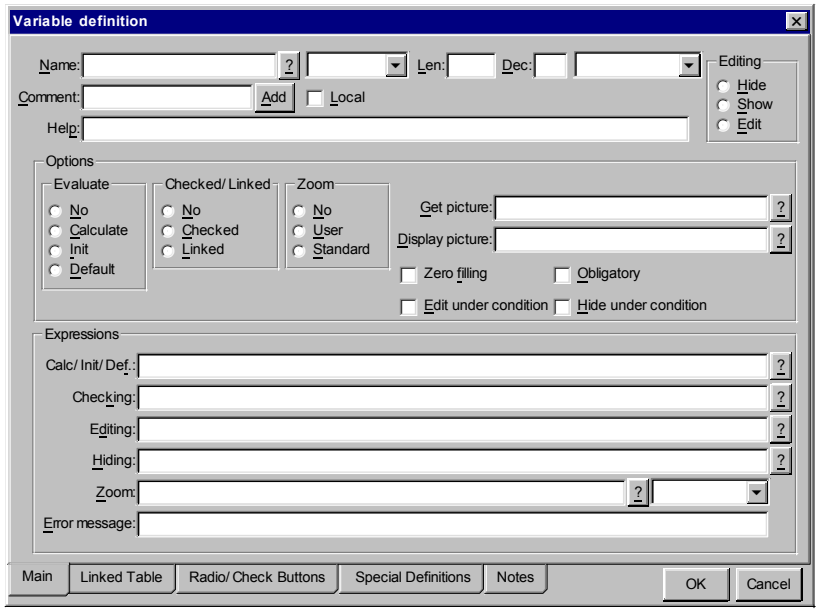
7.7.1 Variable Definition

The definition window allows to define properties for variables. Here you can define how elements are displayed, or define control and calculation formulas, or mandatory fields, or whether to associate a checkbox, radio or combobox, etc.

Main Page

The first page defines the main element's properties. The window has three sections: the first groups the element's characteristics, the second activates controls or calculations and in the third expressions to execute controls are defined.

Picture 332 - Variable Definition: Page 1



pag

Name

Name of the field or variable.

This name must follow the rules for the construction of variable identifiers for the target environment. Typically it must start with a letter and cannot have blanks, commas or points. Some environments also have length limitations.

CodePainter creates for each element (fields and variables) a working variable that takes on the prefix 'w_'. Fields are initializated when the record is read. The user edits the record using the values in the working variables. When the record is saved the contents of 'w_' variables is saved in the corresponding database fields.

Variables that do not have values stored in the database are initializated when the record is loaded. The working variable either does not contain any value (i.e. blank for strings, or 0 for numbers), or it contains the value defined in the 'Init' expression. During the editing phase they behave as fields as defined above.

When the selected element is a field you can list all fields of the associated table clicking the '?' button. In the same way when the element is a variable you can list all variables that CodePainter identifies as useable, but have not yet been placed on the screen.

Field Type

Kind of element.

This combobox defines the kind of element you are editing. Implemented element types are those typically used by business/ commercial applications, i.e. character, numeric, date, logical, memo. The element type here must match with the associated database field. You can define different types only if the type is compatible with what is read by the files: shorter strings, numbers having less digits, etc.

Len

Element's length.

Here you define the length of the variable associated to the element. The length is expressed in characters for strings and in digits for numbers. For logical elements, dates and memos the default length is used.

Dec

Decimal digits for the element.

For numeric elements you can define the number of decimals required. This number is included in the total length.

Key/Index

Defines whether an elements is part of a key or of a search index.

Elements belonging to the primary key can be edited only when a new record is created. They cannot be edited in the change mode.

In the query mode all elements belonging to primary keys or indexes are enabled. When the user enters a value the search activity is quick, because it is executed on the index and not systematically scanning the entire file.

Normally this option is defined for fields and not for values.

Comment

Brief comment on the element.

This comment is displayed in the element list of the edited entity. Fields are automatically initializated using the description defined in the design plan. For variables this must be set by the programmer.

Using the 'Add' button you can add a string next to the element and define the text and comment. This allows you to obtain a prototype for working variables quickly, which is in line with the dialog window.

Add

Using the 'Add' button you can add a string next to the element and define the text and comment. This allows you to obtain a prototype for working variables quickly, which is in line with the dialog window.

Editing

Editing status of the element.

Elements have three editing statuses:

Hide

Hidden elements are not displayed. They are used for calculations or as supporting variables.

Show

Showed elements are displayed on the screen but cannot be edited. They are used in links as return variables, to store calculation results or as support variable that must be checked by the user.

Edit

Editable elements can be changed by the user.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Evaluate

Defines whether the element is calculated.

Elements can be edited by the user or calculated by the procedure.When you define options involving calculations you need to define the expression in the 'Calc/Init/Def.' option in the 'Expressions' area. There are many calculation options:

'No'

When it is set the element is not calculated. Variables are initialized (blank string for characters, 0 for numbers, etc.).

'Calculate':

The element is calculated. The defined expression is checked every time a value affecting the calculation result is entered in the dialog window. When the calculation is simple this strategy works smoothly. When the calculation implies a slow routine you need to limit re-calculations. In the 'Calc/Link depends on' option you need to define which elements trigger the re-calculation. The calculation will be executed only when one of these elements changes. Calculated elements are normally 'shown' and not 'editable'. To make elements editable the calculation must not be re-executed at each input, but only when trigger elements are changed. Otherwise the calculation is re-executed also when a value is entered, thus overriding the input.

'Init':

The calculation is executed only once when the element is edited. This option is used to define a different default value then the one given automatically by the system.

'Default':

In some cases fields cannot be initializated together with the dialog window. This may happen because the field initialization depends upon some values that the user must input first.When the 'Default' option is set the calculation is executed the first time the cursor enters the element. In the initialization expression you can therefore define all elements that the user must enter first.

'Totalize':

When this option is set the elements sums up all values in the body rows. In the calculation expression you need to define the name of the element that must be totalized.

Checked/Linked

Defines whether the element is controlled or linked to a file.

To control the data that the user has input you need to define one of the options in the table below. If you select an option that requires a control you need to define the validation formula in the 'Checking' line in the 'Expressions' area.

'No':

The element is not controlled. Any value is accepted.

'Checked':

Any user input is controlled executing the 'Checking' formula. When the result is TRUE the value is accepted, otherwise an error message is displayed, the element is emptied and the cursor positioned on the element. The standard error message can be changed defining the 'Error message' expression.

'Linked':

The control activity involves a database table. The input value is searched in the database and is accepted only if the linked table contains a record that has the value as primary key. Detail on how this search activity is performed are defined in the 'Linked Table' page. 'Linked' elements can be also controlled by a 'Checking' formula. The value is first searched on the linked table. If found and returned the 'Checking' formula is executed.

Zoom

Defines whether there is an active zoom on the element.

Zooms are procedures run when the user presses the function key F9 when the cursor is on the related element.

'No'

No action is performed.

'User'

The expression defined in the 'Zoom' formula in the 'Expressions' area is executed.

'Standard'

When an element is liked to a table CodePainter creates a standard zoom to

display and search data in the liked table. Parameters defined in the 'Linked Table' page influence the standard zoom. The 'Zoom' expression is used and a specific zoom configuration executed.

Get picture

Format of the input element.

In this option you can define the format that must be applied when the element is inserted.

You need to define an expression that is checked at run-time. Constant values must be defined between apexes.

Typical formats are:

"999.99": for numeric values.

"XXXXX": for characters.

Repl("!", 20): forces upper case in 20 characters strings.

When this option is not defined and you define a display format the input uses the latter. The '?' button creates a standard format for the element type.

Display picture

Format of the displayed element.

In this option you can define a format to be applied when the value is displayed. You need to define an expression that is checked at run-time. Constant values must be defined between apexes.

Typical formats are:

"999.99": for numeric values.

"XXXXX": for characters.

Repl("!", 20): forces upper case in 20 characters strings.

When no input format is defined the format defined for display is used also when data is input.

Zero filling

The value takes on zeros on its left until the value is filled up.

When the 'Autonumber' option is used character keys entries are created as '00001', '00002' etc. The Zero filling option makes it easier for the user who is not required to manually fill the field with zeros. The system automatically adds zeros until the field length is reached.

This option works only is the entered value is numeric and does not start with '0'.

Obligatory

Mandatory element.

An editing phase cannot be terminated as long as no value has been defined for this element. When the user presses the function key F10 and no value has been entered an error message 'Obligatory field' is displayed. The cursor is positioned on the element and the editing mode is kept so that the user can enter a value.

Edit under condition

Defines whether the element can be edited only under specific conditions.

Elements may be editable only when specific conditions occur. To define conditions set the 'Edit' option to edit and define the 'Editing' formula in the 'Expressions' area that must be checked before the cursor is positioned on the element. When the logical expression returns a TRUE value the element is edited. Otherwise the cursor is passed on to the next element.

Hide under condition

Hides elements.

Elements may be hidden under specific conditions. To define these conditions set the 'Edit' or 'Show' option and define the 'Hiding' formula in the 'Expressions' area that must be checked. When the logical expression returns a TRUE value the element is hidden and therefore not displayed on the dialog window.

Calc/Init/Def.

Formula used for calculations.

When calculations formula are defined in the 'Evaluate' option the expression is used to calculate the element's value.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Simple Calculation

Define three elements: PRZART, QTAART and TOTART. The total (TOTART) must be calculated multiplying price and quantity.

The calculation formula must be defined as follows:

```
w_PRZART * w_QTAART
```

Editable Calculation

Consider the same three fields: PRZART, QTAART and TOTART. The total (TOTART) must be calculated and the element must be editable so that the user can adjust roundings manually.

The calculation formula must be defined as follows:

```
w_PRZART * w_QTAART
```

Further, in the 'Calc/Link depends on' option you need to define:

```
w_PRZART
```

```
w_QTAART
```

The calculation is executed only when the value for one of these fields changes.

Checking

Control formula for the element.

When elements are controlled or linked this expression is used to check whether values are acceptable or not. When the logical expression returns a TRUE value the value is accepted. When the returned value is FALSE the value is refused.

When the value is refused the default value is input in the element depending on the element type, an error message is displayed and the cursor passes on to the next element. The default message is 'Value not accepted', but you can change it defining the desired formula in 'Error message'.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Controlled Field

```
The field PRZART accepts only values greater or equal to zero.:
w_PRZART>=0
```

Editing

Formula to activate the conditional editing.

This formula defines whether the field is editable or not. To define conditional editing you need to define the 'Edit' option in the 'Edit' area, set the 'Edit under condition' checkbox, and define the 'Editing' formula in the 'Expressions' area. When the logical expression returns a TRUE value the element is edited. When the returned value is FALSE the fields is disenable.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Hiding

Formula to hide the element.

This formula defines whether the field must be hidden or not. To define these conditions set the 'Edit' or 'Show' option, set the 'Edit under condition' checkbox and define the 'Hiding' formula in the 'Expressions' area. When the logical expression returns a TRUE value the field is hidden. When the returned value is FALSE the field is displayed.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Zoom

Zoom formula.

When elements have an active zoom this formula is used to define what must be executed when the user presses the function key F9.

When the 'Zoom' option is set to 'User' in this formula you need to define what must be executed. The combobox next the option define the type of zoom. Zoom types are:

'User program': the formula is a program line.

'Mask': the formula defines the dialog window that must be executed as zoom.

'Batch': the formula defines a routine created using the Routine Painter.

'UTK Object': the formula contains the output configuration name that must be displayed as zoom.

When the 'Zoom' option defines a 'Standard' zoom the formula can contain the name of a specific configuration that must be loaded in the zoom window.

Error message

Text for the error message.

When the element is controlled the user could enter values that are refused. In these cases the error message 'Value is not correct' is displayed. In this option you can define custom error messages for each element.

Checked/Linked

Defines whether the element is controlled or linked to a file.

To control the data that the user has input you need to define one of the options in the table below. If you select an option that requires a control you need to define the validation formula in the 'Checking' line in the 'Expressions' area.

'No':

The element is not controlled. Any value is accepted.

'Checked':

Any user input is controlled executing the 'Checking' formula. When the result is TRUE the value is accepted, otherwise an error message is displayed, the element is emptied and the cursor positioned on the element. The standard error message can be changed defining the 'Error message' expression.

'Linked'

The control involves a database file. The input value is searched in the database and is accepted only if the linked table contains a record that has the value as primary key. Detail on how this search activity is performed are defined in the 'Linked Table' page. 'Linked' elements can be also controlled by a 'Checking' formula. The value is first searched on the linked table. If found and returned the 'Checking' formula is executed.

Evaluate

Defines whether the element is calculated or not.

Elements can be edited by the user or calculated by the program. When the calculation option is selected you need to define the expression that produces the result in the 'Calc/Init/Def.' option in the 'Expressions' area. There are many calculation options:

'No'

When 'No' is set the element is not calculated. For variables the blank value is initializated (blank string for characters, 0 for numbers, etc.).

'Calculate':

The element is calculated. The defined expression is checked every time a value affecting the calculation result is entered in the dialog window. When the calculation is simple this strategy works smoothly. When the calculation implies a slow routine you need to limit re-calculations. In the 'Calc/Link depends on' option you need to define which elements trigger the re-calculation. The calculation will be executed only when one of these elements changes. Calculated elements are normally 'shown' and not 'editable'. To make elements editable the calculation must not be re-executed at each input, but only when trigger elements are changed. Otherwise the calculation is re-executed also when a value is entered, thus overriding the input.

'Init':

The calculation is executed only once when the element is edited. This option is used to define a different default value then the one given automatically by the system.

'Default':

In some cases fields cannot be initializated together with the dialog window. This may happen because the field initialization depends upon some values that the user must input first.When the 'Default' option is set the calculation is executed the first time the cursor enters the element. In the initialization expression you can therefore define all elements that the user must enter first.

Linked Table Page

In the second page you need to define the rules for the relationship between an element and a file record. The prototype defaults automatically what you defined in the Design phase.

When you define links in the prototype without going back to the design plan the relationship is not programmed in the database. This means that referential integrity for this relationship is not established. Values are checked only when they are entered in the dialog window.

Picture 333 - Variable Definition: Page 2

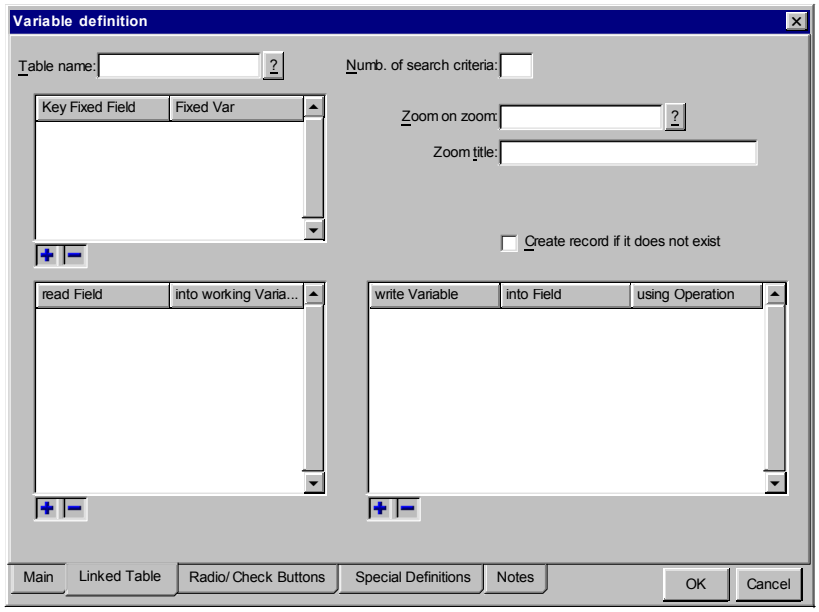


Table name

Name of the related file.

Define the name of the related table. The value is searched in the related table basing on the parameters defined in this dialog window.

The table name is added to the workfiles list so that when the program is run the system checks the existence of the related table. Once you enter the table name, all lists helping the programmer to select fields' names are activated. Moreover, the procedure name that must be activated for the 'zoom on zoom' option is initialized. This procedure is a zoom that can be executed pressing F9 within an active zoom.

The '?' button displays the list of tables available in the database.

Numb. of search criteria

Defines the number of search criteria.

Using this option you can change the way records are searched in the related table. SQL sentences use fields defined in the 'Read Field ... into working Variables' area as search criteria. If the first search goes wrong the SQL sentence uses the second field in the list and so forth as long as there are search criteria.

Define an element 'ARTORD' which is related to the 'Items' file. Your search criteria is CODART and you need to return the field DESART.The standard search criteria is:

```
CODART=w_ARTORD
```

If you define the 'Read Field' list as follows:

```
CODART, w_ARTORD


DESART, e_DA_ORD
```

and the first search criteria does not lead to any result, no error message is displayed and the search activity is executed:

```
DESART=w_ARTORD
```

Only if the second search goes wrong as well the defined value is refused.

The example highlights how the current fields is searched as code first and as description afterwards. If the result is given by the second search criteria values are returned in the defined sequence.

This kind of alternative search is performed for the number of defined criteria, following the list of fields that must be returned. Fields must always be of the same type, you cannot mix e.g. numbers and characters.

Fixed fields belonging to the key are used also used for alternative searches.

Zoom on zoom

Procedure which is executed pressing F9 on an active zoom.

When the user presses F9 a zoom window on the related table is opened. If searched values are not found pressing F9 again (on the active zoom) a new window is opened that allows managing the linked file.

This option allows to define the procedure that must be executed to zoom on an active zoom. Normally the standard procedure to manage the related table is defined. This means that when you select the file reading design definitions, this field is initializated by CodePainter.

Zoom title

Contains the title of the zoom window opened.

Create record if it does not exist

Defines whether a record must be created on the related table.

Linked fields must find a related value in the linked table. This is true for all elements input by the user, because they are validated and not found values are refused.

When linked elements are hidden values are never entered and thus never checked. For example an element may be calculated basing on another element on the screen but linked to another table. It is therefore possible to write totals in a record that does not exits, i.e. the related record is missing.

Using this option the search activity can go on. A new record is created and filled in the related table using the list of fixed key of read fields and totals. Returned values are used to attribute values to fields.

This happens especially when you have Parent/Child relationships whereby totals are required to update the Child entity.

Key Fixed Field

Fields building the primary key, excluding the current field.

This list is used when the primary key of the related table is composite.

The search activity is started when the user enters a value in the current field, which in turn is placed in line with the first element in the list right under the value. Primary key fields are placed next to the corresponding values in the dialog window according to the logic defined in this list. For each field there is value, which has been read by a working variable.

Composite Key

The primary key of the linked table is made of the fields: CODMAG and CODART. In the Dialog Window there are two elements, MAGORD and ARTORD asking the user to input the warehouse code and the item code on which he/she wants to work.

In the ARTORD element you need to define:

```
CODMAG, w_MAGORD
```

In 'read Fields' define:

```
CODART, w_ARTORD
```

The search activity in the related table will have matches of the following kind:

```
MAGART=w_MAGORD and CODART=w_ARTORD
```

The standard zoom uses these fields as filter on the table so that only a selection of the table is displayed and not the entire related table. In the example above after that the user enters a value for the warehouse the zoom window will display only the items in that warehouse.

read Field

List of fields that must be read from the related table.

This list drives the relationship with the related table. When the user enters a value in the current element the search activity is started. The selection criteria is given by the first field in this list and the fixed fields defined in the 'Key Fixed Fields' area (optional).

Example

If the related file has the key CODART and the current element is ARTORD the search activity is as follows:

```
CODART=w_ARTORD
```

If the entered value is found the link is successful and all fields defined in the list are read and values are returned to the corresponding working variables.

If the value is not in the file the error message 'Value is not correct' is displayed and the element is reset.

The search activity can also find partial values, e.g. the user in enters 'Smith' and the value in the file is 'Smith John'. In these cases the value entered in the field is completed with the value found in the table.

When the search activity finds more values (e.g. the user enters 'Smith' and the values in the table are 'Smith John' and 'Smith Bob') a zoom window containing the list of matches found is opened and the user can select the desired record.

When the search activity goes wrong but more search criteria are defined, alternative searches are executed as defined in 'Numb. of search criteria'.

Write Variable

This list defines totalization transactions towards the related table.

This list is made of three columns: the first column defines the variables that will be summed up, the second the fields that will be increased in value and the third the operations that must be performed.

In the third column you can define the constants '+', '-', and '=' to get totals, reversals or a data entry. You can also define fields with one character that contain the transaction that must be executed. When this value is blank no transaction is performed. Transaction types must be stored in a field and not in a variable because CodePainter requires this information to perform reversals in case of changs or deletions.

