confirmit

- User Manual -

Version 5.1

July 11, 2000

The information on these pages describes **confirmit** and it's features as of July 11th, 2000. New features may be introduced into **confirmit** after this date. Please check "News" for latest updates.

Examples depicted herein are fictitious.

TABLE OF CONTENTS

TABLE OF CONTENTS 2
1 Introduction 9
1.1 WHAT IS CONFIRMIT? 9
1.2 ABOUT THIS MANUAL 9
1.2.1 Interview Management 9
1.3 CONFIRMIT USERS 9
1.3.1 Advanced Users 9
1.3.2 Executive Users 10
1.4 ABOUT LOG-IN SESSIONS 10
1.5 SECURITY ISSUES 10
1.5.1 Secure Sockets Layer (SSL) 10
1.5.2 Object Security 10
1.5.3 Respondent Security and Anonymity 10
1.5.4 Back up of Meta Data 11
1.5.5 Back up of Response Data 11
1.6 SYSTEM REQUIREMENTS 11
***1.6.1 confirmit** Users 11*
1.6.2 Respondent Requirement 11
1.6.3 3rd Party Downloads 11
2 Entering confirmit – a General Overview 12
2.1 CONFIRMIT LOGIN 12
2.2 CONFIRMIT MENU STRUCTURE 13
3 General 15
3.1 MAIN PAGE 15
3.2 USER SETTINGS 16
3.3 PROJECT LIST 17
3.4 TASK MANAGEMENT 18
3.5 NEWS PAGE 19
3.6 TEMPLATES 19
3.6.1 Web Interview Templates 19
3.6.1.1 The Fields in the WI Template Edit Modus 20
3.6.1.2 Published Report Templates 20
3.6.2 MS PowerPoint Templates 21
3.7 HELP 21
3.8 FAQ 21
3.9 EXITING CONFIRMIT 21
4 Project Overview 22
4.1 OBJECT PERMISSIONS 23
4.2 DUPLICATE PROJECT 24

4.3 DELETE PROJECT 24
5 Building Questionnaires 25
5.1 CREATING A NEW QUESTIONNAIRE 25
5.2 ADDING NEW OBJECTS TO THE QUESTIONNAIRE 25
5.2.1 Forms 26
5.2.1.1 Text 26
5.2.1.2 Answers 26
5.2.1.3 Scale 28
5.2.1.4 Properties 28
5.2.1.5 Preview 31
5.2.1.6 Tracking 31
5.2.1.7 Languages 31
5.2.2 Conditions 31
5.2.2.1 Description of Fields 32
5.2.2.2 Condition Builder 32
5.2.3 Directives 33
5.2.4 Stop-objects 34
5.2.5 Scripts 34
5.2.6 Folders 34
5.2.7 Loops 35
5.3 EDITING THE ROUTING 36
5.3.1 Moving an Object 36
5.3.2 Duplicating an Object 37
5.3.3 Selecting Multiple Objects 37
5.3.4 Deleting Objects 37
5.3.5 Pitfalls 37
5.4 SCRATCHPAD 37
5.5 LISTS 37
5.6 CONFIRMIT LIBRARY 38
5.6.1 Templates in Library 39
6 CONFIRMIT Scripts 40
6.1 INTRODUCTION 40
6.2 ACCESSING SURVEY VARIABLES – THE F FUNCTION 40
6.2.1 f("qID") 41
6.2.1.1 Single 41
6.2.1.2 Open Text 41
6.2.1.3 Multi 42
6.2.1.4 Multi with OpenText or Ordered Property Checked 42
6.2.1.5 Grid 43
6.2.1.6 Other, Specify 44
6.2.2 f("qID")["precode"] 45

6.2.2.1 Multi 45
6.2.2.2 Multi with Ordered Property Checked 45
6.2.2.3 Multi with OpenText Property Checked 46
6.2.2.4 Grids 46
6.2.3 Loops - f("qID","iteration_precode","iteration_precode",…..) *46*
6.3 CONDITIONS 48
6.3.1 .inc() *48*
6.3.2 .size() *49*
6.3.3 .toNumber() *49*
6.4 PRECODE MASKS 49
6.4.1 a("qID") *49*
6.4.2 set(), nset(), nnset() *49*
6.4.3 .union() *49*
6.4.4 .isect() *50*
6.4.5 .diff() *50*
6.4.6 Precodes *51*
6.5 TEXT SUBSTITUTION/RESPONSE PIPING 51
6.5.1 Other methods *53*
6.5.1.1 Coded Variables (Single Questions and the Elements of Grids) 53
6.5.1.2 Compounds (Multi Questions and Grids) 53
6.5.1.3 Some practical use 53
6.6 VALIDATION CODE 54
6.6.1 Default Validation Rules *54*
6.6.2 Adding Your Own Validation Code *55*
6.6.2.1 A typical Validation Code 55
6.6.2.2 RaiseError() 56
6.6.2.3 Error Messages 56
6.6.2.4 Template Based Error Messages 56
6.6.2.5 Some default functions for Conditions in Validation Code 57
6.7 SCRIPT NODES 58
6.8 USEFUL FUNCTIONS AND METHODS 59
6.8.1.1 Arithmetic functions 59
6.8.1.1.1 Sum() 59
6.8.1.1.2 Count() 59
6.8.1.1.3 Average() 59
6.8.1.1.4 Max() 59
6.8.1.1.5 Min() 59
6.8.2 Conversion *59*
6.8.2.1.1 .toString() 59
6.8.2.1.2 .toNumber() 59
6.8.2.1.3 .toBoolean() 60