Example

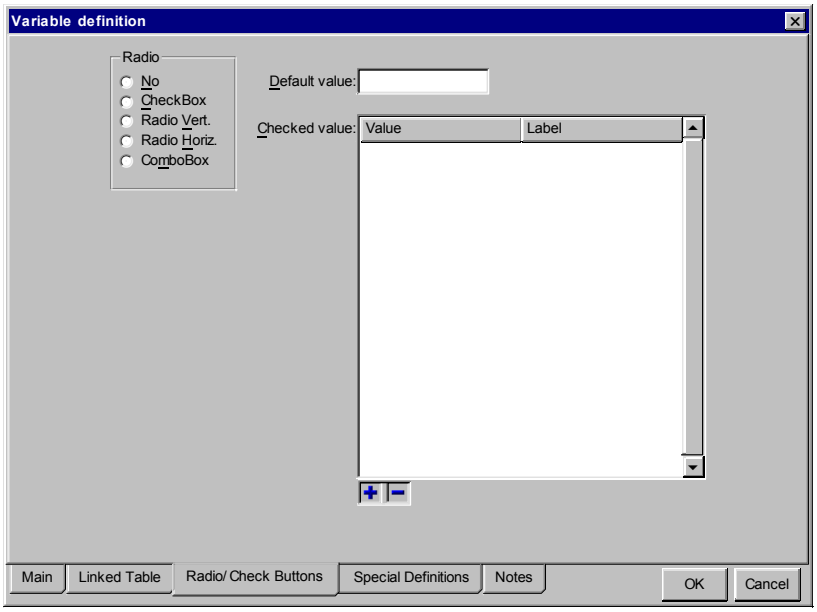
Your application has two tables, namely 'Items' with its fields CODART, DESART and QTAART, and 'Orders' with the fields ARTORD and QTAORD.

You want to sum ordered quantities in QTAART. Define the 'Write Variable' list for the element ARTORD as follows:

```
w_QTAORD, ARTORD, +
```

Radio/Check Buttons Page

The third page contains parameters to change the standard input/ output textbox in other input/ output structures.



Picture 334 - Variable Definition: Page 3

Radio

To display the element as checkbox, radio or combobox.

Fields and variables are created as textboxes. This option allows you to change the method in which data is entered and displayed.

Default value

Defaulted value when the user does not select any value.

Value

List of variables and labels.

This list is made of two columns: the first defines the value which is given to the element, the second the label that must be displayed.

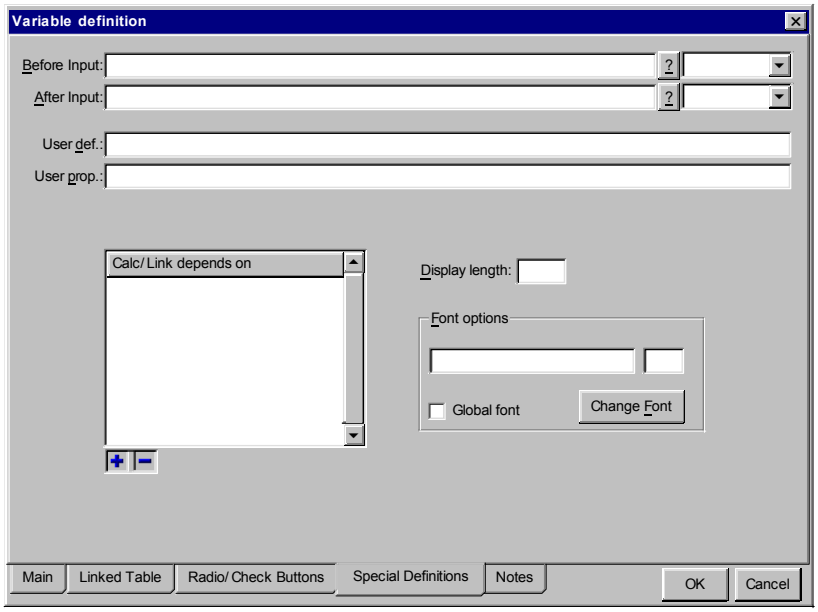
Values must be defined as constants in the target language. Strings must therefore be written between apexes. Labels are always text constants therefore they do not need to be written between apexes.

Special Definitions Page

The fourth page defines properties, which are used seldom.

Rather then using 'Before' and 'After' it is advisable to use the structured options in the first page. 'User Def' require customized templates. 'Depends on' are used for optimizations.

Picture 335 - Variable Definition: Page 4



Before input

Program that must be executed before the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low-level and cannot be defined so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

After input

Program that must be executed after the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low-level and cannot be defines so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

User def.

Free definition.

Could be used by a custom template.

User prop.

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to add definitions to the ones generated by CodePainter.

Example

```
forecolor=RGB(255, 0, 0)
```

Calc/Link depends on

List defining the elements on which the calculation execution or the link executions depends on.

Calculations and links are normally executed every time a value is entered. This strategy works smoothly if the calling routine is quick. If calling routines take long to be terminated you need to limit the number of calls.

Defining working variables in this list, calculations and associated links are executed only when the value for one of these variables changes.

This list is also used for calculated elements, which can be edited. Elements will remain editable, but they will be recalculated only when the value for one of these variables changes. Here you should define all calculation parameters associated to the element.

Using this method you avoid executing many line codes. Nevertheless checking the change of the value is costly and therefore this option should be activated only when calculations are complex.

The same applies to link executions.

Example

A field is linked to the 'Items' file. The field is returned to the linked 'VAT' file. When item's data is entered the VAT value must be defaulted. The field must be editable so that the VAT value can be returned by the first link.

You need to define the 'item number' field in the 'Depends on' list of the 'VAT' field. When the user selects an item the link with the VAT file is re-executed. The VAT value can be freely changed. If the user goes back to the 'item number' field and enters a new value, the link to the VAT file is re-executed.

Display length

Length used to display the element.

This option is used so that CodePainter calculates a length, which is different from the one given by the element parameters.

Change Font

Defines the elements font. If no font is defined the dialog widow's default font is used.

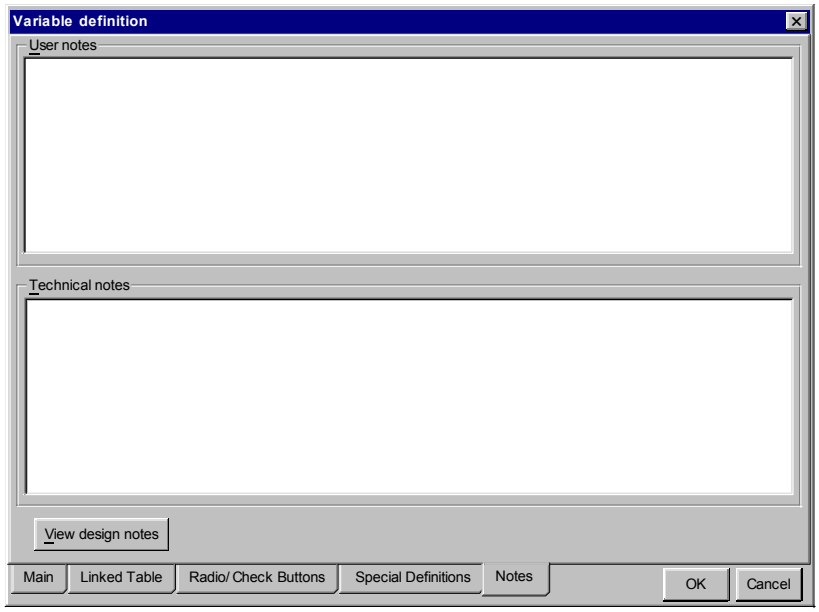
Global font

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.

Picture 336 - Variable Definition: Page 5



User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

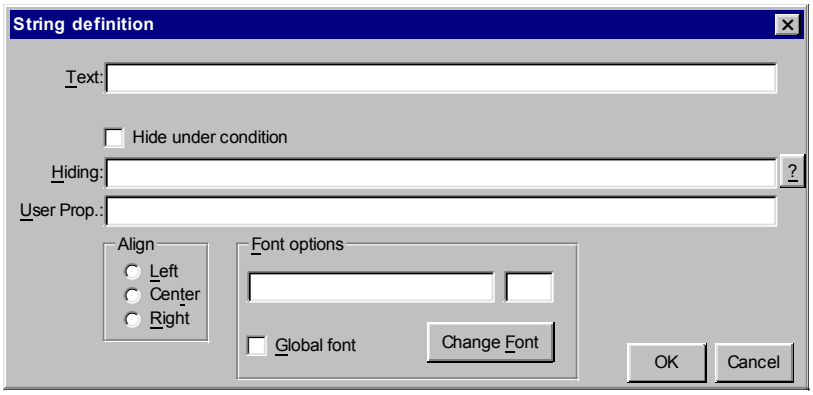
Technical notes on the element.

These notes are used to create technical documentation.

View design notes

7.7.2 String Definition

Dialog window to define the characteristics of comment strings.



Picture 337 - String Definition

Text

String text.

Hide under condition

Hides the string.

To display strings only when specific conditions are met you need to define the hiding formula in the 'Hiding' textbox. When the returned value is TRUE the string is hidden, and is not displayed.

Hiding

Formula defining the conditions to hide the string.

This formula defines whether the string must be hidden or not. To hide the string under given conditions you need to set the checkbox 'Hide under condition' and to define a logical expression in this field. When the result is TRUE the string is hidden, when the result is FALSE the string is displayed.

Clicking the '?' button the list of elements in the dialog window is displayed. These elements can be used to define expressions. Please note that while in the Editing mode CodePainter stores values in working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

User prop

Control properties defined by the programmer.

Elements on the screen are generated as a 'control'. This option allows you to add other specific controls.

Example

```
forecolor=RGB(255,0,0)
```

Align

String alignment.

Defines how the string must be positioned within the string box.

'Left'

Aligned to the left.

'Center'

Centered.

'Right'

Aligned to the right.

To set the window layout you should align strings to the right. You should not align to the left and create boxes that exactly fit the strings. The end-user may use a different font from the one defined and therefore strings may be bigger or smaller. In these cases strings would be no longer aligned.

When you need to translate the application you need to align strings to the right as words in other languages may be longer or shorter. Using CodePainter you can automatically translate dialog windows and set a different language for each user.

Prototypes are created using these principles, i.e. alignment to the right and field space longer than required.

Change Font

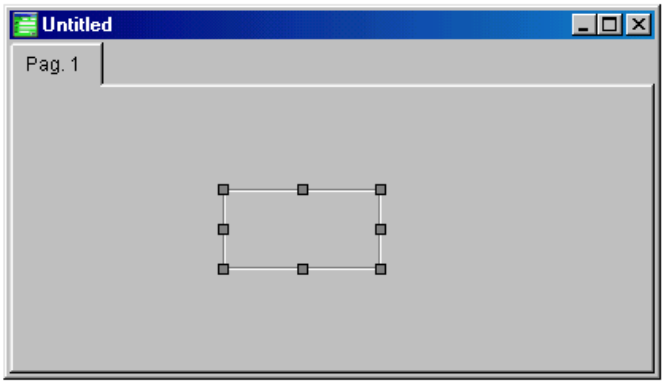
Defines the strings' font. If no font is defined the dialog widow's default font is used.

Global font

When selected the font defined in the 'Global Font' menu is applied to the current element.

7.7.3 Box

This option allows you to add a Box element to the entity in use.



Picture 338 - Box

Boxes are used to graphically divide the form. Boxes can be used as vertical or horizontal lines or rectangles.

Boxes are changed into lines '*collapsing*' them using the mouse or **<Shift>** + **<Cursor Keys>**.

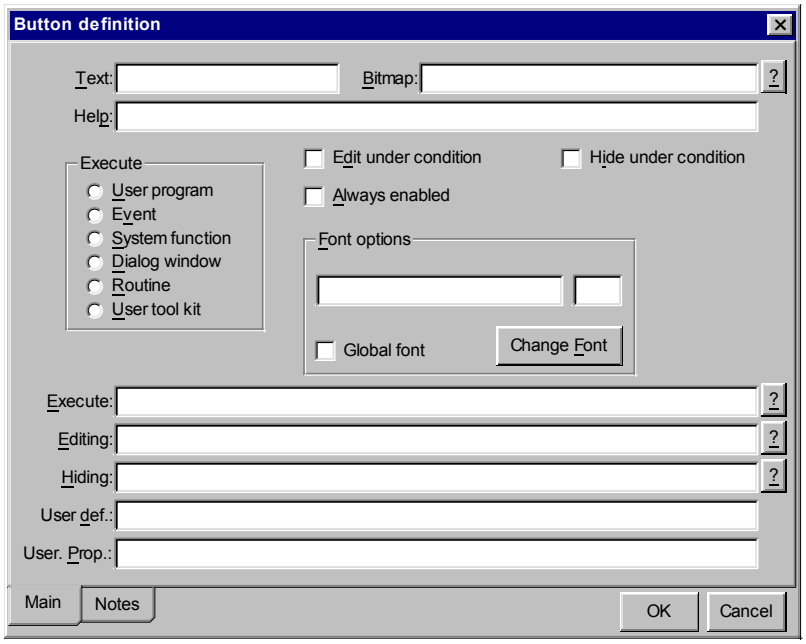
7.7.4 Button Definition

Dialog window defining button options. Buttons are added into dialog windows to execute procedures such as reports, additional dialog windows, etc.

Main Page

In the first page you can define how the button is displayed, activated and its functions.

Picture 339 - Button Definition: Page 1



Text

Text that must be displayed inside the button.

Bitmap

Bitmap displayed inside the button.

Help

Short 'Help' text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Execute

Defines how CodePainter must interpret the execution command line when the button is clicked.

'User program'

Executes a program defined by the developer.

'Event':

Notifies an event.

'System function':

Executes a system function.

'Dialog window':

Opens a dialog window created using the Dialog Window Painter.

'Routine':

Executes a routine developed using the Routine Painter.

'User Tool kit':

Executes an output created using the run-time framework.

Depending on the command type defined in 'Execute' different values are required. 'User Program' is copied in the source. 'Event' requires the name of the event that must be notified. 'System function' requires the name of the function that must be activated. 'User tool kit' requires the name of the query followed by the output format name. In all other cases the name of the file that must be executed is required.

Edit under condition

Defines whether the button is enabled only under specific conditions.

To enable buttons only when specific conditions are met you need to define a formula that must be validated before the cursor is positioned on the element.

Setting this option the 'Editing' formula in the textbox is activated. The formula must be a logical expression. When the returned value is TRUE the button will be enabled, otherwise the cursor is passed on to the next element.

Hide under condition

Hides the button.

To hide buttons under specific conditions you need to define the 'Hiding' formula in the textbox. The formula will determine whether the button must be displayed or hidden. If the logical value TRUE is returned the element is hidden otherwise the button is displayed.

Always enabled

The button is enabled in 'Editing' or 'Query' modes.

Buttons are used to access procedures that are bound to the main dialog window. Using these procedures may make sense in the 'Editing' and/or in the 'Query' mode only. E.g. a button executing a report should always be enabled.

Setting this flag the button will be enabled in the 'Editing' as well ad in the 'Query' mode. When the flag is not set the button is enabled in the 'Editing' mode only.

Execute

Defines what must be executed when the button is pressed.

The meaning of this line depends on what has been defined for the radio button 'Execute'. The button on the left allows quick access to the following definitions:

'User program':

The code line is copied into the source. Using the '?' button you can access the list of variables contained in the dialog window.

'Event':

Defines the name of the Event that must be notified.

'System function':

Executes a CodePainter function. Clicking the '?' button the list of available functions is displayed.

'Dialog window':

Opens a dialog window created using the Dialog Window Painter. The '?' button lists all Dialog Windows available in the project.

'Routine':

Executes a routine developed using the Routine Painter. Beside the routine name you can also define in brackets the parameters that must be passed on to the routine.

'User Tool Kit':

Executes an output created using the run-time framework, typically a query created using the Query Painter. A dialog window is opened in which you can select the kind of output required: preview, report, word, excel or graph. To execute the output you need to add a comma after the query name and digit the output name.

Editing

Formula to set conditions to allow 'Edit' mode.

This formula defines whether the button is active or not. To define conditions you need to set the 'Edit under condition' option and define a logical expression. When the expression result is TRUE the button is enabled. When the result is FALSE the button is disenabled.

Clicking the '?' button the list of all elements in the dialog window is displayed. These elements can be used to define the expression. Please note that during the Editing mode CodePainter stores the values of working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

Hiding

Formula to hide the button.

This formula defines whether the button must be hidden or not. To hide the button you need to set the checkbox 'Hide under condition' and to define a logical expression in the formula. When the result is TRUE the button is hidden, when the result is FALSE the button is displayed.

Clicking the '?' button the list elements in the dialog window is displayed. These elements can be used to define the expression. Please note that while the Editing mode CodePainter stores the values of working variables. These are given by the element's name and the prefix 'w_'. Expressions refer to these variables.

User def

Free definition.

This option can be used by custom templates.

User prop

Control properties defined by the programmer.

Elements on the screen are generated as a 'control'. This option allows you to add other specific controls.

Example

```
forecolor=RGB(255, 0, 0)
```

Change Font

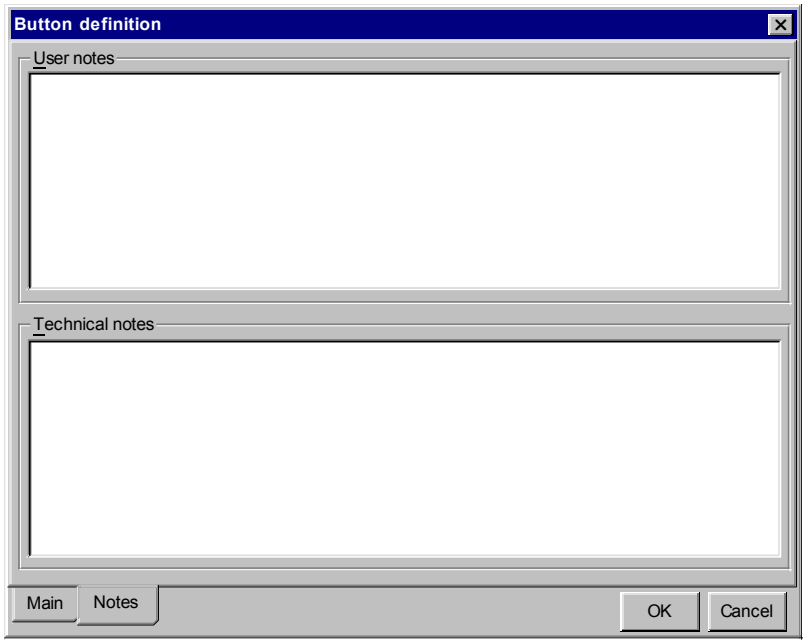
Select the font to be applied to this element. When no font is defined the default font used by the dialog window is applied.

Global font

When selected the font defined in the 'Global Font' menu is applied to the current element.

Notes Page

Notes that are used to create the documentation.



Picture 340 - Button Definition: Page 2

User Notes

Notes on the element.These notes are used to create user documentation.

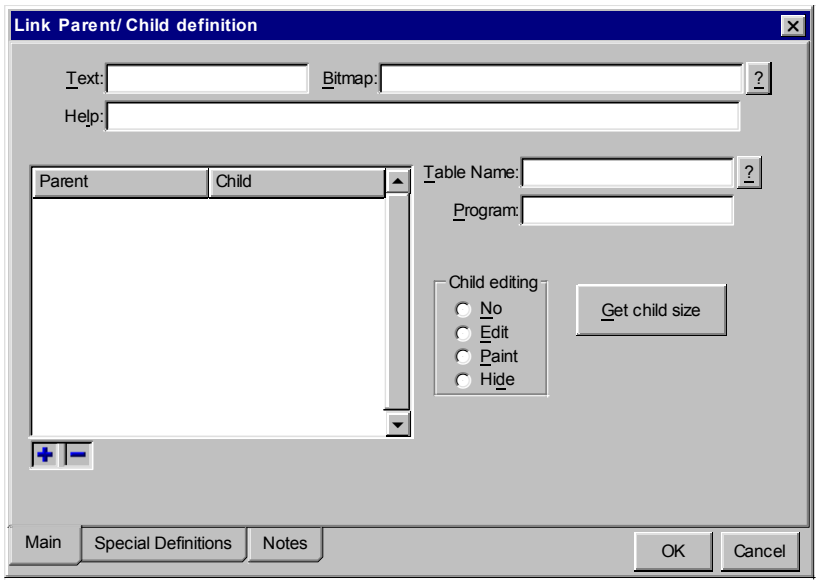
Technical Notes

Technical notes on the element.These notes are used to create technical documentation.

7.7.5 Parent/Child Link Definition

Main Page

Picture 341 - Parent/Children Link Definition: Page 1



Text

Text displayed within the button.

Bitmap

Bitmap displayed within the button.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Parent/Child

List of key fields that bind the Parent to the Child.

This list defines all fields that relate the Parent object to the Child object. CodePainter will pass on these fields when a search activity is required or in Load mode when the Child requires the Parent's key.

Table Name

Name of the Child file.

Program

Name of the Child file's managing procedure.

Child editing

Child's editing mode.

'No'

The Child is activated when the button is clicked.

'Edit'

The Child's managing procedure is integrated in the Parent's dialog window.

'Paint'

The Child is in a different window, but is activated with the Parent window.

'Hide'

The Child is hidden. Data in the Child is canceled when it is canceled in the Parent. Data in the Child cannot be changed.