6.8.2.2 Classification functions 60
6.8.2.2.1 .size() 60
6.8.2.2.2 .inc() 61
6.8.2.2.3 IsNumeric(arg) 61
6.8.2.2.4 IsInteger(arg) 61
6.8.2.2.5 IsDateFmt(arg, format) 61
6.8.2.2.6 IsDate(day, month, year) 62
6.8.2.2.7 IsEmail(arg) 62
6.8.2.2.8 IsNet(quadIP, quadNet, quadMask) 62
6.8.3 Context information *62*
6.8.3.1.1 CurrentID() 62
6.8.3.1.2 CurrentLang() 62
6.8.3.1.3 InterviewStart() 62
6.8.3.1.4 InterviewEnd() 62
6.8.3.1.5 GetStatus() 62
6.8.3.1.6 SetStatus() 62
6.8.3.1.7 Forward() 63
6.8.4 Browser information *63*
6.8.4.1.1 BrowserType() 63
6.8.4.1.2 BrowserVersion() 63
6.8.4.1.3 RequestIP() 63
6.8.5 Handling sets *63*
6.8.5.1.1 a('qID') 63
6.8.5.1.2 set() 63
6.8.5.1.3 nset() 63
6.8.5.1.4 nnset() 63
6.8.5.1.5 .get() 64
6.8.5.1.6 .label() 64
6.8.5.1.7 .set(v) 64
6.8.5.1.8 .union() 64
6.8.5.1.9 .isect() 64
6.8.5.1.10 .diff() 64
6.8.5.1.11 .add() 64
6.8.5.1.12 .remove() 64
6.8.5.2 General utilities 65
6.8.5.2.1 SendMail(from, to, subject, body) 65
6.8.5.2.2 Filter(qID,prefix) 65
7 Preparing for Data Collection 66
7.1 GENERATING RESPONSE DATABASES 66
7.2 GENERATING INTERVIEW FILES 66
7.2.1 Web Interview Files *66*
7.2.2 Options Controlling Appearance *68*

7.2.2.1 Recipient of Error Notifications 68
7.2.2.2 Include Progress Bar....69
7.2.2.3 Reserve a Fixed Area for Error Messages 69
7.2.2.4 Survey Design 69
7.2.2.5 Body Attributes. 69
7.2.2.6 Compact Layout 69
7.2.3 Options Affecting Interview Behavior 69
7.2.3.1 Generate Back Button....69
7.2.3.2 Allow Modification of Previous Answers 69
7.2.3.3 Answer Required Checks 70
7.2.3.4 Exclusivity Tests 70
7.2.3.5 Other – Specify Checking....70
7.2.3.6 Rank Order Tests....70
7.2.3.7 Disable "Answer Required" Validation....70
7.2.4 Options for Generating Limited Surveys.... 70
7.2.4.1 Limited Survey 70
7.2.4.2 Generate Login Page 70
7.2.5 Pop-Ups 70
7.2.5.1 Generate Pop-up Survey....70
7.2.5.2 An Example of a Script for Pop-up Surveys....71
7.2.5.3 .Pop-up Script Wizard 72
7.2.6 Default Language.... 72
7.3 PROJECT STATUS 72
7.4 PROJECT LOG.... 73
7.5 URL SETUP 73
8 Handling Respondents in Limited Surveys.... 75
8.1 PREPARING THE RESPONDENT LIST 75
8.2 UPLOADING THE RESPONDENT LIST 75
8.3 SENDING E-MAIL 76
8.3.1 Step 1: Respondent Selection 77
8.3.2 Mail Merging Functionality.... 79
8.3.3 Avoiding Unnecessary Respondent Inquiries.... 79
8.3.4 Step 2: Preview and Confirmation.... 80
8.3.5 Sending E-mails 80
8.4 HANDLING RESPONDENTS IN MULTILINGUAL PROJECTS 80
8.4.1 Open Multilingual Surveys.... 80
8.4.2 Limited Multilingual Surveys 80
9 Reporting 81
9.1 INTRODUCTION 81
9.2 CHECKING A SINGLE QUESTION.... 81
9.3 BUILDING REPORTS 82

9.4 BUILDING CROSSTABS 83
9.5 REPORTING ON LOOPS 83
9.6 WEIGHT MODEL 85
9.7 TIME SERIES 86
9.8 SETTINGS 87
9.8.1 General Settings 87
9.8.2 Report Elements Settings 87
9.8.2.1 Excel Chart Component 88
9.8.2.2 First Impression Chart Component 90
9.8.2.3 PowerPoint Options 90
9.8.2.4 Filter Options 91
9.8.2.5 Answer Mask/Scale Mask 91
9.8.2.6 Crosstab Options 92
9.8.2.7 Loop Options 93
9.9 FILTERS 94
9.9.1 Defining Filters 94
9.9.2 Applying Filters 94
9.10 REPORT PUBLISHER 95
9.10.1 Publishing a Report 95
9.10.2 Changing a Report after Publishing 97
9.10.3 Publishing Different Data from One Survey to Separate Target Groups, or Same Report in Different Languages 98
9.11 MATRIX VIEWS 98
9.11.1 Introduction 98
9.11.2 Configure Matrix Views 98
9.11.3 Using Matrix Views 99
10 Recoding Data 102
10.1 EXPRESSION SYNTAX AND EVALUATION 102
10.1.1 SQL 102
10.1.2 JScript 103
11 Exports of Data Files 105
11.1 TEMPLATE EDITOR 105
11.2 DATA EXPORT 107
11.3 SPSS EXPORT 109
11.4 MS WORD EXPORT 110
12 APPENDIX A: confirmit Script Reference 111
12.1 ACCESSING SURVEY VARIABLES 111
12.1.1 Variable Types 112
12.1.2 Basic Variable Objects 112
12.1.2.1 Coded Variables 113
12.1.2.2 Open Variables 113

12.1.2.3 Dichotomies 113
12.1.2.4 Compounds 113
12.1.3 The Set Interface 114
12.1.3.1 Basic Set Type 114
12.2 FORM VALIDATION 116
12.2.1 Adding Your Own Validation Code 116
12.2.1.1 Runtime Functions for Error Handling 116
12.2.1.2 Template based error messages 121
13 APPENDIX B: confirmit LANGUAGE CODES 125
14 APPENDIX C: RESERVED KEYWORDS 126

1 Introduction

1.1 What is confirmit?

Future Information Research Management (FIRM)'s vision is helping customers to drive intangible information into tangible knowledge.