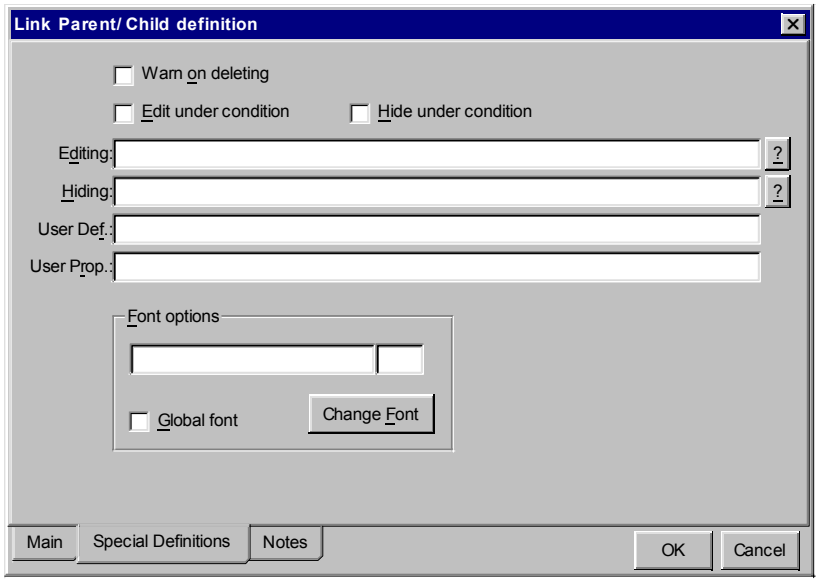
Get child size

Measures the Child's editing window and adjusts the Parent's window so that the Child fits in.

This button is used when the 'Child editing' option is set to 'Edit'.

Special Definitions Page

Picture 342 - Link Parent/Child Definition: Page 2



Warn on deleting

When you are about to delete records of the Parent entity this option warns you that the Child entity data will be deleted as well.

Behind small windows there may be a number of fields hidden in a Parent/Child link button. To avoid deleting this hidden data by mistake you can set this checkbox. Before deleting the system checks the Child's contents. If data is found a message asking you to confirm the delete command is displayed.

Edit under condition

Defines whether the button is enabled only under specific conditions.

Buttons may be enabled only when specific conditions occur. To define conditions set the 'Edit' option to edit and define the 'Editing' formula in the textbox that must be checked before the cursor is positioned on the button. When the logical expression returns a TRUE value the button is enabled. Otherwise the cursor is passed on to the next element.

Hide under condition

Hides buttons

Buttons may be hidden under specific conditions. To define these conditions set the 'Edit' or 'Show' option and define the 'Hiding' formula in the textbox that must be checked. When the logical expression returns a TRUE value the button is hidden and therefore not displayed on the dialog window.

Editing

Formula to activate the conditional enabling.

This formula defines whether the button is enabled or not. To define conditional enabling you need to define the 'Edit' option in the 'Edit' area, set the 'Edit under condition' checkbox, and define the 'Editing' formula in the textbox. When the logical expression returns a TRUE value the button is enabled. When the returned value is FALSE the button is disenable.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

Hiding

Formula to hide the button.

This formula defines whether the button must be hidden or not. To define these conditions set he 'Edit under condition' checkbox and define the 'Hiding' formula in the textbox. When the logical expression returns a TRUE value the button is hidden. When the returned value is FALSE the button is displayed.

Clicking the '?' button all elements in the dialog window that can be used to create the expression are listed.

Please note that during the editing mode CodePainter stores values in working variables, i.e. elements with the prefix ' w_'. Expressions refer to these variables.

User def

Free definition.

Could be used by a custom template.

User prop

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to add definitions to the ones generated by CodePainter.

Example

```
forecolor=RGB(255,0,0)
```

Change Font

Define the elements font. If no font is defined the dialog widow's default font is used.

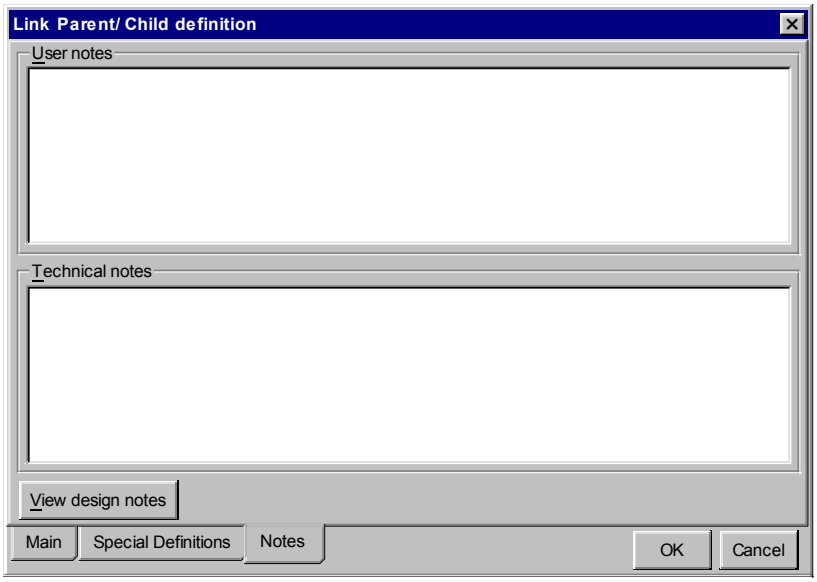
Global font

When selected the font defined in the 'Global Font' menu is applied to the current element.

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.



Picture 343 - Link Parent/Child Definition: Page 3

User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

Technical notes on the element.

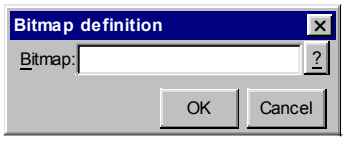
These notes are used to create technical documentation.

View design notes

Quick access to the Design notes of the current element.

7.7.6 Bitmap Definition

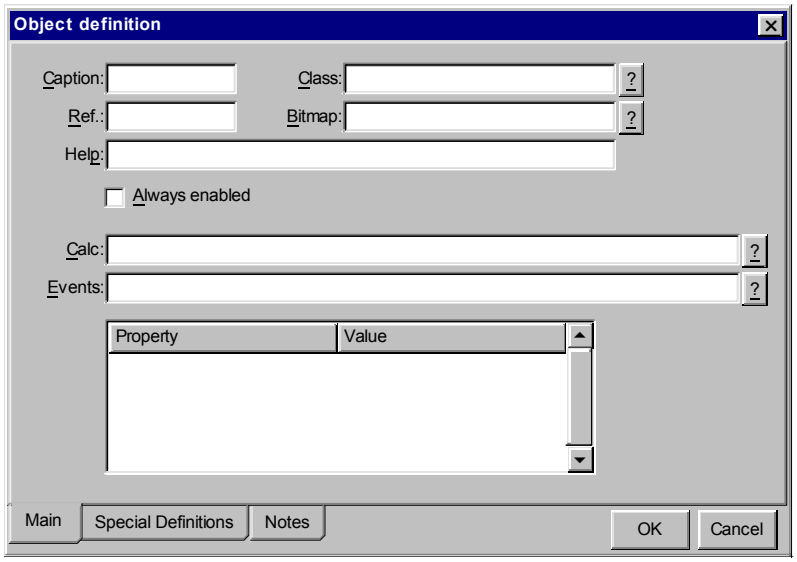
Picture 344 - Bitmap Definition: Page 1



Bitmap

7.7.7 Object definition

Main Page



Picture 345 - Object Definition: Page 1

Caption

Object text.

This text is used as title for those objects that need a string to be displayed.

Class

Object class.

Objects that can be added to dialog windows must belong to predefined classes. CodePainter uses a run-time class library or dedicated templates to generate the relevant source code. The class identifies which tool must be used.

Clicking the '?' button you can access all classes installed in your development environment. For more information on classes and their use, please refer to the COMPONENT GUIDE.

Ref.

Name of the variable associated to the object.

Objects are created as 'controls' in the current form. Associated working variables are not created, because objects are not read by CodePainter's standard procedures.

When you require to access the object from a manual area or a routine, you first need to define the variable name in this property. The working variable created works like the ones associated to fields or variables and is initializated with the pointer to the object.

Bitmap

Bitmap associated to the object.

Help

Short help text.

The text defined in this option is used to associate a tooltip to the element. When the user positions the mouse on the icon and waits a few seconds a small yellow window containing the defined text is displayed.

Always enabled

The object is enabled in 'Editing' and 'Query' mode.

Objects are used to access procedures that are bound to the main dialog window. Using these procedures may make sense in the 'Editing' and/or in the 'Query' mode only. E.g. an object executing a graph should always be enabled.

Setting this flag the object will be enabled in the 'Editing' as well as in the 'Query' mode. When the flag is not set the object is enabled in the 'Editing' mode only.

Calc

Passes on values to the object every time the dialog window is re-calculated.

The input dialog window is recalculated when specific actions are performed: when a new record is input, or when the user inputs specific fields, etc. Each time the dialog window is re-calculated it notifies the object passing on as parameter the value defined in this expression. The object will react according to the class to which it belongs.

For example the 'Dash Board' ('cruscotto') changes the position of the arrow, the 'Calendar' ('calendario') selects the current date, etc.

For more information on how each class exploits recalculations please refer to the COMPONENT GUIDE.

When calculations are complex you can limit re-calculations defining the 'Depends on' property as for any field or variable.

Events

List of events to which the object is hooked.

Objects can communicate with the dialog window in which they are in either through recalculations or events.

Events are notified by the dialog window. The object will react according to the class's specifications to which it belongs. Clicking the '?' button next to properties you access the list of available events. For more information on how each element answers to a given event, please refer to the COMPONENT GUIDE.

Properties

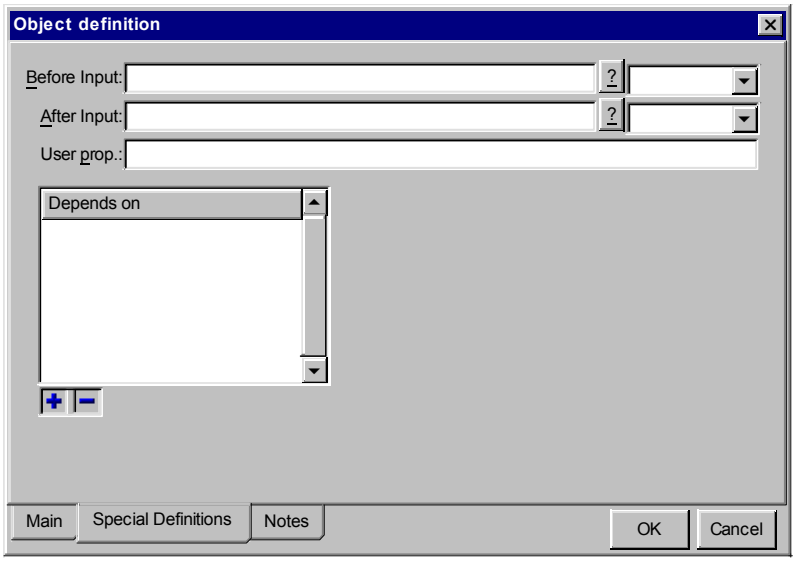
List of object properties.

Objects have a set of properties that differ according to the class to which they belong.

This list displays the property name in the first column and the default value in the second. These values can be changed in order to configure the object. For more information on object properties, please refer to the COMPONENT GUIDE.

Special Definitions Page

Picture 346 - Object Definition: Page 2



Before input

Program that must be executed before the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low level and cannot be defined so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

After input

Program that must be executed after the input.

The combobox next to it tells CodePainter how to interpret what is defined in this option.

This operation is low level and cannot be defines so well as the options in the first page. In particular it cannot distinguish the various status in which the program could be (Edit, Load , Query). This option should be used only if the other options don't allow you to get the required result.

User prop

Control properties defined by the programmer.

Each element in the window is created as 'control'. This option allows to adddefinitions to the ones generated by CodePainter.

Depends on

List defining the elements on which the calculation execution for associated objects depends.

Calculations and links are normally executed every time a value is entered. This strategy works smoothly if the calling routine is quick. If calling routines take long to be terminated you need to limit the number of calls.

When defining working variables in this list, calculations and associated links are executed only when the value for one of these variables changes.

In this list you should define all calculation parameters associated to the element.

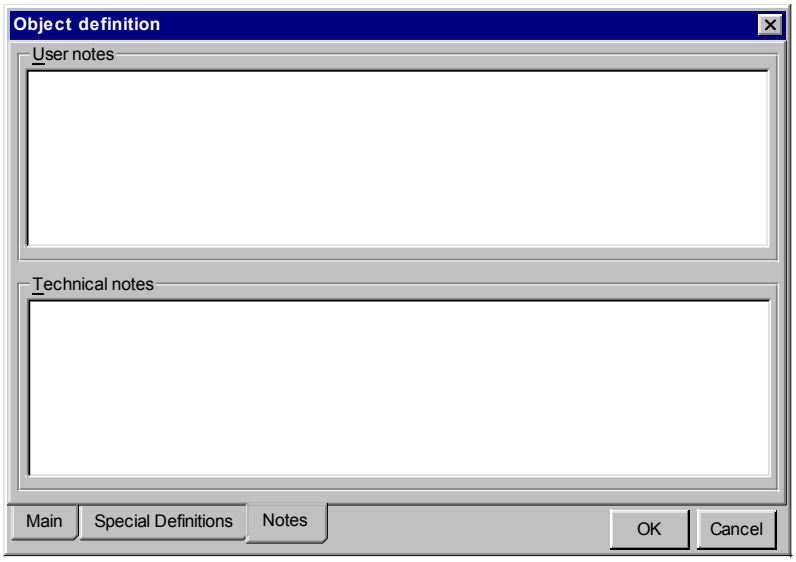
Using this method you avoid executing many code lines. Nevertheless checking the change of the value is costly and therefore this option should be activated only when calculations are complex.

Notes Page

Defines technical and user notes.

Both kind of notes become part of the automatically generated documentation.

Picture 347 - Object Definition: Page 3



User Notes

Notes on the element.

These notes are used to create user documentation

Technical Notes

Technical notes on the element.

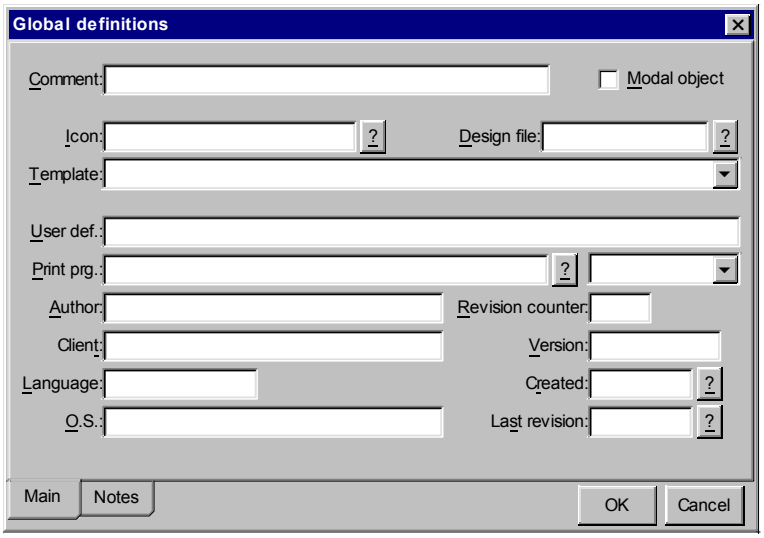
These notes are used to create technical documentation.

7.8 'Globals' Menu Advanced Options

This section details the 'Globals' menu functions.

7.8.1 Global Definitions

Main Page



Picture 348 - Global Definitions: Page 1

Comment

Brief comment on the entity. This comment is displayed in the project file so that you can search entities by file name (i.e. the associated procedure), or by brief description.

Modal object

Defines whether the entity is a modal dialog window or not. Modal dialog windows ask to terminate the input before you can go on to another dialog window. This means that the user cannot use the mouse to go to other windows until the current window is saved (F10) or exit (Esc).

Icon

Icon associated to this entity.

Template

Name of the template used for code generation. In this property you need to define the name of the template used to generate the source code. Templates are procedures' skeletons, finalized by the programmer to generate the required source code. Change the template and the generated source code changes as well.

User def

Free string. Can be used by custom templates.

Print prg.

Program activated during queries pressing F2.

In this property you can define the program that must be called when the user clicks the 'Print' button during queries. The combobox next to the property defines how the command line must be interpreted.

'User program': the command is copied in the source code. 'Mask': a dialog window created with the Dialog Window Painter is opened. 'Batch': a routine created with the Routine Painter is executed. You can also pass on parameters defining them ion brackets. 'UTK Object': defines an output configuration created with user tools such as the Query Painter.

The '?' button next to the property displays a list of possible choices depending on the defined activity.

Author

Author of the program. Short text defining the developer in charge of the entity.

Client

The commissioner. Short text to define the customer who commissioned the entity.

Language

Development language. Short text describing the programming language used to develop this entity.

O.S.

Operating System. Short text defining the limits given to the entity by the operating system.

Revision counter

Revision counter. This read only property shows you how often the entity has been changed. The number is increased automatically every time the entity is saved. This property can be used to check whether copied objects have been changed by the original author.

Version

Entity version number.

Created

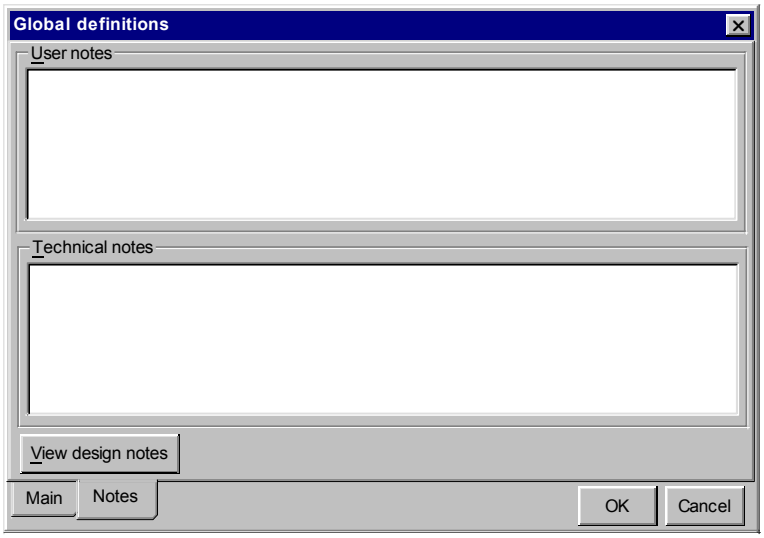
Date in which the entity was created.

Last revision

Date in which the last revision was made.

Notes Page

Picture 349 - Global Definitions: Page 2



User Note

Notes on the element. These notes are used to create user documentation.

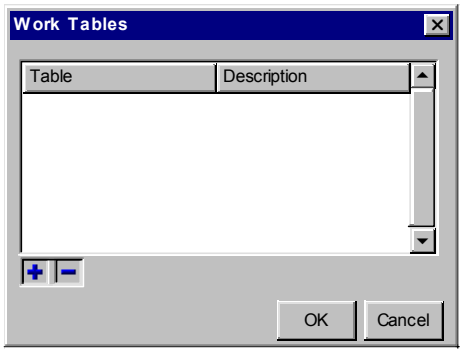
Technical Notes

Technical notes on the element. These notes are used to create technical documentation.

View design notes

Quick access to the Design notes of the current element.

7.8.2 Tables



Picture 350 - Tables

Tables

List of tables.

The generated procedure controls that the database contains all tables displayed in this list. When one or more tables are missing an error message is displayed and the procedure is not executed. This is done to avoid errors.

The procedure can also use tables which are not included in the list. Programs should not be executed if they can generate errors.

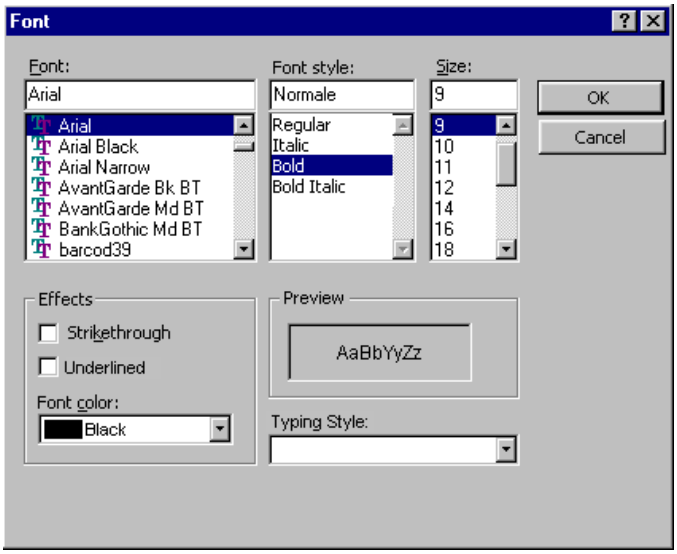
Tables used in links are automatically added to the list.

7.8.3 Font

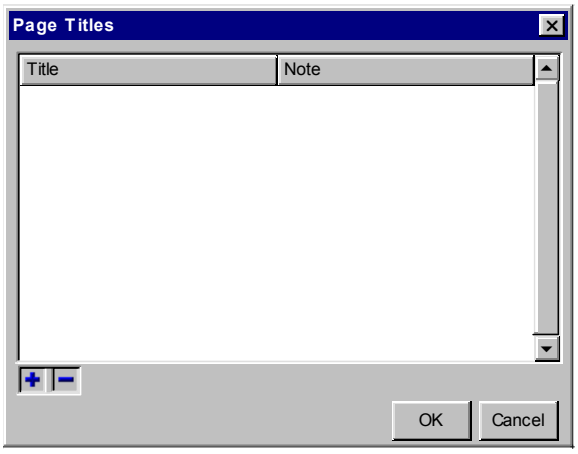
Allows changing default fonts for the entity in use.

Default fonts in this area are the ones you defined in CodePainter's Front End in the 'Project' menu in the 'Project Options' option.

Picture 351 - Font Definition



7.8.4 Page Titles



Picture 352 - Page Titles: Page 1

Titles

Page titles. Entities can have dialog windows having more pages. The default title is 'Pag.' followed by the page number. This list allows changing pages' title.