The **confirmi**t software is FIRM's answer to that vision. **confirmit** is a comprehensive Web-based information retrieval and presenting software.

1.2 About this Manual

confirmit software is available either as a server installation or "log-in". They do not differ in a substantial way, but in the instances they do, "log-in" specifics are marked with a gray background like this:

Login only:

The "log-in"-version also includes CATI (Computer Assisted Telephone Interviewing) functionality. This functionality is described in a separate manual.

Throughout this manual we refer to different URLs, e.g. http://www.yourconfirmitserver.com/confirmit/firm.asp. You will have to substitute yourconfirmitserver.com with your actual server address, or www.confirmit.com for login users.

Please note that the terms "question" and "form" are to be interpreted in the same way.

1.2.1 Interview Management

The interview management module and more detailed information on its use will be available after the revision the functionality is undergoing at the moment.

1.3 confirmit Users

confirmit users may have either Advanced or Executive rights. Your **confirmit** administrator is entitled to define this user level.

Login only:

Contact support@confirmit.com to change user level.

1.3.1 Advanced Users

Advanced users have to acquire the most complete knowledge of the **confirmit** web application. They are typically responsible for these areas of a survey:

- initiating a new project
- programming questions and skip logic
- setting up project properties (permissions, public survey/limited survey etc)
- handling respondents in limited surveys
- compiling/generating interview files to make the project go "live"
- extend the library of interview layout templates
- extend the library of questions and forms
- create and publish reports

1.3.2 Executive Users

These users are set up with permissions to view and comment on questionnaires and response data. They may view and order generated reports. They may NOT program questionnaires or create reports themselves.

1.4 About Log-in Sessions

Whenever a **confirmit** user logs on to the application, a new *session* object is created on the server side. This object holds information about the user's *state* (current project etc). The web server allocates memory for this state information. By maintaining this information as the user moves from page to page, the individual requests are kept small and simple.

However, the web server does not want to keep track of this information when the **confirmit** user exits the web application. A user may exit the application without explicitly sending a "log off" command to the web server. If the time between user requests exceeds 20 minutes, an automatic *time out* occurs. **confirmit** assumes the user has logged off and frees the memory that holds information about the state. If a new request is sent after a time out, the user is redirected to the first page of the web application. State information like *current project* is lost.

1.5 Security Issues

Login:

1.5.1 Secure Sockets Layer (SSL)

Regardless of what many people say, the Internet is a secure communications medium. Network traffic travel from your organization's local network, through a network of lines leased from public telephone network providers, before reaching another locally controlled network at the other end of the communication stream. As long as you trust the server on the other end of connection, very few things can go wrong; the probability of someone listening in on your traffic is very small indeed.

FIRM offers you the possibility to communicate with **confirmit** over the Secure Sockets Layer (SSL), an encrypted protocol that further secures all traffic between you and the system.

1.5.2 Object Security

While the previous section deals with Internet security in general, this section deals with security *inside* the **confirmit** application.

A project, a question, a report– these are all *objects* inside **confirmit**'s powerful object oriented database. For each object, specific permissions can be set up, including *read* and *write* permissions.

When you initiate a new project (object) inside **confirmit**, the project is by default only visible to you, because no one else has read or write permissions. However, you may specify that other users or groups of users should have these permissions.

1.5.3 Respondent Security and Anonymity

Web surveys can be set up to run as *public surveys*, in which anybody who accidentally tries to access the web interview page may participate.

confirmit also supports *limited surveys*, in which a limited number of respondents are able to log on to the survey. In these surveys, a set of respondent background data may be merged with the response data to produce the final data file. It is the responsibility of the project administrator to make sure that this information is not misused in any way. In theory, the merging may take place without the respondent knowing it. The good thing about the merging is that it makes it possible to produce *respondent specific questionnaires*. For instance, a web panelist does not have to enter information about age or gender simply because this information already exists in his or her profile.

1.5.4 Back up of Meta Data

Login only:

By this we mean all the data of a project that is *not* response data. Every night, a back up copy of all this meta data is saved to external media and stored in a safe. The data is stored in a safe place, and may be restored should a system crash occur.

For server installations, your company has to set up routines for backup.

1.5.5 Back up of Response Data

It is possible to delete response data from the web application. You are always asked twice about doing this ("Are you 100% sure?"), but it *has* happened that a **confirmit** user has accidentally deleted the response data of a running project.

It is the responsibility of the **confirmit** user to make sure that this does not happen.

Login only:

However, the MS SQL server 7.0 periodically runs full back up of all response data in the database. In most cases it is possible to restore most of the data from a recent back up. In the case of a system failure, external media back ups of response data also exists.

For server installations, your company has to set up routines for backup.

1.6 System Requirements

1.6.1 **confirmit** Users

CONFIRMIT is designed to function correctly in Microsoft Internet Explorer 4.0 or 5.0 in Microsoft Windows™ 95/98/NT/2000. However, there may be bugs in local (non-English) versions of Internet Explorer 4.0 that have been resolved in later versions of this browser, and these bugs may impact the operation of **confirmit**. We therefore recommend our customers to apply the latest Microsoft Internet Explorer patch for their country. (Internet Explorer 5.0 or Internet Explorer 4.01 Service Pack 2).

1.6.2 Respondent Requirement

The web survey participants are presented with an easy to use, standard HTML page for entering data into **confirmit**. The survey itself requires no knowledge of **confirmit** whatsoever. Participants are directed to a survey through links on other web sites, or through email invitations sent directly to them.

All interview files support the HTML 2.0 standard, which means that older versions of Netscape and MS Internet Explorer are supported.

1.6.3 3rd Party Downloads

The **confirmit** application does not rely heavily on 3rd party components. Currently three downloadable components are integrated:

Area	Type	Company	See also
Reporting	Chart component	Visual components	http://www.visualcomponents.com
Questionnaire design	Grid component	VideoSoft	http://www.videosoft.com/
Reporting	Excel chart component	Microsoft	http://www.microsoft.com

Table 1: 3rd party downloads

2 Entering confirmit – a General Overview

2.1 confirmit Login

Users log in to CONFIRMIT from this URL:

http://www.yourconfirmitserver.com/confirmit/firm.asp

Login only: You can also log in to **confirmit** by choosing "Login" on FIRM's home page, http://www.confirmit.com/.