7.8.5 Autonumber

Manages progressive numbering automatically.

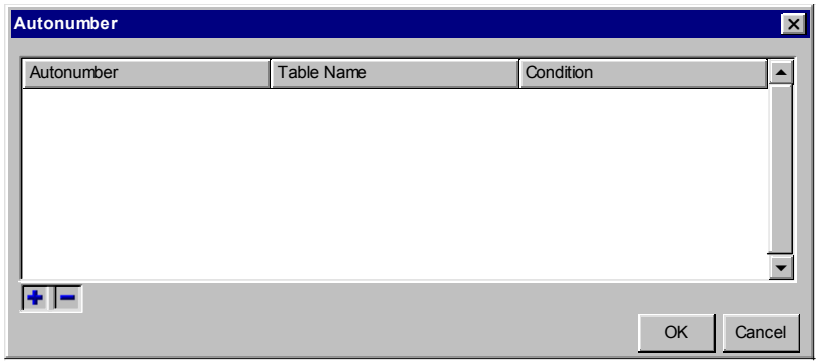
You can define fields that are numbered automatically in each entity. Once the starting number is defined the field is increased every time a new record is input.

When the 'autonumber' option for a field is defined, the field is initialized with the next value in the numbering sequence when a new record is input. If the user does not change the value and the record is saved the table containing progressive numbering is read. The system checks whether any other user has taken the 'booked' number. If the 'booked' number is still available the record is saved using it. If not the next available number is identified, the record saved with this new number and a message displayed so that the user is notified of the change. This method makes sure that in multiuser environments all available numbers will be used and that no numbering gaps are created.

When the user edits the 'booked' value the field value is overridden. The 'custom' number could be greater or less than the one given by the system. When the number is greater than the one given by the system the record is saved with that number and a numbering gap is created. When the number is less than the one given by the system the record is saved without changing the 'booked' field value nor the autonumbering table.

This second method allows to postpone data entry for known quantities. For example Invoices are input by the Head office. Unit B has issued 10 invoices but has not yet send the data. Head office can leave a gap of ten units overriding the next invoice number. When the data from Unit B arrives it can be input using the 10 numbers left empty.

Picture 353 - Autonumber



Autonumber

List of progressive numbers.

This list contains the fields building the progressive number, the reference table, and if required the expression that defines whether the progressive system must be used or not.

Autonumber

Name of the field that will be linked to a progressive table. You can define more fields simply dividing them with commas. In this latter case the last field is the one taking on progressive numbering, whereas the other fields are the 'variations'.

Example

You need to number a specific document and you also need to differentiate in-coming from out-going documents. You need to define two fields:TIPDOC and NUMDOC. The system will create progressive numbers for each TIPDOC value and inputting the progressive number in NUMDOC. Fields receiving progressive numbers can be either numeric or alphabetical. In order to match the alphabetical order with the sequence character types '0' are added to in front of the letter until the field is filled in completely.

Table name

Name of the table containing the progressive value. Progressive values are stored in a dedicated table. This field contains a symbolic name required to identify which numbering must be used. You can create separate numberings, e.g. 'cli' for the customer key, 'art' for the item code) or numberings that can be used by more entities, e.g. different documents such as orders and invoices, using the same numbering.

Condition

Sometimes you may be required to use progressive numbering only under certain conditions. For example you may need to save in-coming and out-going in one entity only. You define the field NUMDOC that must be automatically numbered for out-going documents, but in which you want to manually input numbers for in-coming documents. If out-going documents have the value 'O' in the field TIPDOC you need to define the condition TIPDOC='O' so that progressive numbers are used only for these documents.

Autonumber

You want automatic progressive numbering on invoices.

The field TIPODOC contains the document type ('I' for invoices, 'R' for receipts). The field NUMDOC must contain the progressive number for invoices and the manually inputted numbers for other documents.

The list will contain:

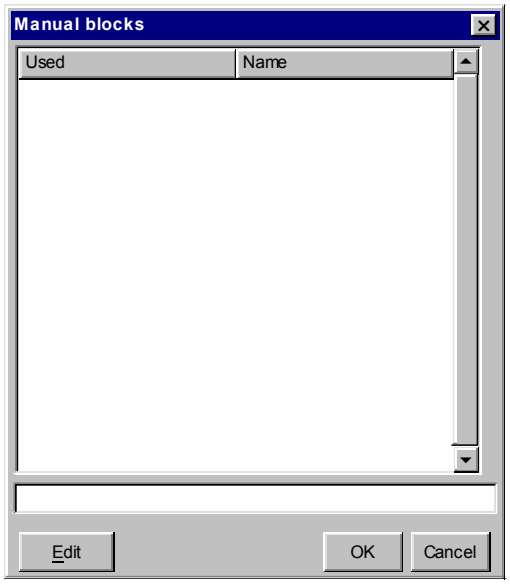
'w_NUMDOC' in the 'Autonumber' column.

'doc' in the 'Table Name' column.

'TIPDOC'="F" in the 'Condition' column.

7.8.6 Manual Blocks

Picture 354 - Manual Blocks: Page 1



Used - Name (List of manual areas)

List of manual areas contained in the template.

Manual areas containing some code are highlighted in the left column.

Manual Areas

Name of the manual area that must be edited.

You can select manual area either from the list or defining different name in case the desired manual area is not included in the list.

Edit

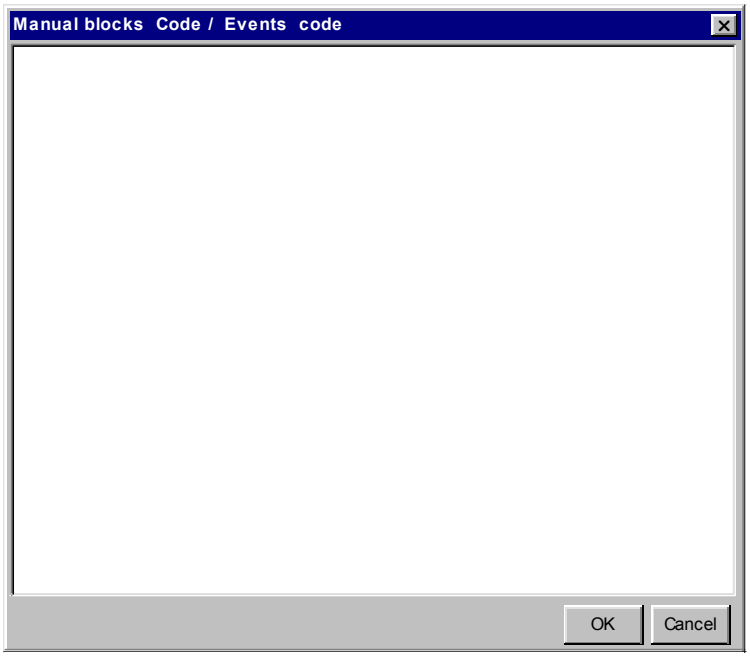
Access the code defined in the selected manual area.

Double clicking the manual area a dialog window is opened where you can define the code :

7.8.7 Manual Blocks Code

Editing dialog window for manual areas.

Picture 355 - Manual Blocks Code



Code

Source code contained in the manual area.

7.9 Selection List

CodePainter has a set of lists based on the project's data dictionary.

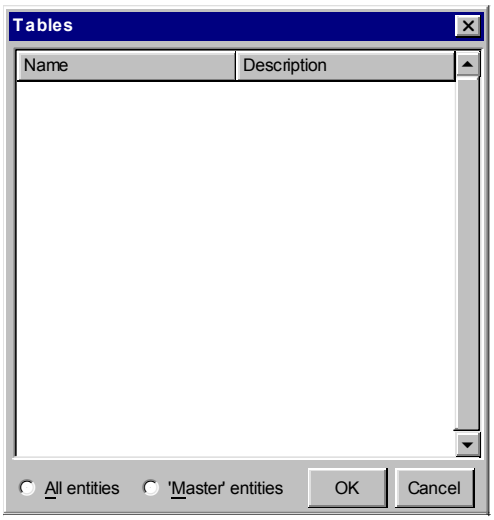
These list are generally activated in the Codify tool using the '?' or '...' buttons. These lists make information required for development available.

These lists can be sorted double clicking the column title. Selected values are input in the desired window section.

The sequence of selected elements can be moving them up or down in the list. To move elements you need to position the mouse on the first column on the left and wait until the mouse changes into a hand.

This section details the use of selection lists.

7.9.1 Tables



Picture 356 - Tables

Tables List

List of available tables.

Entity Type

All entities

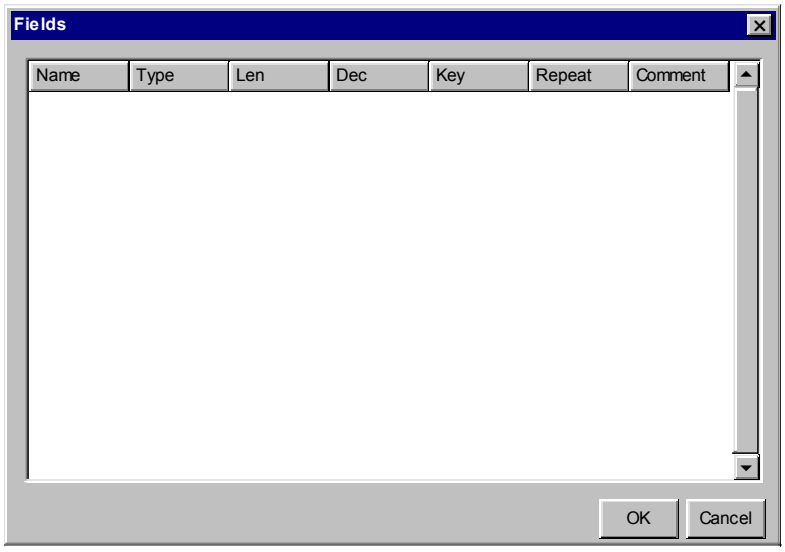
All tables are displayed.

'Master' Entities

Displays only tables belonging to predefined entity classes.

7.9.2 Fields

Picture 357 - Fields

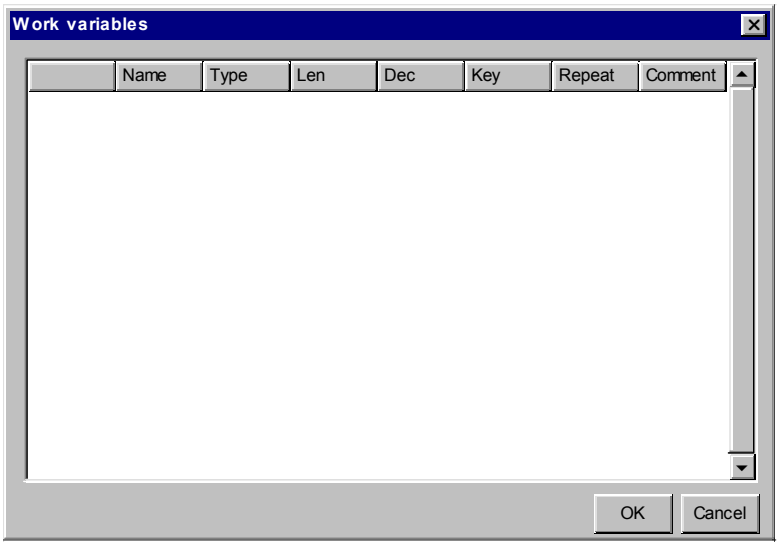


Fields List

List of fields. The list contains all fields defined.

7.9.3 Work Variables

List of variables that have not been implemented so far.



Picture 358 - Work Variables

Variables

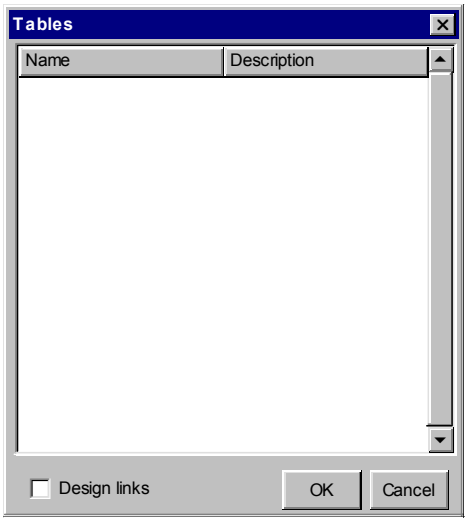
List of declared variables that have not been implemented so far.

Very often when you define links, variables are defined before they are displayed on the screen. This list analyzes all link definitions and details all available variables. The main advantage is that data is read directly from the data dictionary. Therefore it also reads data type and length, so that the new element can be added to the screen with the correct settings.

7.9.4 Tables

List of files that can be used to define links.

Picture 359 - Tables



Files

List of all tables in the design plan.

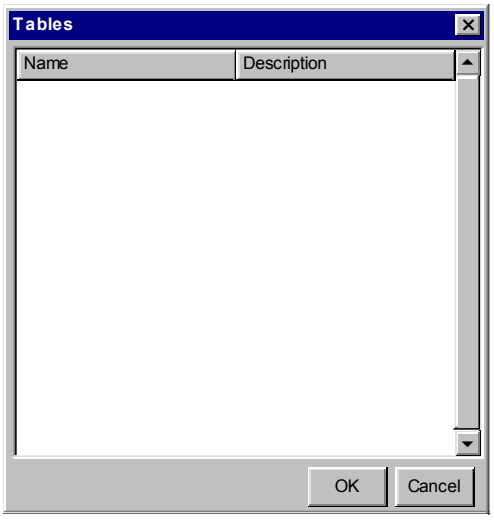
Design Links

Defines whether to list all tables in the design plan or only those for which a link has been defined during the Design phase.

In the design plan each entity has a set of declared links. During the Codify phase you can create new links for which the source code is generated but not the database referential integrity. It is recommended to use declared links only.

When this flag is set the list above shows declared links. When the flag is not set all tables are displayed.

7.9.5 Tables



Picture 360 - Tables

Tables list

List of available tables.

Entity Type

All entities

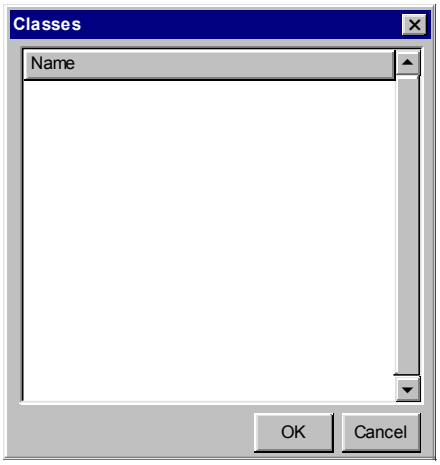
Displays all entities.

'Master' entities

Displays only entities belonging to the Master class.

7.9.6 Classes

Picture 361 - Classes



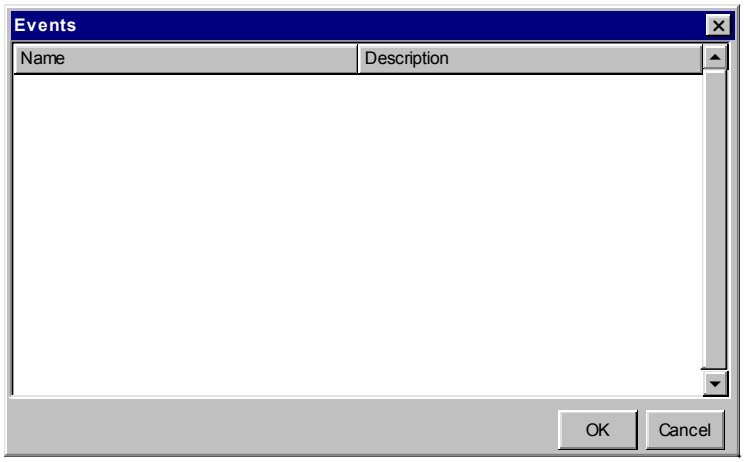
Classes List

List of available classes.

This list contains the name of object classes defined in CodePainter.

All defined classes are stored in the file CLASSES.CPL under the Painter's Classes directory. To add new classes you simply need to define them in this file.

7.9.7 Events



Picture 362 - Events

Events list

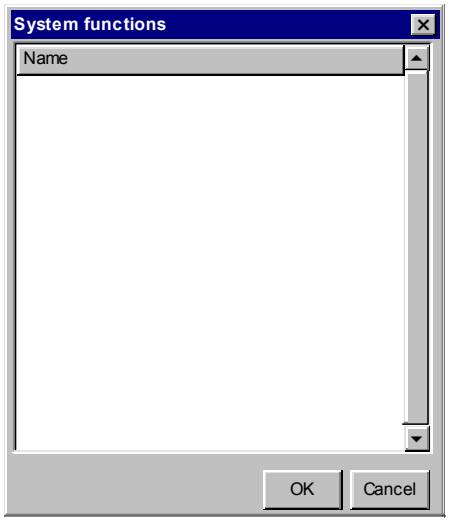
List of Events.

At run-time CodePainter notifies events to the objects in the dialog windows. Objects will therefore be able to react to given situations, such as data loading, change of an element or saving a record.

This list contains the name and brief description of all events that can be notified.

7.9.8 System Functions

Picture 363 - System Functions

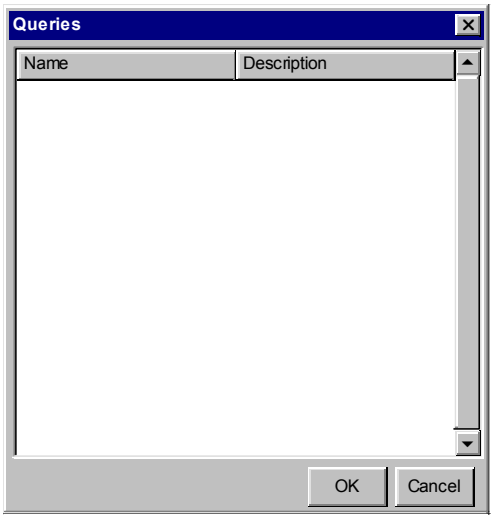


System Functions

List of system functions.

System functions are identified by strings. This list contains all system functions implemented by CodePainter's run-time system.

7.9.9 Queries



Picture 364 - Queries

Name - Description (Tables list)

List of available tables.

All entities - Master entities

All entities

Displays all tables.

'Master' entities

Displays only tables that belong to a to an entity of this class.

Design links

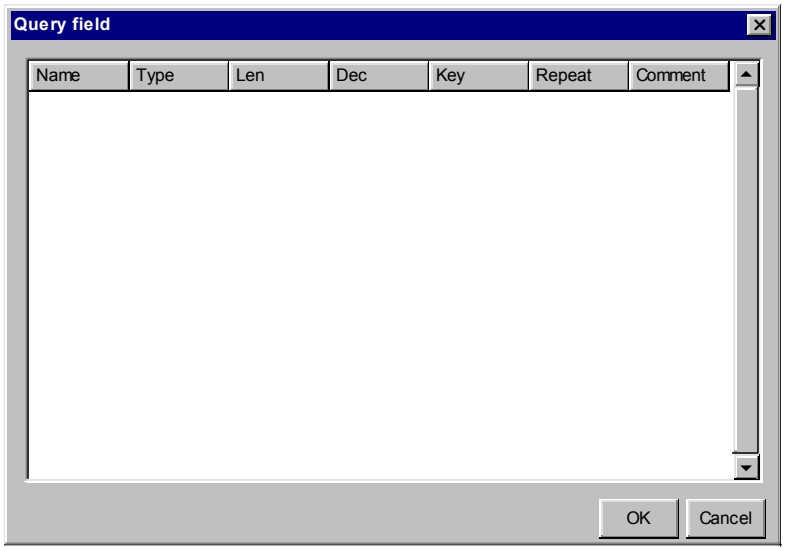
Defines whether to display all tables or only those associated to a table for which a link has been defined in the design plan.

When the flag is set the list displays only linked tables having referential integrity. When the flag is not set the list displays all tables.

You can create links in the Codify phase without going back to the design plan. The code to manage the link is created but the referential integrity to the database is omitted.

7.9.10 Query Field

Picture 365 - Query Field: Page 1



Fields list

Field list.

This list contains all fields of the selected query.

Fields list

Field list.

This list contains all files of the selected entity.

7.9.11 Design Notes

Picture 366 - Design Notes

Design Notes

Displays the notes added to the elements during the Design phase.

7.10 Productinformation

The Help menu option 'About' displays information on the product.

7.10.1 About

The About dialog window contains information on the procedure in use.

The logo CODE**PAINTER REVOLUTION** is displayed and information on the version, series number and build number are given.

The build number helps you identifying CodePainter's distribution, no matter if the product is released totally or partially or through patches.

Picture 367 - About: Page 1

Chapter 8

Routine

8.1 Introduction

Developing SW applications you are often required to create procedures and functions to process or check data in managing procedures. The *Routine Painter* helps you creating and maintaining these procedures and functions that are called inside the various application procedures.

The **Routine Painter** combines flexibility and versatility of C.A.S.E. tools with database oriented instructions, thus offering a user friendly environment.