Accessing this link will make your browser request a user name and a password, see picture. Enter your personal user name and password, and you will be logged on. If the browser who tries to log on to **confirmit** does not meet the browser requirements, an error message will display.

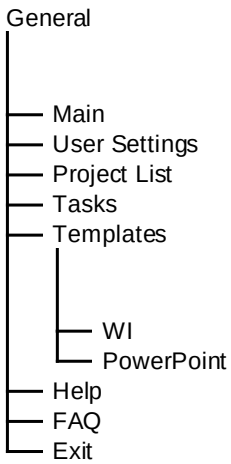
Figure 1: Login box

2.2 confirmit Menu Structure

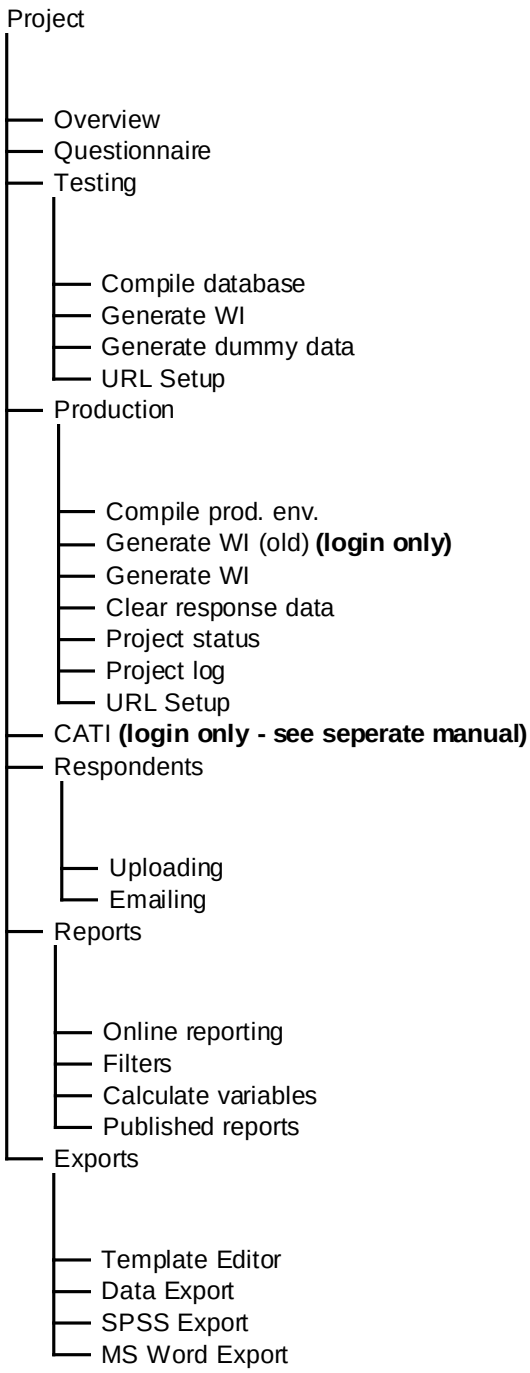
When you have logged in to **confirmit**, you will be brought to the first page of **confirmit**. On the left-hand margin you will find a menu structure that is divided into to main sections: *General* and *Project*. These menus navigate you through both, the main and project specific, features of **confirmit**.

The following chapters offer a more detailed description of every choice in the menu.

Figure 2: confirmit menu



This and the following figure offer you an overview of all the menu links with their subsections.


Project
Overview
Questionnaire
Testing
Compile database
Generate WI
Generate dummy data
URL Setup
Production
Compile prod. env.
Generate WI (old) **(login only)**
Generate WI
Clear response data
Project status
Project log
URL Setup
CATI **(login only - see seperate manual)**
Respondents
Uploading
Emailing
Reports
Online reporting
Filters
Calculate variables
Published reports
Exports
Template Editor
Data Export
SPSS Export
MS Word Export

3 General

This section describes the functionality under “General” in the menu.

3.1 Main Page

The first page of the application indicates some natural starting points for your session.

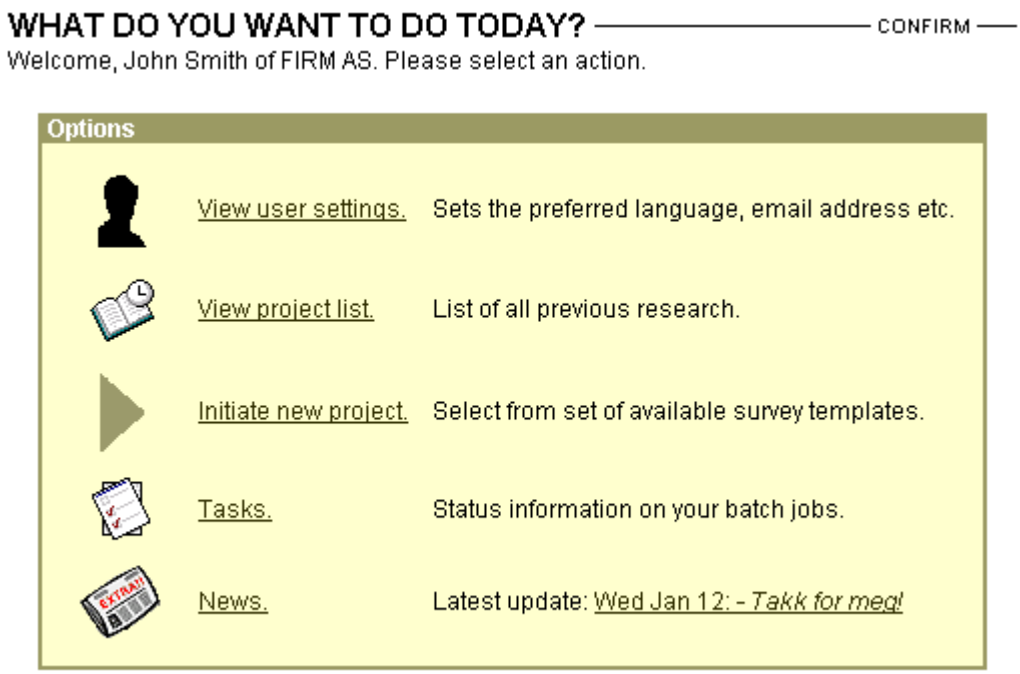


Figure 3: Welcome page

3.2 User Settings

Each user has a profile set up at login. The profile holds information about the user's name, email address, preferred language, screen resolution.

The user may change his/her user settings at any time.

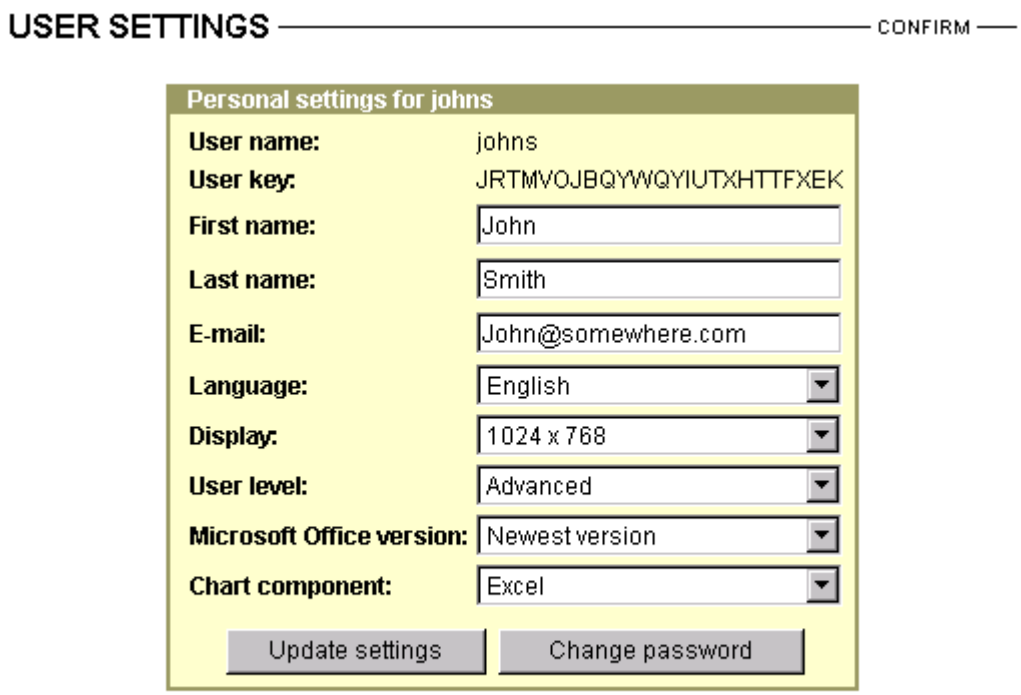


Figure 4: User settings page

Setting	Changeable	Description
User name	-	User login name. In combination with a password, this user name will log the user on to the **confirmit** application.
User key	-	Unique to each user. To make yourself "visible" to **confirmit** users outside your organization, you'll have to supply them with your user key. (This is really a security issue.)
First name, last name	X	Name of the user. Used mainly for interaction with other users.
Language	X	Preferred language of the user. This is the language used when previewing questions and reports.
Display	X	Use this for optimizing your screen resolution with the **confirmit** software. After having made changes to your display settings, refresh the browser (F5).
Email	X	Email address of user. This address is used by the **confirmit** message system to send messages to the **confirmit** users. This is also the default email address where requested report and export files are sent.
User level	(X)	Some users are set up with permissions to change their user level between Advanced and Executive. See 1.3.
Microsoft Office Version	X	Users can choose what Office version to use for PPT/Excel exports from their user settings. By selecting 'Newest version', the newest version installed on the export server will always be used. Currently this is Office 97. Excel exports will never be made in Excel 4.0 and older, though, since these old versions do not support more than one worksheet for each workbook.
Chart Component	X	Use this to choose between First Impression or Excel components in Online Reporting. Excel component is recommended, especially if you are planning to publish the report (see 9.10 Report Publisher).

Table 2: User settings fields

3.3 Project List

The project list is the main project organizer. It consists of *Favorite projects*, i.e. projects you have been working on recently. The list may be sorted by clicking the header of the column by which you want to sort the projects.

The list may be filtered in a number of ways, by *time* (*created* column) or by *search parameters* (all other columns). You only have to enter the first letters of the criteria on which the search is to be based. The system will return all matches. Example: Specifying "te" in the *name* column will return all projects starting with the letters "te".

If you want all projects to be listed, enter a '*' in the *name* column under *search projects*.

You may enter a project directly from this list by clicking the project name or the project ID.

When the list of your Favorite projects contains a considerable number of projects, you may want to remove the ones that you currently do not work on by clicking the *Rem* (=remove) link. This will not delete the projects, but move them from the Favorite list.

In case you wish to access the removed projects, you can use the *Search projects* fields. All projects the user has been granted access to will be listed according to search criteria. You may move these projects to your Favorite list by clicking the *Add* link.