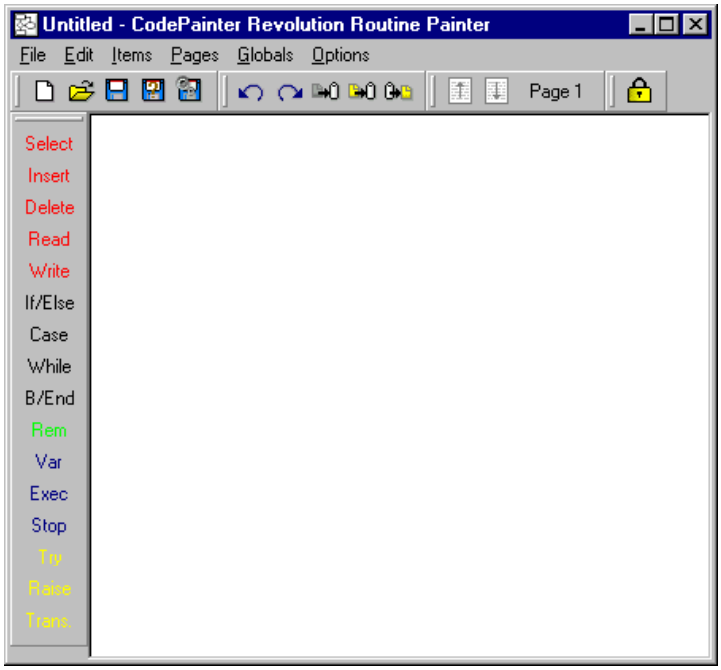
The way how you can add instructions to the entity makes you save time and helps you during the testing of procedures. The possibility to '*collapse*' or '*expand*' instructions blocks/cycles makes the routine more understandable and therefore reduces the error rate.

Using the 'Routine Painter' you can create complex processing procedures based on database tables, import/export files and validate data entries.

8.2 Working Logic

In the Routine Painter you can open or create routine procedures. A set of toolbars help you interacting with the routines.

Picture 368 - The Routine Painter: New Form



To open existing Routine entities open the 'File' menu and select the 'Open' option. All existing Routine entities are listed. The routine includes only information defined in the Design Phase, i.e. procedure name, title, template and tables.

You can also create new Routine entities basing on the data dictionary.

To create new entities open the 'File' menu and select the 'New' option. In this case you can define the same information as you would do opening the 'Globals' menu and selecting the 'Global' option. You can choose amongst two templates:

File	Description
DBBAP	Procedures Generation Template
DBBAPF	Functions Generation Template

Using the Routine Painter you can '*design*' the procedure/function that must be executed. When the Code is generated the macro instructions defined in the '*design*' phase will be '*expanded*' in source language instructions.

N.B.

For the MS Visual FoxPro version the created functions are stored during the design phase in a general file (CP_FUNC.PRG).

Procedures and functions are similar as they both have one entry and one exit point only. 'Procedures' process data and don't have returns. 'Functions' receive input and return values.

The Routine Painter exploits the '*Main Chain*' methodology to identify routines' entry and exit points.

The procedures main chain starts with the declaration of variables and ends with the last element.

The functions main chain starts with the declaration of parameters and ends with the 'Stop' instruction. The 'Stop' instruction, when used in this context, does not exit the routine but becomes a 'return <value>' returning the result to the function.

When you design routines you can define a series of instruction *chains*. The one positioned in the top left corner is the '*Main Chain*', the only one for which the code is generated. Other instruction chains are used as support where instruction blocks are '*parked*' and not considered during the code generation.

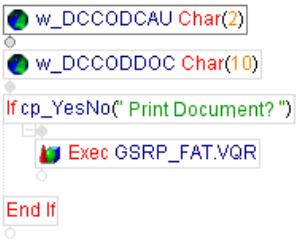
This methodology makes development and debugging easy. To exclude instruction blocks from the generated code you simply need to '*take it out*' from the other instructions.

You can also '*collapse*' instruction blocks by adding comments on the block's functionalities. This makes routines even more understandable looking like 'Flow Charts'.

8.3 Selecting And Adding Elements

You can select a group of instructions clicking and dragging the mouse on the routine form, creating a rectangle around the elements you want to select. Each selected element is highlighted by a frame.

Picture 369 - Selected Group of Instructions

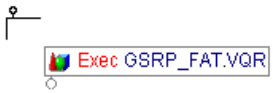


Each selected instruction has an anchor on the bottom left side of its selection frame.

Picture 370 - Instruction Selection

Routine instructions are linked through their anchors. Selecting an instruction the mouse changes into a full anchor. Drag the mouse to the other instruction's anchor. The two instructions are linked.

Picture 371 - Selected And Dragged Instruction



To select and move one or more linked instructions you simply need to select the top one.

You can nest more instruction chains easily selecting and dragging the inner chain into the outer one.

Once a group of linked elements is selected you can move the cursor within the selection using the '**<Arrows>**' keys. Pressing 'Up' or 'Down' you will move vertically in the instruction chain. Pressing 'Right' or 'Left' you will move within the elements levels.

For instance, if two instruction and a 'While' cycle are linked you can move up or down in the main chain using the Up and Down <Arrow> keys, while you can enter and exit cycle instruction blocks using the Left or Right <Arrow> keys

In all instructions made of more parts ('Begin/End', 'If/Else', 'Case/EndCase') or in which scrolling cycles ('While', 'Select') have been defined, you can add instruction blocks. These blocks can then be '*collapsed*' or '*expanded*' easily clicking the '-' or '+' symbols under the instructions. The same result can be obtained selecting the instruction and pressing the '-' or '+' keys.

To add elements to a chain you can add the desired instruction in the routine form and then drag it where you want to add it until the target anchor is highlighted.

You can also add elements to a chain directly from the chain: either pressing '**<Ctrl> + <Letter>**' or '**<Ins>**'. The instruction is added beneath the selected chain element.

For the use of the shortcuts '<Ctrl> + <Letter>' please refer to the 'Items Menu' section.

Pressing the <Ins> key a combobox is opened from which you can select one of instructions contained in the 'Items Menu'.

Picture 372 - Combobox To Add New Instructions

To select instructions in the combobox you need to press the first letter of the instruction. Next to the combobox you can write a statement.

Picture 373 - Example Of A Statement For 'Var'

The statement takes on different meanings depending on the selected instruction. When you select a 'Variable' in the statement fields you need to define the name, type and size.

When an instruction chain is selected you can delete the entire chain pressing the 'Del' key or right clicking the chain and selecting delete.

Not all instructions added to the routine are generated, but only those linked directly or undirectly with the first instruction in the '*Main Chain*'. The '*Main Chain*' is the one placed in the top left corner of the routine form.

The 'Main Chain' methodology has been introduced to make you save time in testing and adjusting elements. To test an element you simply need to link it to the 'Main Chain'. To exclude elements from the chain you simply need to delete the link. Not linked elements are considered comments.

N.B.

All instructions, which are not linked to the first element of the main chain will NOT be generated.

N.B.

Variables will be generated even if they are not linked to the first element.

8.4 Routine Painter Toolbars

Let us now analyze the toolbars available for the Routine Painter that help you interacting with the tool.

8.4.1 Painter Tools

The Painter Tools Toolbar allows you to add elements to the routine in use. It has the same functionalities as the 'Items' menu.

Picture 374 - Painter Tools Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
Select	The 'Select' instruction is used to select and scroll tables.
Insert	The 'Insert' instruction is used to input records in tables.
Delete	The 'Delete' instruction is used to cancel records from tables.
Read	The 'Read' instruction is used to read records from tables.
Write	The 'Write' instructions is used to write records on tables.
If/Else	The 'If/Else' instructions is used to add the conditional statements 'If - Then - Else'.
Case	The 'Case' instruction is used to build the conditional structure 'Case/ End Case'.
While	The 'While' instruction is used to add 'While' loops.
B/End	The 'B/End' instruction is used to add the limits 'Begin - End' to instruction blocks.
Rem	The 'Rem' instruction is used to add remarks.
Var	The 'VAR' instruction is used to add statements that define/assign variables.
Exec	The 'Exec' instruction is used to add execution statements for external objects.
Stop	The 'Stop' instruction is used to add the return command, i.e to end the routine.
Try	The 'Try' instruction is used to add 'Try/Catch' statements.
Raise	The 'Raise' instruction is used to add 'Raise' statements.
Trans	The 'Trans' instruction is used to add statements that manage transactions.

8.4.2 File Toolbar

The 'File' toolbar has a set of button that help you interacting with the tool. This toolbar has the same functionalities as the 'File' menu.

Picture 375 - File Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Creates new entities.
	Opens existing entities.
	Saves the changes made to the current entity.
	Saves the current entity with a different name.
	Saves and generates the source code for the current entity.

8.4.3 Align Toolbar

The Align Toolbar allows you to lock instruction blocks in the defined position.

Picture 376 - Align Toolbar

Button	Meaning
	Locks instruction blocks in the defined position.

8.4.4 Clipboard Toolbar

The 'Clipboard' toolbar allows you to execute typical Window's commands such as Undo, Redo, Cut, Copy and Paste.

Picture 377 - Clipboard Toolbar

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).
	The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).
	The 'Copy' option copies one or more selected elements.
	The 'Cut' option deletes selected elements.
	The 'Paste' option pastes copied or cutted elements in the selected position.

8.4.5 Pages Toolbar

Using the 'Pages' toolbar you can browse the entity's pages. The toolbar shows two buttons and the number of the page in which you are positioned.

Picture 378 - Pages Toolbar

Here to follow you will find a brief description of the buttons and their meanings

Button	Meaning
	Moves the edit functionality from the previous page to the current page. This button does not work if you are on the first page.
	Moves the edit functionality from the following page to the current page. This button does not work if you are on the last page.

8.5 Main Menu

8.5.1 File

Picture 379 - File Menu

The 'File' menu has a set of options to:

- Create new entities.
- Load existing entities.
- Save changes to the current entity.
- Save the current entity with a different name.
- Save and generate the current entity.
- Read information about CODE**PAINTER REVOLUTION**.
- Exit the tool.

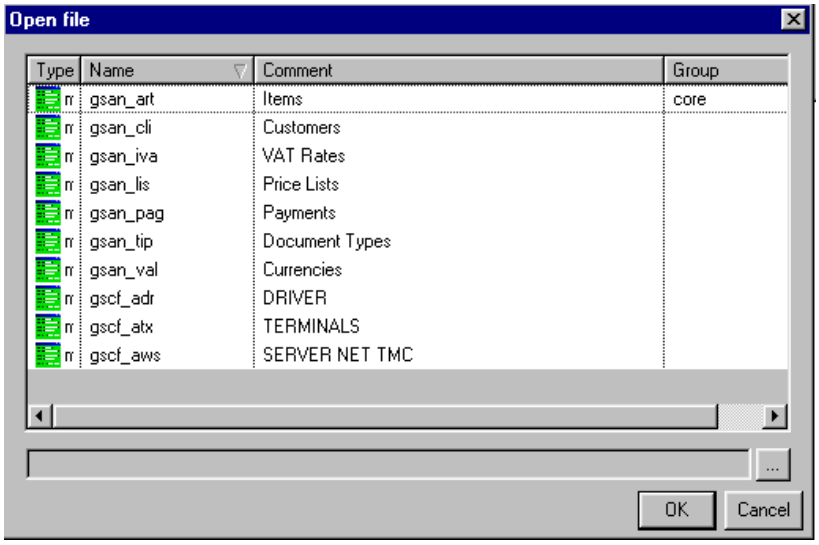
New

The 'New' option closes the current entity asking whether you want to save the changes or not and opens a new entity.

Open...

The 'Open' option loads definition files belonging to the current project.

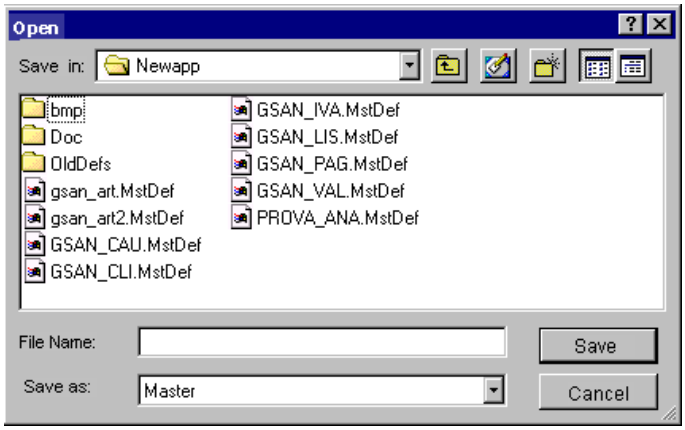
Picture 380 - Open File



The dialog window lists all entities in the project. You can order files by Type, Name, Comment or Group clicking the corresponding column. You can identify the sorting column by the arrow next to the column description.

Picture 381 - Sort Columns

Clicking the '...' button you can browse to search entities of the same type belonging to other project modules.



Picture 382 - Open Dialog Window To Select Design Plans

Save

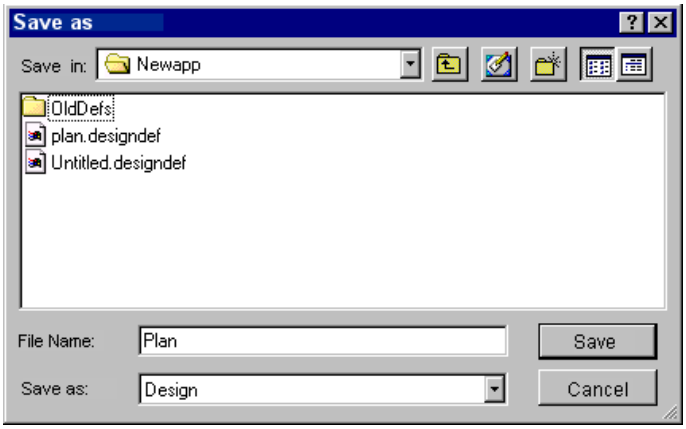
The 'Save' option saves the entity in use.

When you save the entity back-up file (.BAK) is created to store the entity without including the latest changes.

Save As...

The 'Save As' option saves the entity with a different name.

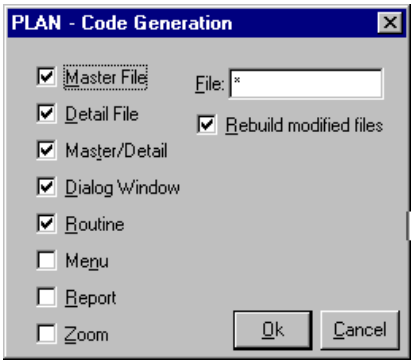
Picture 383 - Save As Dialog Window



Save and generate

The 'Save And Generate' option saves the current entity and runs the generation. The same is done opening the 'Generation' menu and selecting the 'Design'.

Picture 384 - Code Generation



About

The 'About' option displays information about the product.

For more information please refer to section 'Product Information'.

Exit

The 'Exit' option makes you exit the tool asking you whether you want to save the changes or not.

8.5.2 Edit Menu

Picture 385 - Edit Menu

The 'Edit' menu has a set of options to:

- Do /undo commands.
- Copy, delete and add recorded objects.
- Search, display and replacing objects.
- Display and hide toolbars.

Undo

The 'Undo' option undoes the last action. You can undo the latest actions in reverse sequence (from the last to the first). The option tells you what you are about to undo (e.g. Undo Move).

Redo

The 'Redo' option executes the next action in the list. The redo option takes you back to the situation you where in before you did Undo. The option tells you what you are about to redo (e.g. Redo Move).

Copy

The 'Copy' option copies one or more selected elements. More elements can be either selected or grouped in a selection frame you can draw with the mouse.

Cut

The 'Cut' option deletes selected elements from the plan.

Paste

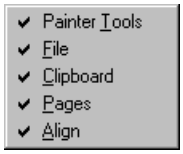
The 'Paste' option pastes copied or cutted elements in the selected position.

Find...

The 'Find' option opens the Find and Replace dialog window to find and/or replace elements.

Toolbar

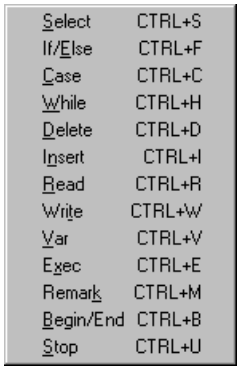
The Toolbars option displays a submenu to enable or disenable toolbars on the Design Painter.



Picture 386 - Toolbars Menu

8.5.3 Items Menu

The 'Items' menu allows adding instructions to the entity in use. It has the same functionality as the 'Painter Tools' toolbar.



Picture 387 - Items Menu

Here to follow you will find a brief description of the various instructions:

Select

The 'Select' instruction is used to select and browse tables.

Insert

The 'Insert' instruction is used to input records in tables.

Delete

The 'Delete' instruction is used to cancel records from tables.

Read

The 'Read' instruction is used to read records from tables.

Write

The 'Write' instructions is used to write records on tables.

If/Else

The 'If/Else' instructions is used to add the conditional statements 'If - Then - Else'.

Case

The 'Case' instruction is used to build the conditional structure 'Case/ End Case'.

While

The 'While' instruction is used to add 'While' loops.

B/End

The 'B/End' instruction is used to add the limits 'Begin - End' to instruction blocks.

Rem

The 'Rem' instruction is used to add remarks.

Var

The 'VAR' instruction is used to add statements that define/assign variables.

Exec

The 'Exec' instruction is used to add execution statements for external objects (Visual Fox instructions, Routines, UTK objects, etc.).

Stop

The 'Stop' instruction is used to add the return command, i.e to end the function.

Try

The 'Try' instruction is used to add 'Try/Catch' statements.

Raise

The 'Raise' instruction is used to add 'Raise' statements.

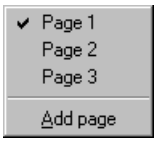
Trans

The 'Trans' instruction is used to add statements that manage transactions.

8.5.4 Pages

Using the 'Pages' menu you can browse the entity's pages. This menu has the same functionalities as the 'Pages' toolbar. New pages can be added only using the 'Pages' menu.

Picture 388 - Pages Menu



The menu displays the names of the available pages. The current page has the 'check' symbol next to the name.

Add Page

Adds a page to the entity. To delete a page you need to delete all contained elements. When you open the entity the next time, CodePainter identifies that the page is empty and deletes it automatically.

N. B.

In order to be able to access the pages in the generated application, at least one field needs to be editable. If your project requires that all fields in a page cannot be edited, you need to add two editable buttons with the system functions 'Page Up' and 'Page Down'.

8.5.5 Globals

In the 'Globals' menu you can define general information on the Routine.

Picture 389 - Globals Menu

Global...

This menu opens the 'Global Definition' window where you can define the generation template, the author, the commissioner, the design in which the entity is included, etc.

Tables...

In this option you can define the working tables.

Font...

It allows to modify the default character and related characteristics, as length, style and effects for the routine in use.

This option allows changing default fonts for the routine in use.

Default fonts are defined in the Front End under 'Project/Project Options'.

Pages Title...

Defines page titles.

Manual Blocks...

This option displays the list of manual areas defined for the entity in use.

8.5.6 Options Menu

The *Options* Menu is used to lock the positioning of instructions in the routine.

Picture 390 - Options Menu

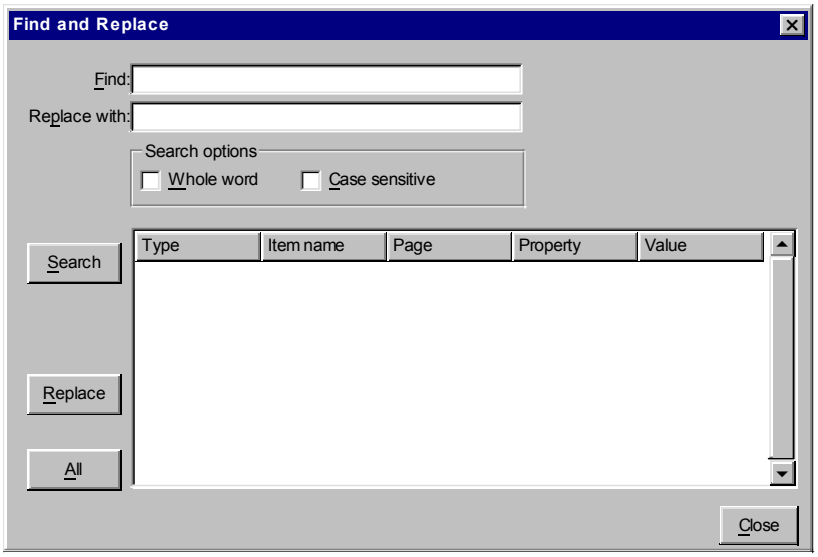
Lock Items

While you work on the routine you may find it helpful to lock the positioning of some instructions that have been already completely defined. The 'Lock Items' option locks the selected elements in their current position.

8.6 Edit Menu Advanced Options

To 'Find and Replace' commands in Routine entities open the 'Edit' menu and select the 'Find' option.

8.6.1 Find And Replace



Picture 391 - Find And Replace

Find

String that must be found.

Replace With

String that must replace the found string.

Whole Word

When this field is flagged the search activity must be performed on whole words only. Otherwise search results could be also substrings of long words.

Case Sensitive

The search activity matches upper and lower Case.

Search Button

Starts the search activity.

List

List of found elements.

Replace Button

Replaces strings found with the string defined in the textbox 'Replace with' for all elements selected in the list.