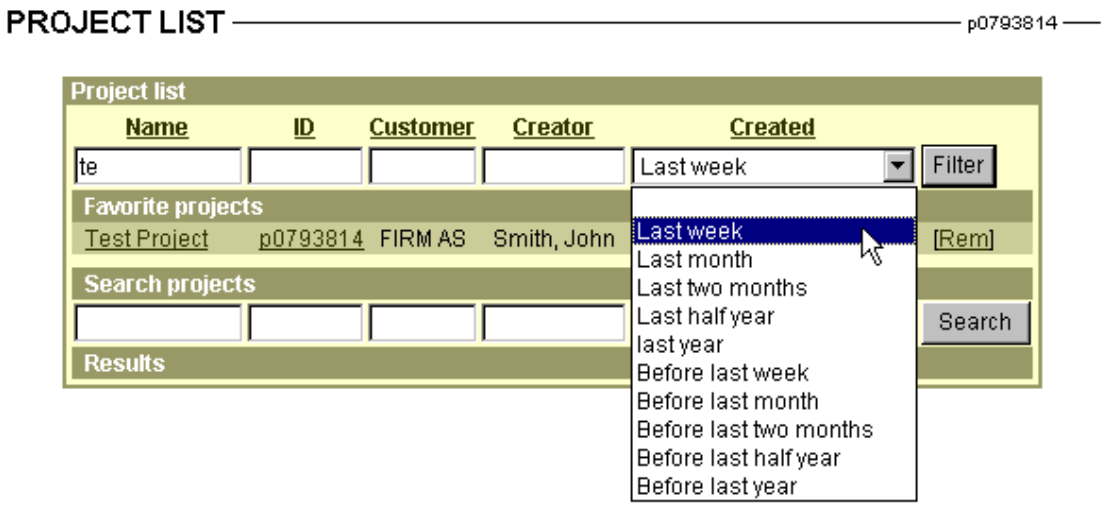


Figure 5: Filtered project list

3.4 Task Management

Task management lets you track the status of your **confirmit** batch jobs, including MS PowerPoint exports, SPSS exports etc.

When placing requests for exports (i.e. SPSS), users may select a starting time for the execution of the task: Immediately, at midnight or at a specified hour.

The task management page has five tabs:

1. In queue: A list of the user's pending tasks.
2. Executing tasks: Task currently being run at the batch server.
3. Killed tasks: Tasks that for some reason did not finish within the specified timeframe.
4. Completed tasks: Tasks that did finish within the specified timeframe. Does not guarantee that they are successful. If not, users should get an error notification in their inbox.
5. Cancelled tasks: Tasks that have been cancelled by the user before execution.

Figure 6: Task management

NB: confirmit sets a time limitation for how long a task can run before it is killed. This is done to avoid unnecessary strain on the server if there is an error with the file. However, this may also result in heavy tasks (e.g. based on long questionnaires) being killed before they are completed. Ask your **confirmit** administrator to manually adjust the time limitation, should it become necessary.

Login only:

Contact FIRM Customer Relations and Support for help

3.5 News Page

Click *News* to learn about the latest updates to **confirmit**. The page is divided into six separate categories. The most recent changes are to be found on top. The lists are searchable. Click the subject links or the red arrows for details.

Figure 7: News

3.6 Templates

From the *General* menu, a *Templates* link leads to two subsections, MS PowerPoint templates and Web Interview (WI) templates. **confirmit** templates are always bound to a specific organization.

3.6.1 Web Interview Templates

Web interview templates define the layout of web interview files and published reports. Currently a WI template is defined by

- name and description
- page body (i.e. background image or background color)
- page header and footer
- font attributes
- header/body/footer of individual questions

It is possible to add web interview templates by duplicating and editing existing ones. This requires some HTML knowledge. It is important that you push the Save button after having made changes *before* you go to the Preview.

The parameters you can use in the templates have to be prefixed and suffixed by caret (^):

SNAME Survey name

HELP Help link

LOGO Your company logo

NAVCTL Navigation bar (back/fwd)

FTITLE Form title

FNAME Form name

FTEXT Form text

FCOMMENT Form instruction

INPUTS Form inputs

QNUM Question number

3.6.1.1 The Fields in the WI Template Edit Modus

Name: The name of the template

Body attributes: Controls the design of the whole page: background colors, picture (background picture), margins etc.

Error font attributes: Controls the font size/color of the error messages in a WEB survey.

Grid color: Controls the color of the gridlines in a grid question.

Horizontal label font attributes: Controls the "scale" text in a grid question.

Vertical label font attributes: Controls the "answers" text in all questions.

Page header: This field controls the layout of the top of the pages. You may e.g. insert and edit logos and the name of the survey in this field, e.g. ^LOGO^, ^SNAME^.

Page footer: Controls the bottom of the pages, e.g. company's name and the navigation buttons, e.g. ^NAVCTL^.

Information form header: Controls the design of the upper part of an information form, both background color and text (font size, color etc.), e.g. ^FTITLE^.

Information form footer: Controls the area right under the information form (similar to Page footer).

Information form body: Controls the design of the information form body, background color, text of the information form, and comments, e.g. ^FTEXT^, ^FCOMMENT^.

Data form header: Controls the design of the upper part of an open, grid, single, multi question form; background color and text (font size, color, etc.), e.g. ^FTITLE^.

Data form footer: Controls the area right under the question (similar to Page footer).

Data form body: Controls the designs of the body of the grid, single, multi question form; background color, text of the question (^FTEXT^), as well as comments (^FCOMMENT^) and answer alternatives (^INPUTS^).

Progress bar font attributes: Controls the design/font size of the percent numbers under the bar.

Progress bar background color: Controls the background of the progress bar.

Progress bar image: Controls the moving progress bar that shows the progress of the survey.

3.6.1.2 Published Report Templates

Last four fields at the bottom of the WI Template Editor control the layout of Published Reports (See 9.10). No preview of these templates is available at the moment. In order to see the edited template, you will have to publish the report.

Report header: This field controls the layout of the top of the pages. You may e.g. insert and edit logos and the name of the survey in this field.

Report footer: Controls the layout of the bottom of the pages, e.g. company's name.

Report light/dark color: The index and tables in the published reports are by default of the light and the dark green colors that are used in **confirmit**. If you wish to apply other colors to your report, you may specify HTML color codes in these fields.

There are three parameters that are used only in report templates:

RNAME Report name
RGEN Report generation time
RPER Report data period

3.6.2 MS PowerPoint Templates

Predefined MS PowerPoint templates may be uploaded to **confirmit**. All templates uploaded are visible to **confirmit** users within the same organization. A user may select from these templates when ordering MS PowerPoint presentations. Extension: .pot.

3.7 Help

In the General menu you find a Help button. When one clicks on the button, a new browser window named **confirmit** Help opens offering **confirmit** users four links with various help information:

A **Quick card** is a one-page text based help file dedicated to a certain topic/area in **confirmit**. It has short explanations of all functions available within the topic/area.

Step-By-Step include demonstrations with screen shots and text from all the main areas in **confirmit**.

User Manual is identical to this paper edition .The document is indexed, searchable and has internal linking (e.g. by clicking on a topic in Table of Contents, you get directly to the topic).

3.8 FAQ

FAQ or Frequently Asked Questions is a repository of the most commonly asked questions and answers concerning the use of **confirmit**.

Support link gives **confirmit** users information about how FIRM's Support service can be contacted.

The Quick Cards and The User Manual are in PDF format and require Adobe Acrobat Reader plug-in installed in your browser. This can be downloaded free of charge from the Adobe's homepage by clicking on the icon on screen.

3.9 Exiting CONFIRMIT

Log out of **confirmit** by clicking *Exit* in the left margin. To prevent other users from accessing **confirmit**, please close the browser e.g. by clicking the *Exit Internet Explorer*-button.

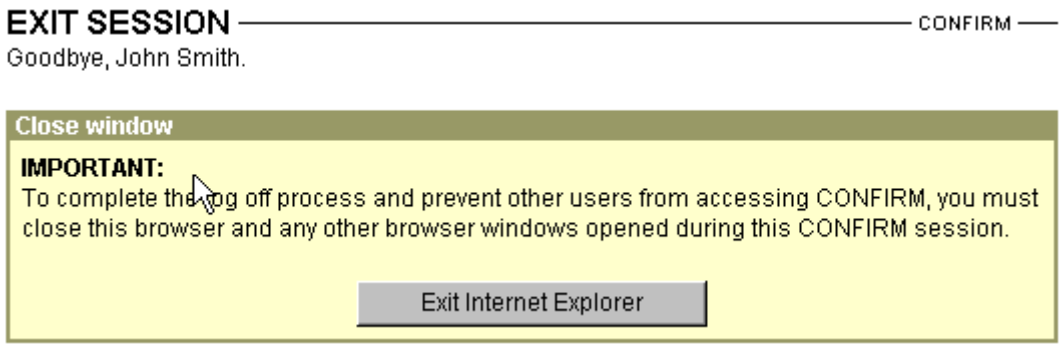


Figure 8: Exiting confirmit

4 Project Overview

A new blank project may be initiated by clicking, *Initiate new project* from the main page, and then clicking *New.* You can also choose whole questionnaire templates from this page (for more information on this subject see 5.6.1Templates in Library). This will automatically take you to the *Project list.* Your initiated project will be positioned at the very bottom of the project list, named *New blank project.* Click on it in order to come to the *Overview* page of this project.

This is the page you are brought to when you click on existing projects in the project list.

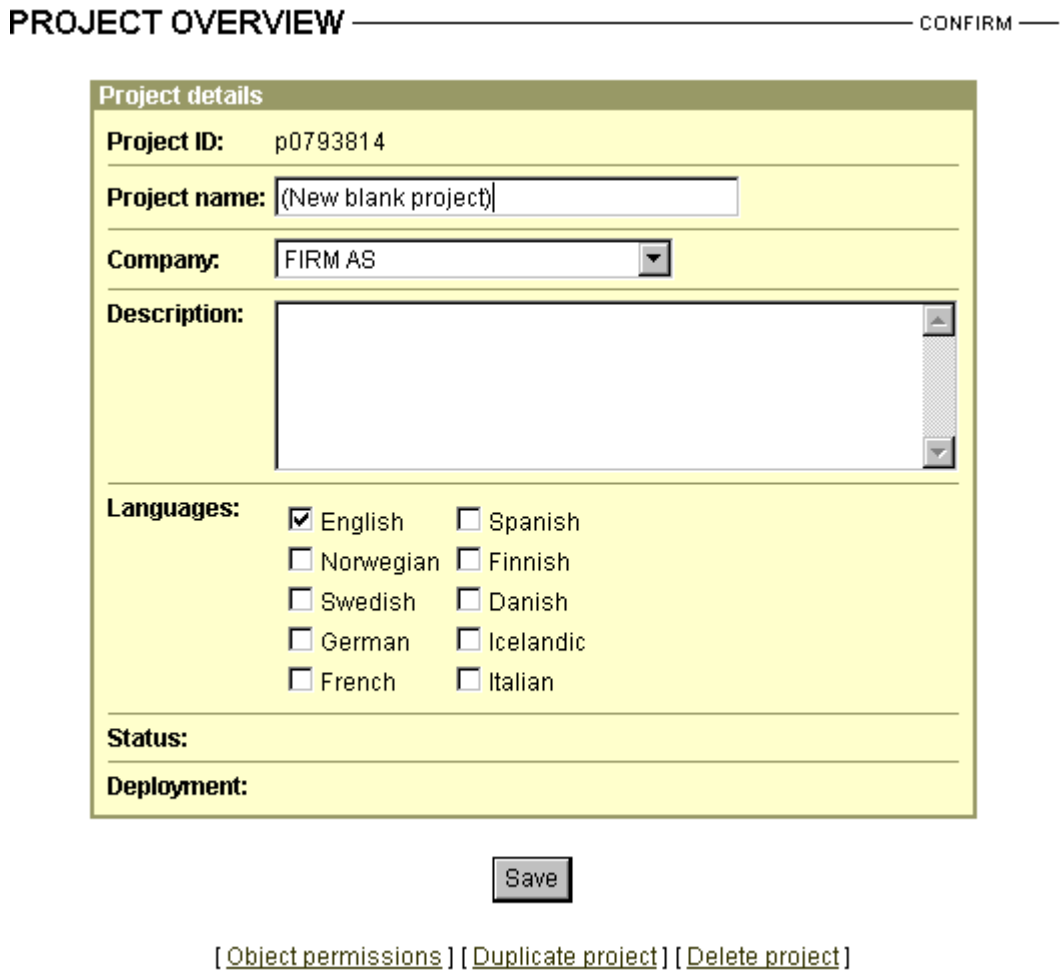


Figure 9: Project overview page

Field	Description
Project ID	System generated unique identification number.
Project name	A title you choose for the project. Be aware that this will be the title of the browser window in a live WI survey.
Company	Your company's name.
Description	Here you may enter a more detailed description of the project.
Languages	Working language(s) of the project.
Status	Project status. (See 7.3)
Deployment	Survey type (Public/Limited/Pop-Up) Web Interview. **Login only:** CATI