All Button

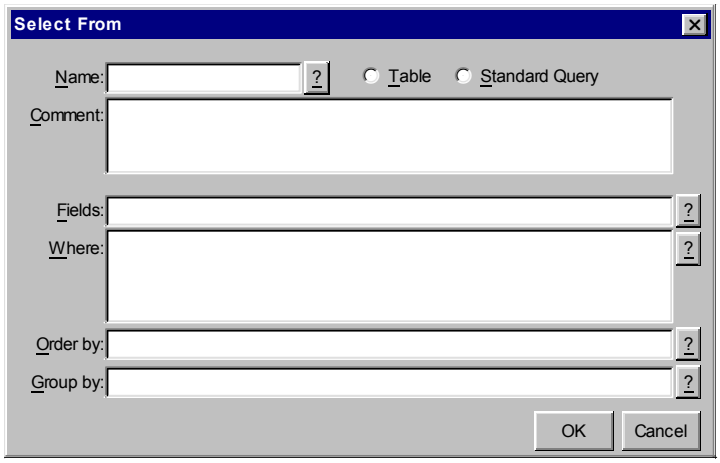
Replaces strings found with the string defined in the textbox 'Replace with' for all elements in the list no matter if elements are selected or not.

8.7 Item Menu

You can access the Item Menu opening the 'Edit' menu and selecting the 'Items' option or right clicking the window. This menu allows you to add new command items to the routine in use. Let us now analyzehe single options in detail.

8.7.1 Select From

The 'Select' instruction executes a query on the database. The temporary cursor '_curs_<query name>' is created, all records are retrieved and the instructions between 'select and 'end select' executed.



Picture 392 - Select From

Name

Name of the table from which data is selected when executing the query (VisualUTK).

Query Type

Table

The query is executed on a database table.

Standard Query

The query is created with the Query Painter.

Field

For queries on tables, it contains the list of fields to be returned. This string is part of the SQL Sentence that extracts data from the database.

Where

For queries on tables, it contains record selection criteria. This string is part of the SQL Sentence that extracts data from the database.

Order By

For queries on tables, it contains the criteria by which the result must be sorted. This string is part of the SQL Sentence that extracts data from the database.

Group By

For queries on tables, it contains the criteria by which data must be grouped. This string is part of the SQL Sentence that extracts data from the database.

Comment

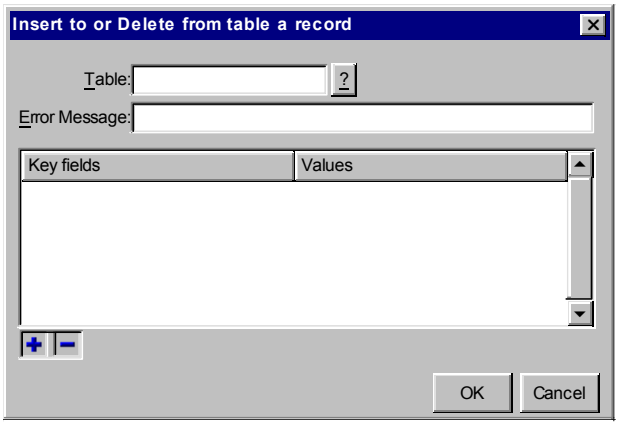
Comment displayed when the structure is collapsed.

The 'Insert' and 'Delete' commands open a dialog window in which you can define records that must be added to or deleted from the table.

8.7.2 Insert To Or Delete Records From Tables

The 'Insert' and 'Delete' commands open a dialog window in which you can define records that must be added to or deleted from the table.

Picture 393 - Insert To Or Delete Records From Tables



Table

Table on which you want to work.

Key Fields

Search expression.

When the primary key is a composite key you need to define the expression building the primary key so that the way the index was build is taken into account.

Error Message

Error message displayed when the activity is terminated without success.

Values

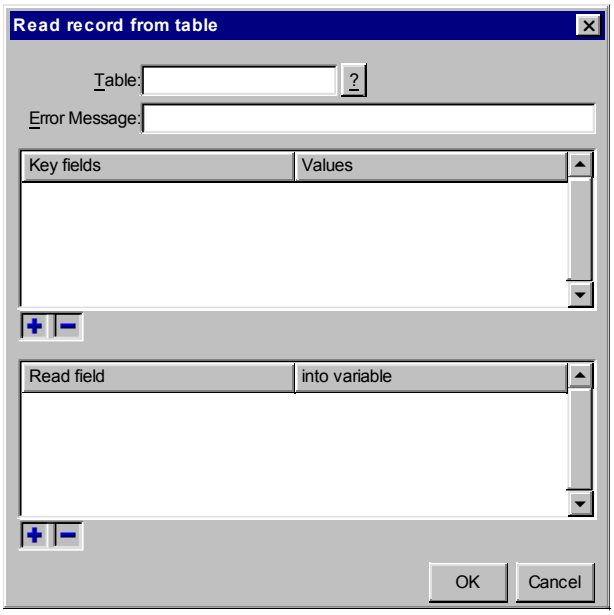
List of values.

This list has two columns: the left one contains the fields building the primary key of the table in which you are working. The right one contains the expression giving the value to each field.

When you use 'insert' activities the list is used to write new data in the database. When you use 'delete' activities the list is used to find the record that must be deleted.

8.7.3 Read Record From Table

Picture 394 - Read Record From Table



table

The table on which you want to work.

Key Fields

Search expression.When the primary key is a composite key you need to define the expression building the primary key so that the way the index was build is taken into account.

Error Message

Error message displayed when the activity is terminated without success.

Values

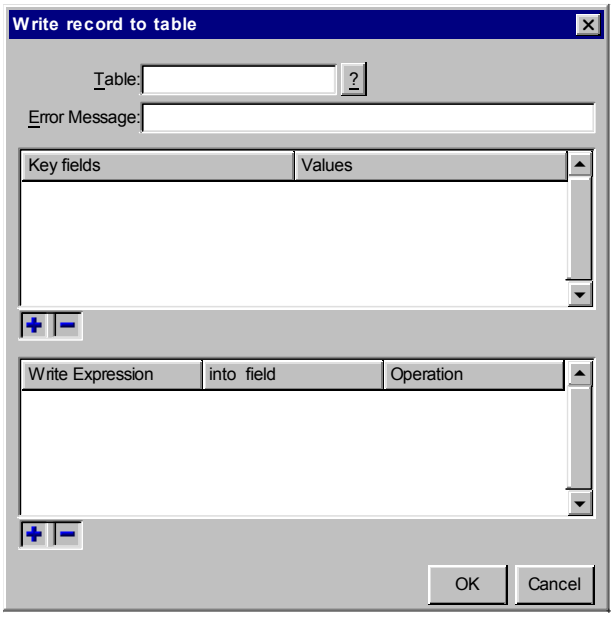
List of fields building the primary key. This list contains search values for the fields building the table's primary key. The record having the primary key with the defined values is read from the table.

Read Values

List of values that must be read. This list contains all fields that must be read from the database. The list has two columns: the left one contains the name of the field that must be read. The right one contains the name of the variable in which the read value must be returned.

8.7.4 Write Record To table

Writes data in a record belonging to the database.



Picture 395 - Write Record To Table: Page 1

Table

Table on which you need to work.

Key Fields

Search expression.When the primary key is a composite key you need to define the expression building the primary key so that the way the index was build is taken into account.

Error Message

Error message displayed when the activity is terminated without success.

Values

List of fields building the primary key. This list contains search values for the fields building the table's primary key. The record having the primary key with the defined values is read from the table.

Write Expression

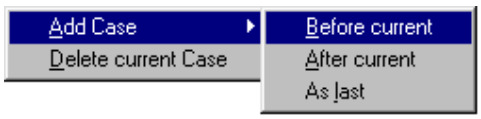
List of values in input. This list contains the field/values combinations that in input in the found record. The list has three columns. The first contains the expression producing the value, the second the field that must be input and the third the transaction that must be applied. The transaction usually used is '=' so that the current value is replaced by the new one. You can also define '+', '-' or a field as you would to calculate totals in links defined in the Design phase. Here you can also use variables because the routine must not reverse transactions.

8.7.5 'Case' statement

The 'Case' instruction allows adding to the Routine commands like:

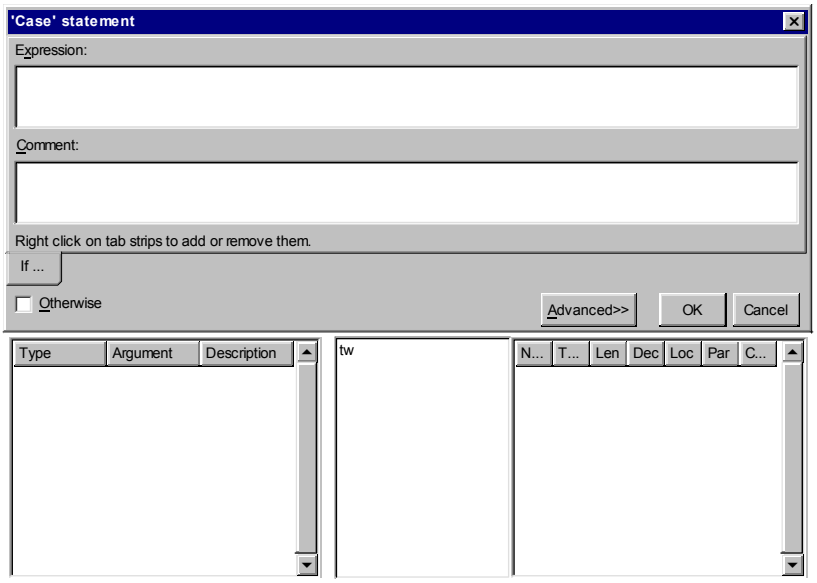
```
Case <condition 1>
Case <condition 2>
Case <condition n>
Otherwise
EndCase
```

The Conditions are defined in the 'Expression' area. In order to add or cancel conditions you need to open the menu on the tab-strip right clicking on the tab-strip name.



Picture 396 - Menu Relevant To The 'If' Instruction

Picture 397 - 'Case' Statement



Expression

Contains the conditional expression of the branch identified by the current tabstrip.

Comment

Comment of the current conditional branch. It is displayed when the branch is collapsed.

Advanced>>

Reduces or increases the size of the current property window and thus allows to use the editing tools.

Context List

List of working entities displaying in alphabetical order the ending of the words you are typing in the 'Expression' field. The list is populated as soon as you start writing anything: variables, tables, manual areas, queries, etc. Here to follow you find the key combinations to select or scroll the list:

- "Ctrl + PgDown": scrolls down;
- "Ctrl + PgUp": scrolls up;
- "Ctrl + ENTER": writes the selected element in the Expression field.

Context Treeview

The Treeview is populated while you select the Exec properties. It can contain the list of tables or variables, or of available queries, etc.

Properties List

The list contains all properties related to the Context Treeview.

Otherwise

Adds the 'Otherwise' branch to the conditional structure.

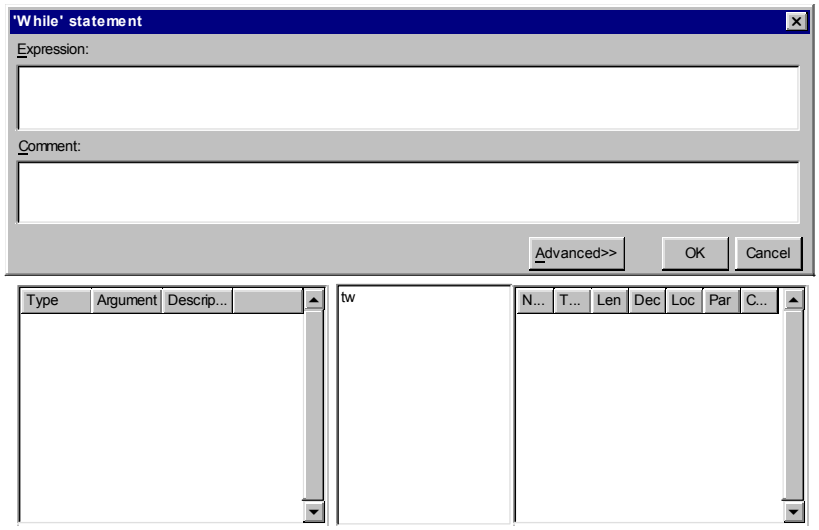
8.7.6 'While' Statement

The 'While' instruction allows you to add interactions to the routine:

```
While <condition 1>
  ...
EndWhile
```

Instructions added to the 'While' branch are executed until the condition defined in the in the 'Expression' property is 'false'.

Picture 398 - 'While' Statement



Expression

Contains exit conditions.

Comment

Comment of the interaction branch. It is displayed when the branch is collapsed.

Context List

List of working entities displaying in alphabetical order the ending of the words you are typing in the 'Expression' field. The list is populated as soon as you start writing anything: variables, tables, manual areas, queries, etc. Here to follow you find the key combinations to select or scroll the list:

- "Ctrl + PgDown": scrolls down;
- "Ctrl + PgUp": scrolls up;
- "Ctrl + ENTER": writes the selected element in the Expression field.

Context Treeview

The Treeview is populated while you select the Exec properties. It can contain the list of tables or variables, or of available queries, etc.

Properties List

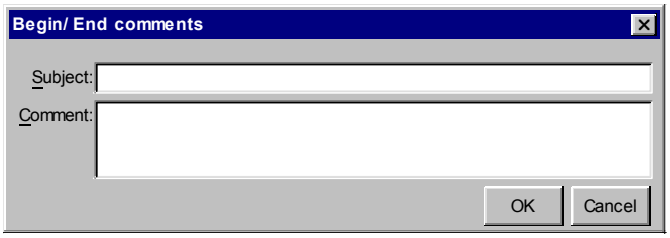
The list contains all properties related to the Context Treeview.

Advanced>>

Reduces or increases the size of the current property window and thus allows to use the editing tools.

8.7.7 Begin/End comments

The 'Begin/End' command groups instructions that perform a functionality that can be seen as a single command. It is similar to a procedure but has a less important meaning and it is easier to use. To display Begin/End instructions you simply need to expand the branch. To display procedures you need to go to the page or file where the procedure is declared.



Picture 399 - Begin/End Comments

Subject

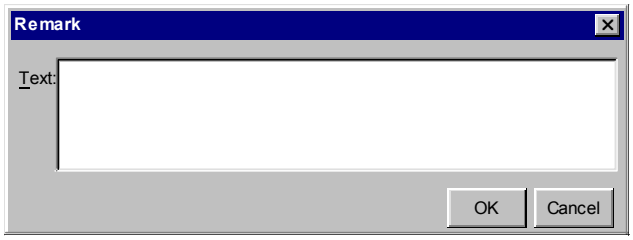
Identifies the subject of the 'Begin/End' instruction. It contains a brief description of the instructions contained in the branch.

Comment

It details the notes of the 'Begin/End' element. It contains a more detailed description than the one contained in 'Subject'. This information is displayed in 'View' when the branch is collapsed.

8.7.8 Remark

Picture 400 - Remark

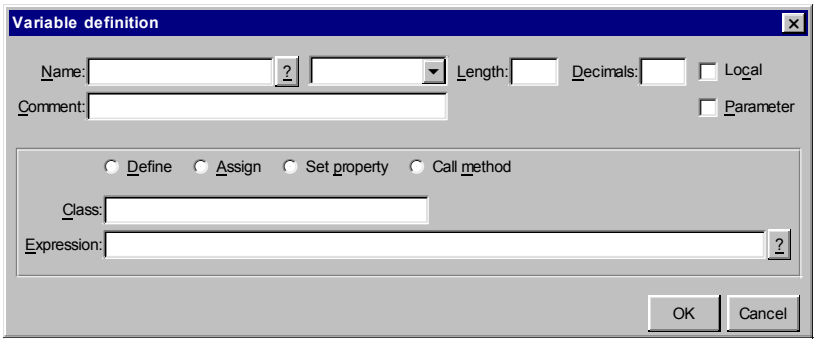


Text

Comment text.

8.7.9 Variable Definition

This dialog window allows to declare variables, define values or call methods and set properties for variables pointing to objects.



Picture 401 - Variable Definition

Name

Name of the variable.In routine procedures the 'w_' rule is not applied. You can define variables having the prefix 'w_' but the string will not be automatically added as happens in the other painters.

Type

Logic

Logical variable.

Memo

Memo variable.

Date

Date variable.

Object

The variable points to an object.

Character

Character variable.

Numeric

Numeric variable.

Length

Variable length. For character and numeric variables you need to define the size of the variable's value.

Decimals

Number of decimals. For numeric variables you can define the number of decimals allowed.

Local

Local variable. Defines whether the defined variable is local to the routine or must be searched in the calling procedure.

Parameter

Routine parameter. Defines whether the variable is a routine parameter. Parameters are initialized from the calling procedure.

Routines generated with CodePainter have two ways of communicating with the calling procedure, namely using parameters or global variables. In the first case the calling procedure must define the values while it calls the routine and cannot receive return values.

In the second case the calling procedure creates a local variable. The routine must declare a variable with the same name as the local variable and set the variable flag to local. This enables to transfer data back and forth between the routine and the calling procedure.

Comment

Comment displayed in the list of defined variables

Definition Type

Call method

The instruction is the call to an object method.

Define

The instruction is a variable definition.

Assign

The instruction defines a value for the variable.

Set property

The instruction is sets the properties for an object variable.

Class

When variables point at objects contained in this textbox you can define the declared object class or the name of the property or the name of the method.

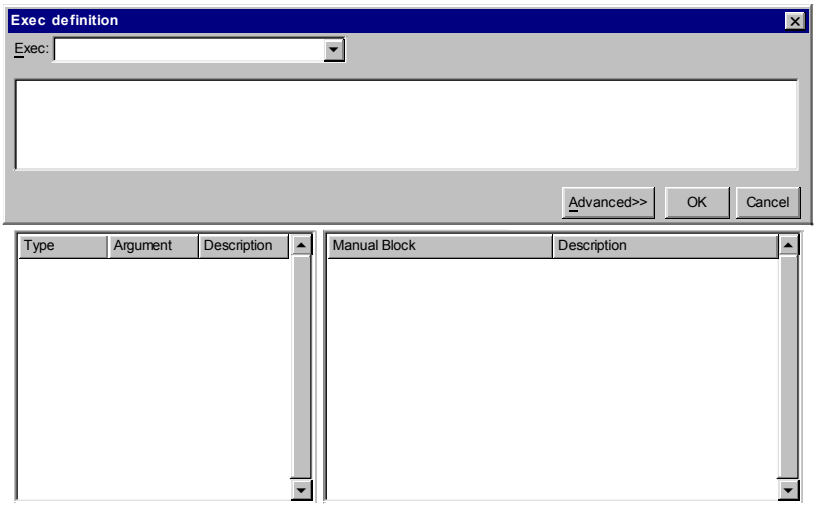
Expression

This expression must be assigned to variables or properties of an object variable. When the method is called method parameters must be defined using commas to separate them.

8.7.10 Exec Definition

The 'Exec' instruction is used to write instructions, commands, dialogs etc., i.e. all those elements that don't find 'place' within other instructions. Usually these instructions are made by a 'single row' and are not bound to the database and transaction management. These instructions can be defined using 'Exec' as detailed below:

Picture 402 - Exec Definition



Exec

Allows selecting the instruction type that must be executed.

UTK Object

Executes 'UTK' files: the 'Expression' property must contain an output file created with the routine framework.

Report

Executes 'Report' files: the 'Expression' property must contain the name of the file created using the 'report' tool.

Dialog Window

Executes 'Dialog Window' files: the 'Expression' property must contain a file having the extension 'MskDef'.

Routine

Executes 'Routine files: the 'Expression property must contain a file having the extension 'BtcDef'.

External program

Executes Programs: the 'Expression' property must contain the name of a file, which is external to your project. For example: in Visual FoxPro you can define a .PRG file or a public procedure.

Page

Executes instructions defined in a page: the 'Expression' property must contain a page number of the current routine.

Statement

Executes 'Statements', i.e. executes instructions following the syntax of the target language: the 'Expression' property must contain the instruction.

N. B.
Please note that this kind of instruction within Routines should not be used too frequently. In case the generation language is changed, 'Statement' instructions must be re-written.

Manual Block

Executes 'Manual Areas': the 'Expression' property must contain the name of a manual area defined in the current routine.

Expression

Contains the Expression: takes on different meanings depending on the instruction ('Exec' property) selected.

Advanced>>

Resizes the property window of the current instruction allowing the use of editing tools.

Context List

List of working entities that suggests in alphabetical order the ending of the words you are writing in the 'Expression' field. The list is populated when you start writing anything: variables, tables, manual areas, queries, etc. To scroll the list you need to use the keyboard:

- "Ctrl + PgDown": scroll down;
- "Ctrl + PgUp": scroll up;
- "Ctrl + ENTER": writes the selected element in the 'Expression' field.

Context Treeview

This tree is populated as you select the property type in the 'Exec' field. It can contain the list of available tables, queries, variables, etc.

Property List

This list contains all the properties corresponding to the 'Context Treeview'.

Manual Blocks

This list contains all 'Manual Areas' defined in the current Routine. The list is displayed when, in the 'Exec' property, you select 'Manual Block'.

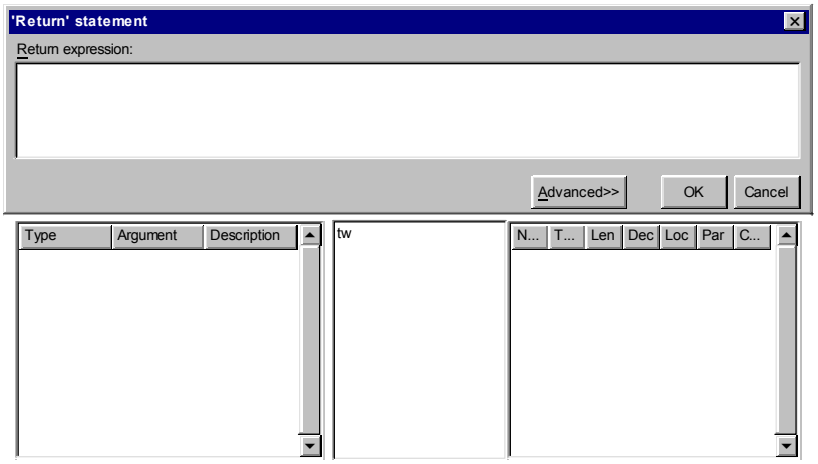
8.7.11 'Return' Statement

The 'Stop' instruction terminates the routine execution where it has been placed. It can also return a value, but in this case the instruction is called 'Return' and the following is displayed:

```
Return <value to be returned>
```

The value that must be returned is defined in the 'Return Expression' property. The 'Context Lists' can be displayed with the 'Advanced' button and using editing tools you have an overview of the properties and objects available in CodePainter's project.

The 'Return' instruction, i.e. 'Stop' + 'return value' is considered only when the function template DBBAPF is set in the routine.



Picture 403 - 'Return' Statement

Return Expression

Value to be returned.

Advanced>>

Reduces or increases the size of the current property window and thus allows to use the editing tools.

Context List

List of working entities displaying in alphabetical order the ending of the words you are typing in the 'Expression' field. The list is populated as soon as you start writing anything: variables, tables, manual areas, queries, etc. Here to follow you find the key combinations to select or scroll the list:

- "Ctrl + PgDown": scrolls down;
- "Ctrl + PgUp": scrolls up;
- "Ctrl + ENTER": writes the selected element in the Expression field.

Context Treeview

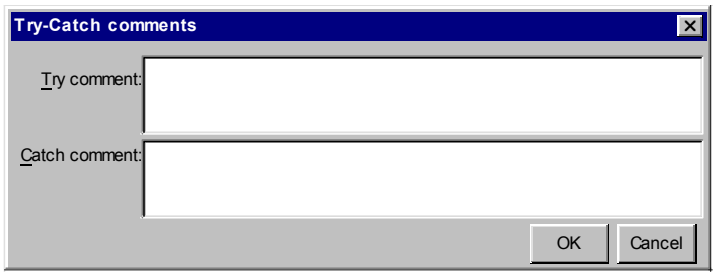
The Treeview is populated while you select the Exec properties. It can contain the list of tables or variables, or of available queries, etc.

Properties List

The list contains all properties related to the Context Treeview.

8.7.12 Try-Catch Comments

Picture 404 - Try-Catch Comments



Try Comment

Comment of the 'Try' branch. The comment is displayed when the branch is collapsed.

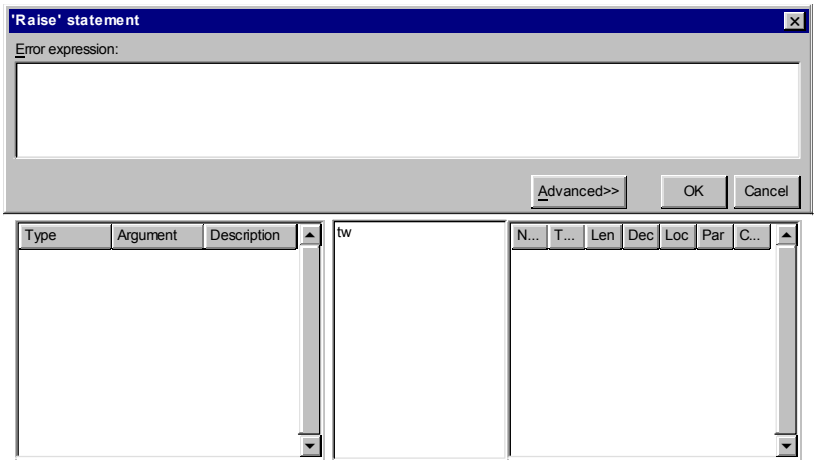
Catch Comment

Comment to the 'Catch' branch. The comment is displayed when the branch is collapsed.

8.7.13 'Raise' Statement

The 'Raise' instruction is used to communicate errors. Depending on where the instruction is placed it impacts the sequence in which the Routine is executed:

- **'Raise' in the 'Try' branch of the 'Try/Catch' instruction:**the routine execution passes on to the 'Catch' branch and the error message defined in the 'Error expression' property is displayed.
- **'Raise' anywhere else:**the execution of the current page is terminated. Unlike the 'Stop' instruction 'Raise' displays the error message defined in the 'Error Expression' property.



Picture 405 - 'Raise' Statement

Error Expression

Error message to be displayed.

Advanced>>

Reduces or increases the size of the current property window and thus allows to use the editing tools.

Context List

List of working entities displaying in alphabetical order the ending of the words you are typing in the 'Expression' field. The list is populated as soon as you start writing anything: variables, tables, manual areas, queries, etc. Here to follow you find the key combinations to select or scroll the list:

- "Ctrl + PgDown": scrolls down;
- "Ctrl + PgUp": scrolls up;
- "Ctrl + ENTER": writes the selected element in the Expression field.

Context Treeview

The Treeview is populated while you select the Exec properties. It can contain the list of tables or variables, or of available queries, etc.

Properties List

The list contains all properties related to the Context Treeview.

8.7.14 Transaction

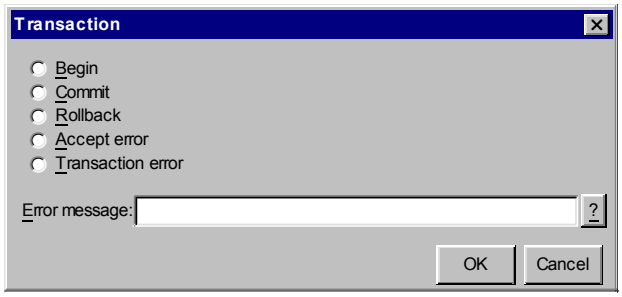
CodePainter generates transactional programs. This means that each input in the database is made by logic units and may involve more inputs.

The transaction command allows the routine to start a new transaction and to group more inputs in one block that will be either saved or refused in whole.

When you don't define explicit transactions, each transaction is processed on its own.

Sometime routines are launched when the system is already under transaction, e.g. in a manual area of a Master File. In these cases there is no need to start a new transaction all inputs will be driven by the active transaction. If an error occurs the entire transaction is refused, including the inputs made by the calling procedure.

Transactions cannot be nested.



Picture 406 - Transaction

Type

Begin

Instruction starting a new transaction.

Commit

Instruction ending a transaction saving data in the database.

Rollback

Instruction ending a transaction abandoning any input made.

Accept error

Instruction accepting transaction errors. The routine is continued erasing the error. This instruction can be used only within a 'Try/Catch' structure.

Transaction error

Instruction overriding transaction errors.

Error Message

Message displayed in case transaction errors are overridden.

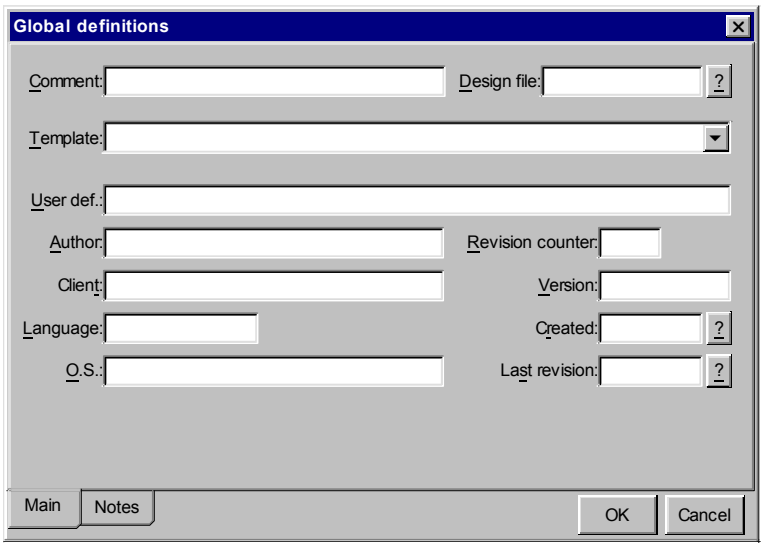
8.8 Global Menu Advanced Options

Let's analyze the 'Global' menu options in detail.

8.8.1 Global Definitions

Main Page

Picture 407 - Global Definitions: Page 1



Comment

Brief comment on the entity.

This comment is displayed in the project file so that you can search entities by file name (i.e. the associated procedure), or by brief description.

Design File

Name of the design file associated to this entity.

Each entity uses a design file as its data dictionary. In this property you need to define the name of the design file that is used to display lists of files and fields.

The '?' button displays the list of all design plans defined in the current project.

Template

Name of the template used for code generation.

In this property you need to define the name of the template used to generate the source code.

Templates are procedures' skeletons, finalized by the programmer to generate the required source code. Changing the template the generated source code changes as well.

User def

Free string. Can be used by custom templates.

Author

Author of the program. Short text defining the developer in charge of the entity.

Client

The commissioner. Short text to define the customer who commissioned the entity.

Language

Development language. Short text describing the programming language used to develop this entity.

O.S.

Operating System. Short text defining the limits given to the entity by the operating system.

Revision counter

Revision counter.

This read only property shows you how many times the entity has been changed. The number is increased automatically every time the entity is saved.

This property can be used to check whether copied objects have been changed by the original author.

Version

Entity version number.

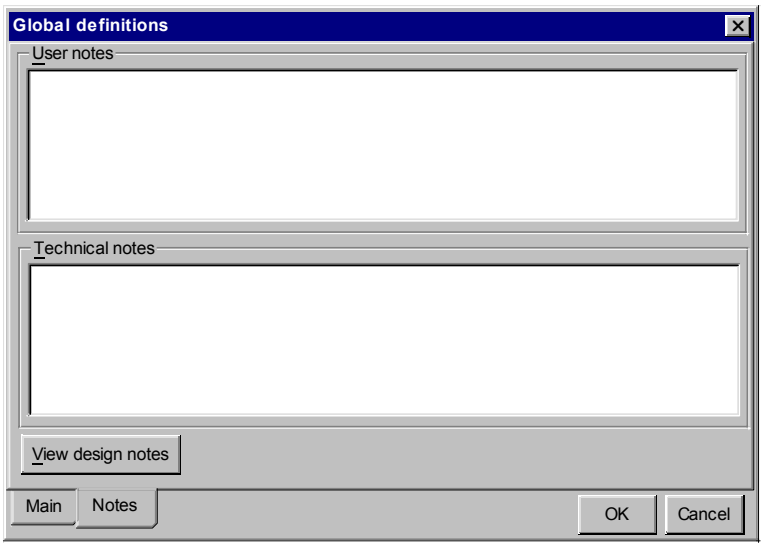
Created

Date in which the entity was created.

Last revision

Date in which the last revision was made.

Notes Page



Picture 408 - Global Definitions: Page 2

User Notes

Notes on the element. These notes are used to create user documentation.

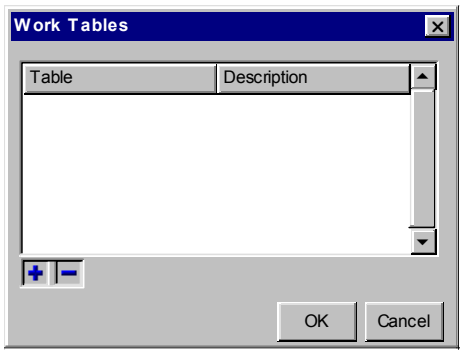
Technical Notes

Technical notes on the element. These notes are used to create technical documentation.

View design notes

8.8.2 Tables

Picture 409 - Tables



Tables

List of tables.

The generated procedure controls that the database contains all tables displayed in this list. When one or more tables are missing an error message is displayed and the procedure is not executed. This is done to avoid errors.

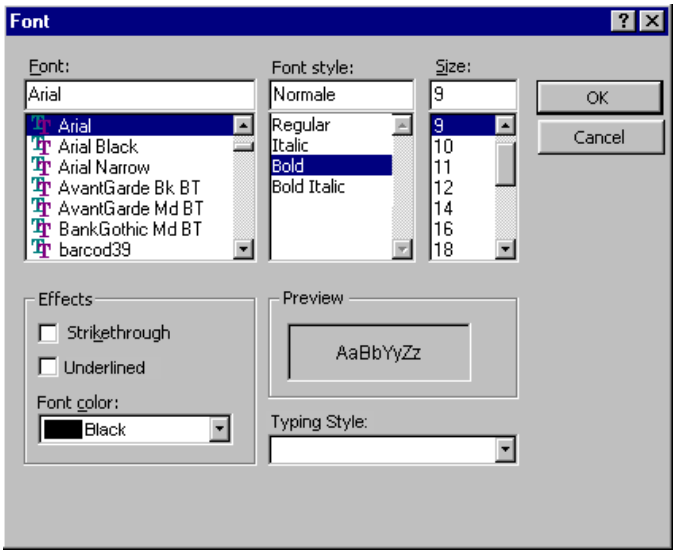
The procedure can also use tables which are not included in the list. Programs should not be executed if they can generate errors.

Tables used in links are automatically added to the list.

8.8.3 Font

Allows changing default fonts for the entity in use.

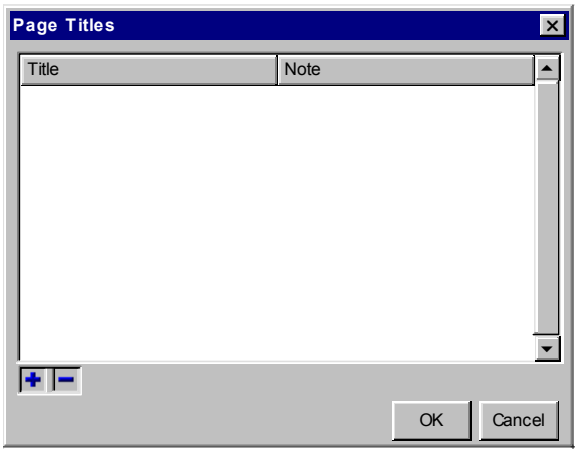
Default fonts in this area are the ones you defined in CodePainter's Front End in the 'Project' menu in the 'Project Options' option.



Picture 410 - Font Definition

8.8.4 Page Titles

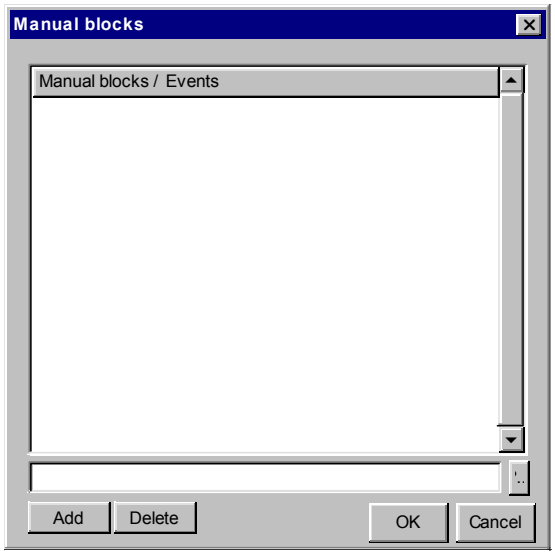
Picture 411 - Page Titles: Page 1



Titles

Page titles. Entities can have dialog windows having more pages. The default title is 'Pag.' followed by the page number. This list allows changing pages' title.

8.8.5 Manual Blocks



Picture 412 - Manual Blocks

Manual Blocks

List of manual areas defined in the template. Manual area containing code lines are displayed in the left column.

List

Name of the manual area to be edited. You can either select it from the list or add different names if the desired manual area is not included in the list.

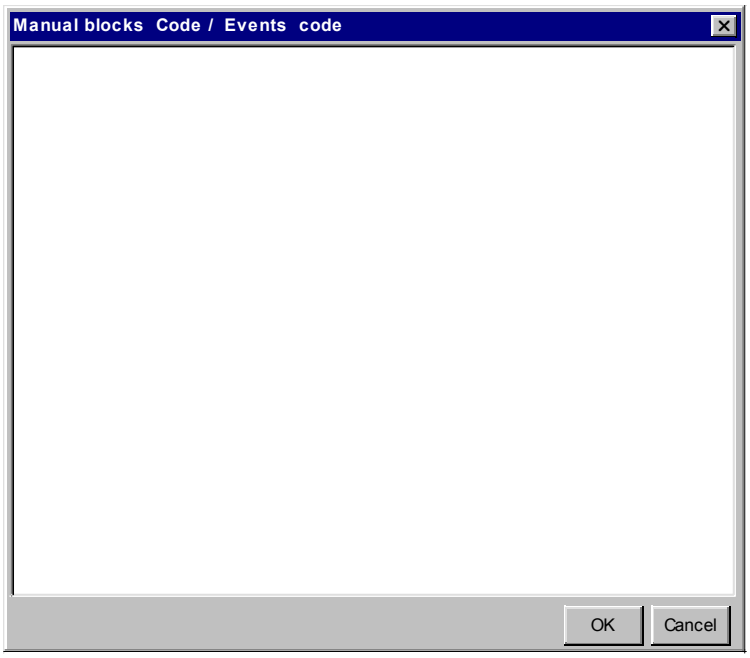
Edit Button

Allows to add, delete and edit manual blocks.

8.8.6 Manual Blocks Code

Editing dialog window for manual areas.

Picture 413 - Manual Blocks Code



Code

Source code contained in the manual area.

8.9 Selection Lists

CodePainter has a set of lists based on the project's data dictionary.

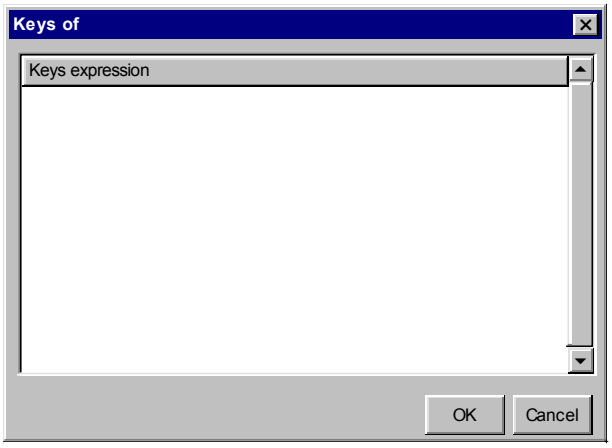
These list are generally activated in the Codify tool using the '?' or '...' buttons. These lists make information required for development available.

These lists can be sorted double clicking the column title. Selected values are input in the desired window section.

The sequence of selected elements can be changed moving them up or down in the list. To move elements you need to position the mouse on the first column on the left and wait until the mouse changes into a hand.

This section details the use of selection lists.

8.9.1 Keys Of



Picture 414 - Keys Of

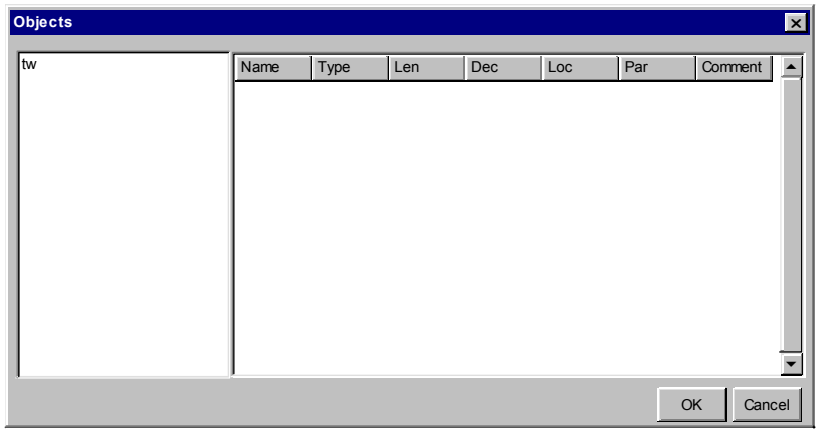
Indexes

List of indexes of the current file. The list displays the expression of each index, so that you can easily select the sort criteria.

8.9.2 Objects

This dialog window is opened clicking '?'. It allows to select table fields, variables or functions depending on the properties window from which it was opened. The selection is made double clicking or pressing 'OK'.

Picture 415 - Objects



Context Treeview

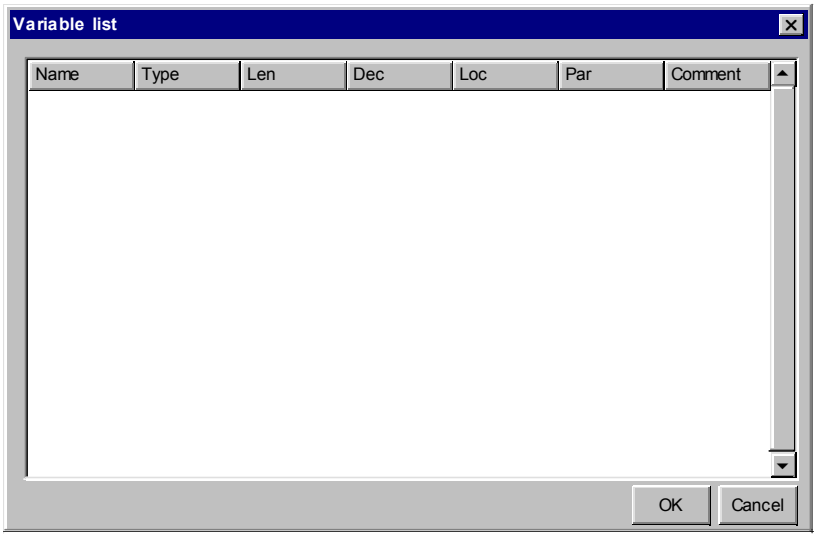
Can contain the list of available tables, queries, variables, functions, etc.

Properties List

This list contains all properties related to 'Context Treeview' selection.

8.9.3 Variable List

Displays the list of variables declared in the routine.



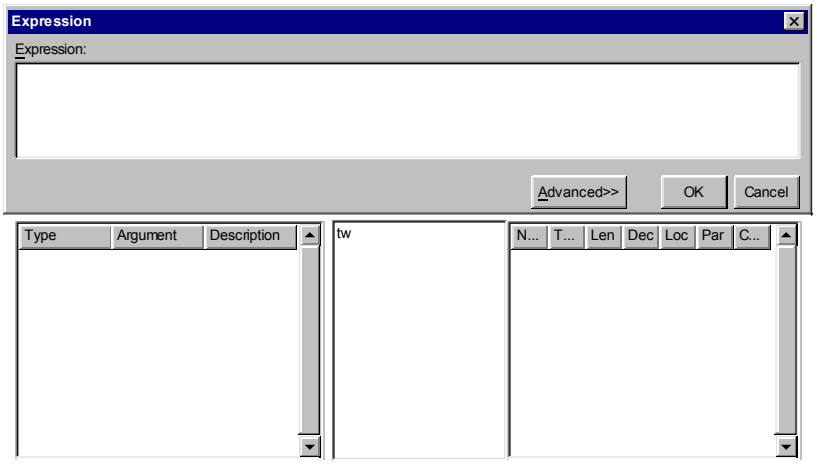
Picture 416 - Variable List

Variable List

List of variables declared in the routine. In case of local variables and parameters the list columns detail type, length and comment.

8.9.4 Expression

Picture 417 - Expression



Expression

Expression. Depending on the kind of window it takes on different meanings:

'Return' Instruction

Value that must be returned.

'Raise' Instruction

Error message.

Advanced>>

Reduces or increases the size of the current property window and thus allows to use the editing tools.

Context List

List of working entities displaying in alphabetical order the ending of the words you are typing in the 'Expression' field. The list is populated as soon as you start writing anything: variables, tables, manual areas, queries, etc. Here to follow you find the key combinations to select or scroll the list:

- "Ctrl + PgDown": scrolls down;
- "Ctrl + PgUp": scrolls up;
- "Ctrl + ENTER": writes the selected element in the Expression field.

Context Treeview

The Treeview is populated while you select the Exec properties. It can contain the list of tables or variables, or of available queries, etc.

Properties List

The list contains all properties related to the Context Treeview.

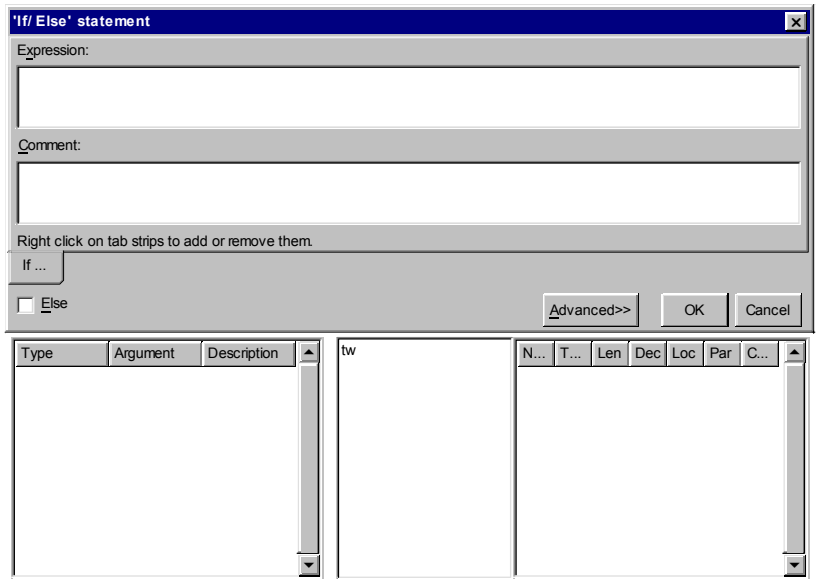
8.9.5 'If/Else' Statement

The 'If/Else' instruction allows to add conditions in the routine, such as:

```
If <condition 1>
ElseIf <condition 2>
ElseIf <condition n>
Else
EndIf
```

Conditions are defined in the 'Expression' property. To add or delete conditions open the menu of the 'If' tab-strip and select one of the options shown in the following picture:

Picture 418 - Menu Of The Tab-strip 'If'



Picture 419 - 'If/Else' Statement: Page 1

Expression

Contains the conditional expression of the branch identified by the current tab-strip.

Comment

Comment of the current conditional branch. The comment is displayed when the branch is collapsed.

Else

Adds the 'Else' branch to the conditional structure.

Advanced>>

Reduces or increases the size of the current property window and thus allows to use the editing tools.

Context List

List of working entities displaying in alphabetical order the ending of the words you are typing in the 'Expression' field. The list is populated as soon as you start writing anything: variables, tables, manual areas, queries, etc. Here to follow you find the key combinations to select or scroll the list:

- "Ctrl + PgDown": scrolls down;
- "Ctrl + PgUp": scrolls up;
- "Ctrl + ENTER": writes the selected element in the Expression field.

Context Treeview

The Treeview is populated while you select the Exec properties. It can contain the list of tables or variables, or of available queries, etc.

Properties List

The list contains all properties related to the Context Treeview.

8.10 Product Information

The Help menu option 'About' displays information on the product.

8.10.1 About

The About dialog window contains information on the procedure in use.

The logo CODE**PAINTER REVOLUTION** is displayed and information on the version, series number and build number are given.

The build number helps you identifying CodePainter's distribution, no matter if the product is released totally or partially or through patches.

Picture 420 - About: Page 1

Chapter 9

Documentation Painter

9.1 Introduction

Creating the documentation for a software application takes time and resources. Once created it must be continuously updated to keep it in line with the product.

CODE**PAINTER REVOLUTION** helps you creating three kinds of documentation: design, technical and user documentation.

Dedicated areas in all development phases allow you to add notes that will become part of the automatically generated documentation.

At design level you can define notes for each entity that build the *Design Documentation*. This kind of documentation has been thought as a tool for users, analysts and developers to discuss the work in progress. Being able to see and 'touch' the SW application in an early stage of the project helps avoiding misunderstandings.

This kind of documentation is also used as a basis for 'User Reference Guides'.

Using the various Painters you can add '*User*' and '*Technical*' notes to each entity. User notes will become part of 'User Reference Guides' and build the basis for the application's On-line Help. Technical notes extend and complete the design documentation supporting maintenance activities. In technical notes you typically define validations defined in the 'Codify' phase giving important information to those technical people that will support the application.

During the 'Documentation Generation' phase XML and HTML files are created. Using the 'Documentation Painter' you can organize and manage these files in order to produce publication files (extension .PUB). *Documentation Files* are a list of hypertextual publications made of a set of linked documents that allow to structure the various kind of documentation.

9.2 Working Logic

The Documentation Painter is divided into two areas and has a toolbar that helps you interacting with the tool.

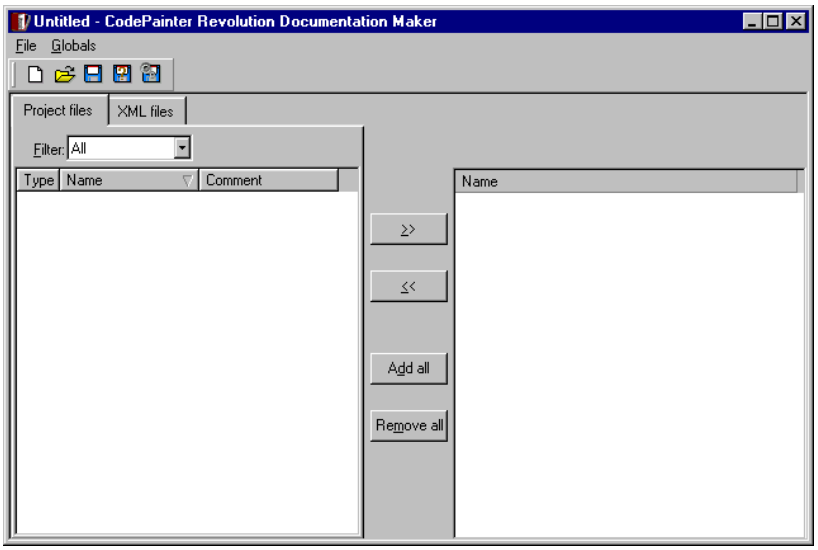
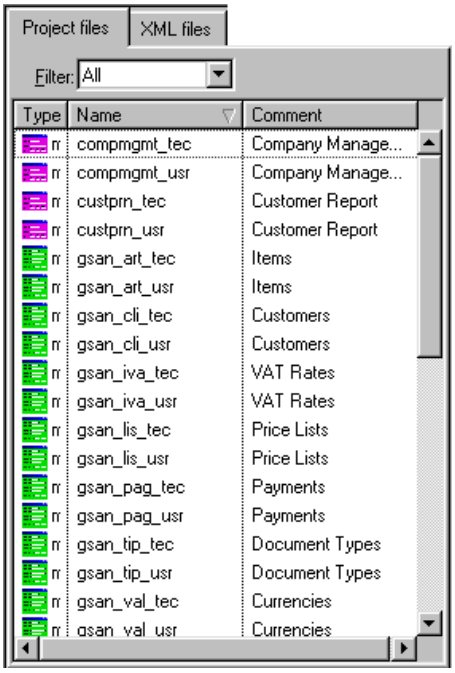


figura 13 - Documentation Painter

The area on the left is made of two tab-strips, namely '*Project File*' and '*XML File*'.

figura 14 - Area On The Left



In the '*Project File*' tab-strip you can see the list of all files in the project that could potentially be used to create publication files. Files can be filtered by type using the 'Filter' combobox.

figura 15 - Filter Combobox

Files used for the Design Documentation have the design file name as prefix. If the default design file name is 'PLAN' then all files for the Design Documentation have the prefix 'PLAN_'.

Technical and User notes are distinguished by the suffixes '_usr' and '_tec'. This means that if a documentation file has technical as well as user notes it will be listed twice: once followed by the suffix '_tec' and once by '_usr'.

The following table explains you the meaning of the various filters.

Filter	Meaning
All	Displays all project files.
Design	Displays design files only.
Codify	Displays files containing technical and user notes ('_tec', '_urs').
Technical	Displays files containing technical notes only ('_tec').
User	Displays files containing user notes only ('_usr').
Other	Displays external files only.

The list of files in the 'Project File' tab-strip can be sorted by type, name or comment in ascending or descending order simply clicking the column header.

The '*XML File*' tabstrip contains all .XML files generated from the corresponding project files. In this area you can also create new XML files, e.g. in order to write general notes on the application.

Files can be filtered by type using the 'Filter' combobox.

figura 16 - Filter Combobox

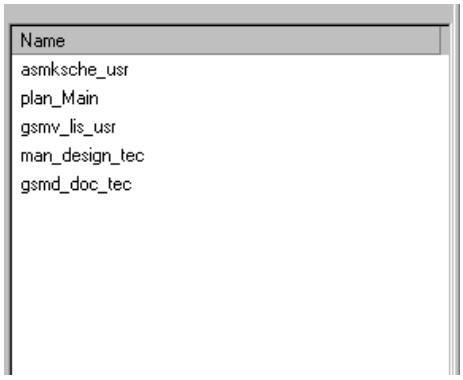
The following table explains you the meaning of the various filters.

Filter	Meaning
All	Displays all project files.
Design	Displays Design files only.
Codify	Displays files containing technical and user notes ('_tec','_urs').
Technical	Displays files containing technical notes only ('_tec').
User	Displays files containing user notes only ('_usr').
Other	Displays external files only.

The list of files in the 'XML Files' tab-strip can be sorted by name in ascending or descending order simply clicking the column header.

The area on the right contains the list of files that make up the publication you are creating.

figura 17 - Publication Area



The current publication file can be made of files contained in the 'Project Files' or 'XML Files' tab-strips.

To add or remove files from the Publication area you can use the buttons between the two window areas.

The following table explains you the meaning of the various buttons.

Button	Meaning
>>	Moves selected files from the left to the right.
<<	Moves selected files from the right to the left
Add all	Moves all files from the left to the right.
Remove all	Moves all files from the right to the left.

The file sequence on the right can be changed selecting the file positioning the cursor on the left of the file, waiting until the mouse changes into a hand and moving it upwards or downwards.

9.3 Documentation Painter Toolbars

Let us now analyze the toolbars available for the Documentation Painter that help you interacting with the tool.

9.3.1 File Toolbar

The 'File' Toolbar has a set of buttons that help you interacting with the Document Painter. The Toolbar has the same options as the 'File' menu.

figura 18 - Toolbar File

Here to follow you will find a brief description of the buttons and their meanings.

Button	Meaning
	Creates new publications.
	Opens existing publications.
	Saves the publication in use.
	Saves the publication in use with a different name.
	Saves and generates the source code for the publication in use.

9.4 Main Menu

Here to follow you will find a brief description of the main menu's options.

9.4.1 File Menu

Picture 421 - File Menu

The 'File' menu has a set of options to:

- Create new publications.
- Load existing publications.
- Save changes made to the publication in use.
- Save the current publication with a different name.
- Save the changes and generate the current publication.
- Read information on CODE**PAINTER REVOLUTION**.
- Exit the tool.

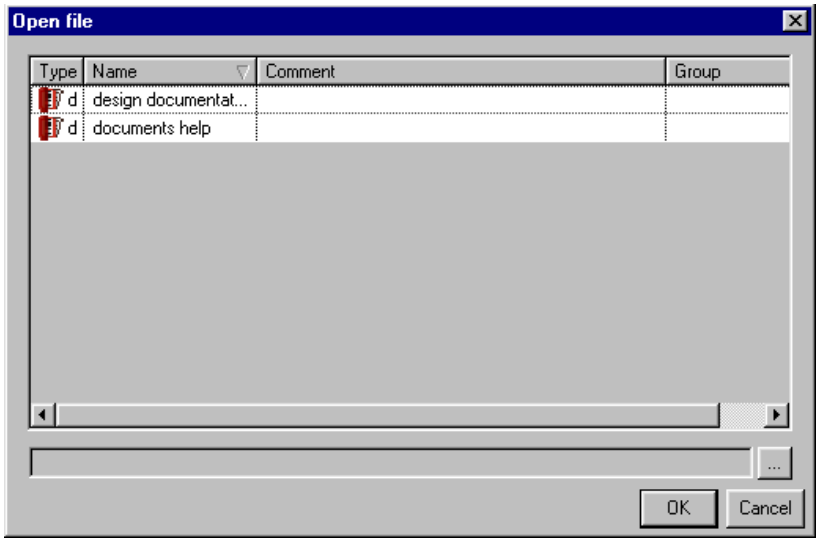
New

The 'New' option closes the current publication asking whether you want to save the changes or not and opens a new publication.

Open...

The 'Open' option loads publication files belonging to the current project.

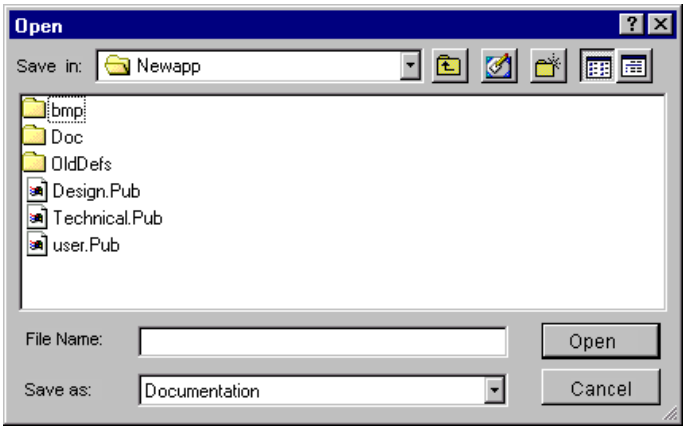
Picture 422 - Open File



The dialog window lists all publication files in the project. You can order files by Type, Name, Comment or Group clicking the corresponding column. You can identify the sorting column by the arrow next to the column description.

Picture 423 - Sort Columns

Clicking the '...' button you can browse to search publication files in other project modules.



Picture 424 - 'Open' Dialog Window To Select Publications

Save

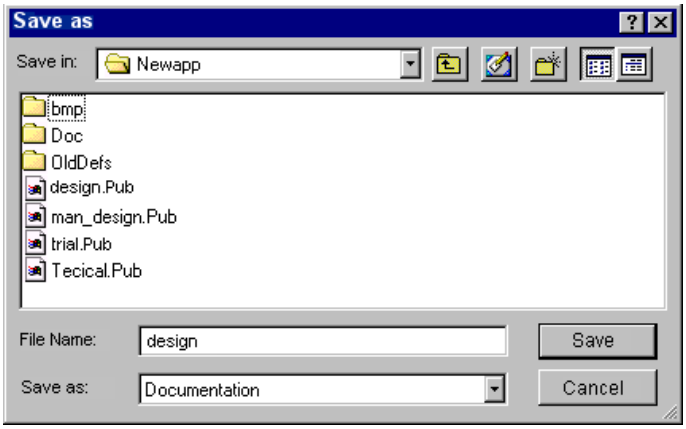
Saves the publication in use.

When you save publication files back-up files (.BAK) are created. These store the file without including the latest changes.

Save As...

The 'Save As' option saves the publication file with a different name.

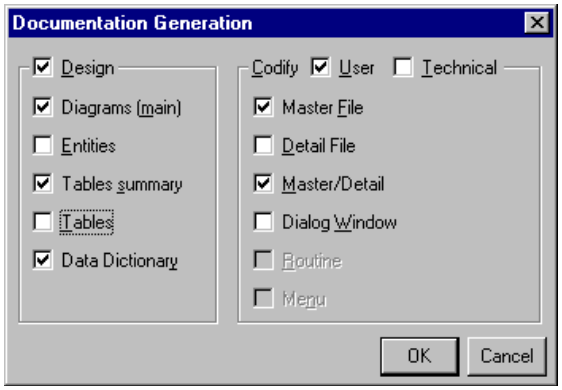
Picture 425 - Save Dialog Window



Save And Generate

The 'Save And Generate' option saves the current publication file and runs the generation. The same is done opening the 'Generation' menu and selecting the 'Documentation' option.

Picture 426 - Code Generation



About

The 'About' option displays information on the product.

For more information please refer to section 'Product Information'.

Exit

The 'Exit' option makes you exit the tool asking whether you want to save the changes or not.

9.4.2 Globals

Picture 427 - Globals Menu

Global...

Selecting the 'Global' option the 'Global definitions' window is opened. Here you can define general information on the documentation. For more information please refer to 'Global Menu Advanced Options'.

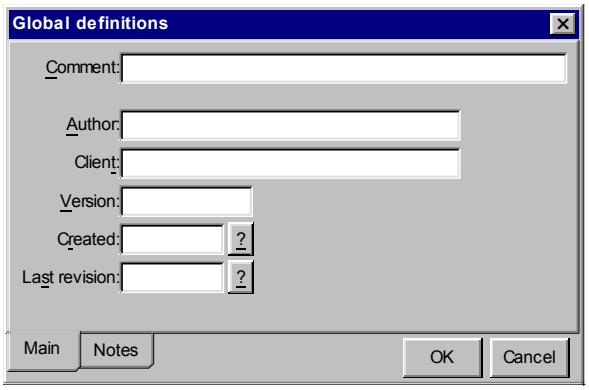
9.5 Global Menu Advanced Options

Let's analyze the 'Global' menu options in detail.

9.5.1 Global Definitions

Main Page

Picture 428 - Global Definitions: Page 1



Comment

Brief comment on the documentation entity.This comment is displayed in the project file so that you can search entities by file name (i.e. the associated procedure), or by brief description.

Author

Author of the program.Short text defining the developer in charge of the documentation entity.

Client

The commissioner.Short text to define the customer who commissioned the reference guides.

Version

Version number of the documentation entity.

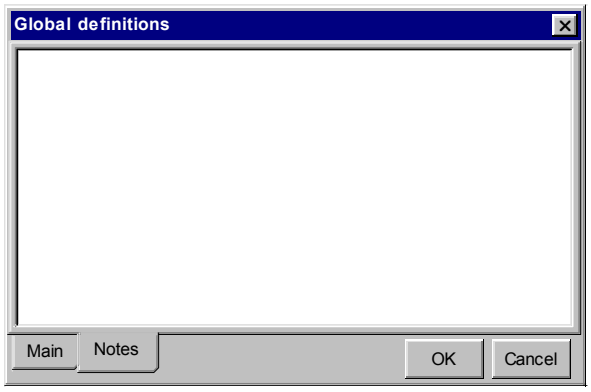
Created

Date in which the documentation entity was created.

Last revision

Date in which the last revision was made.

Notes Page



Picture 429 - Global definitions: Page 2

note

Notes on the element.

9.6 Product Information

The Help menu option 'About' displays information on the product.

9.6.1 About

The About dialog window contains information on the procedure in use.

The logo CODE**PAINTER REVOLUTION** is displayed and information on the version, series number and build number are given.

The build number helps you identifying CodePainter's distribution, no matter if the product is released totally or partially or through patches.

Picture 430 - About: Page 1