Table 3: Project overview

4.1 Object Permissions

When you click on the link *Object permissions*, you are brought to the following page:

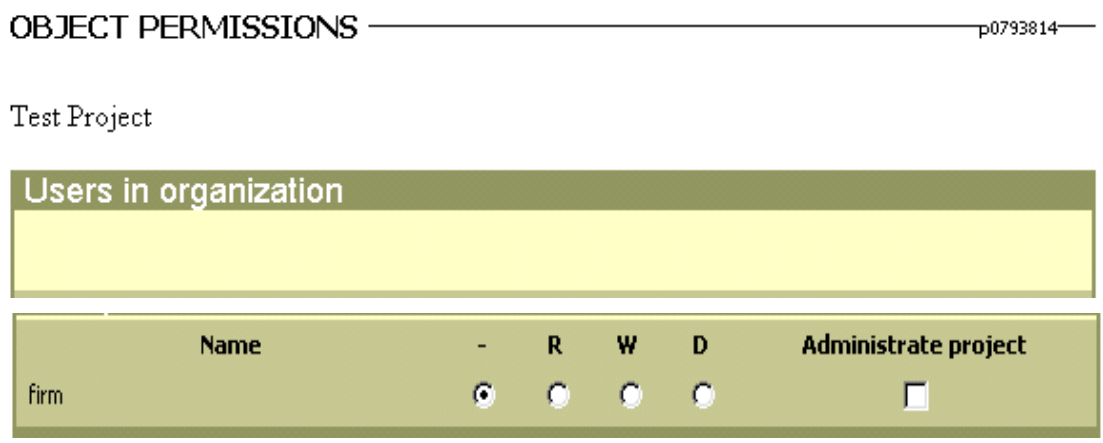


Figure 10: Object Permissions

Field	Description
Name	The userid and the full name of the user (if entered in **confirmit**) will appear in this field.
-	The user does not have access to the project.
R	The user has only Read permission, i.e. he/she is not allowed to add new or delete existing elements in the questionnaire.
W	The user has Write permission, i.e. he/she is allowed to add questions to the questionnaire.
D	The user has Delete permission, i.e. he/she is fully allowed to work on the questionnaire including the permission to delete it.
Administrate project	The user is the administrator and owner of the project. He/she may alter the questionnaire, set the project live, check response status, etc. This is usually the person who initiates the project.

Table 4: Object permissions

For all projects it is true that the user who initiates the project is the project administrator and owner. A project is initially invisible to all other **confirmit** users.

All **confirmit** users in your organization will be listed in *Object permissions*. The project administrator may set up other **confirmit** accounts with permissions to view the new project.

Login only:

Personnel at FIRM may be granted access for support purposes.

Initially, only users and groups within your organization are listed on the “object permissions” page. However, more users may be added by supplying **confirmit** with correct user keys of these users. Typically, an external user will send you his or her user key by email; this key is then copied and pasted into **confirmit** by clicking the *Add users* button at the bottom of the *Object permissions* page.

4.2 Duplicate Project

It is possible to duplicate whole projects in **confirmit** by clicking this link under *Project Overview*. The system duplicates the questionnaire structure with both skip-logics and, if used, advanced scripting. It duplicates the constructed reports as well.

You may afterwards apply changes to the questionnaire.

The data from the previous survey are not duplicated to the copy of the project.

4.3 Delete Project

You may delete projects by clicking this link under *Project Overview*.

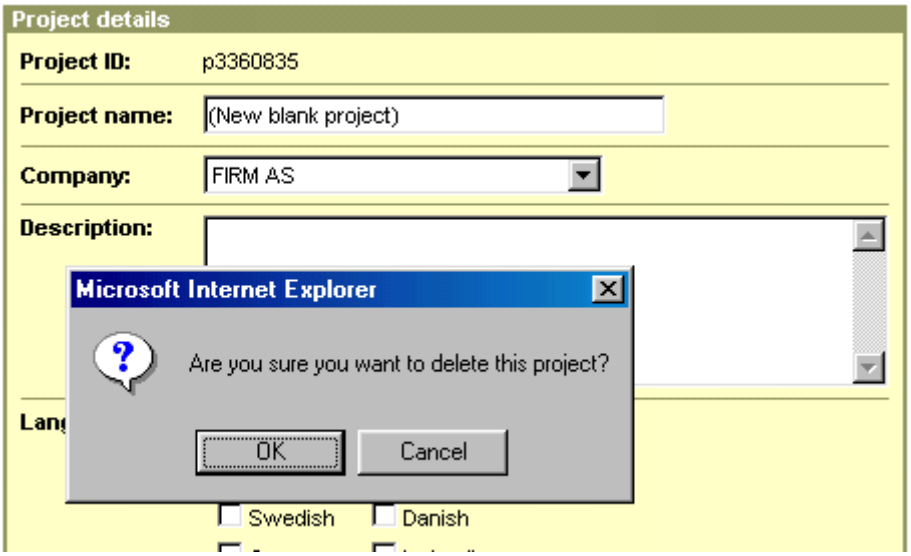


Figure 11: Deleting a project

Before actually deleting, **confirmit** will generate the message above, warning you that you are about to delete a project.

5 Building Questionnaires

5.1 Creating a New Questionnaire

A project's questionnaire is created as a list of consecutive objects. Together they define the routing of the survey. You access the questionnaire by clicking on the menu item *Questionnaire*. An empty questionnaire will look like ***Figure 12***. Note the three folders, *Routing*, *Scales and lists* and *New objects*.

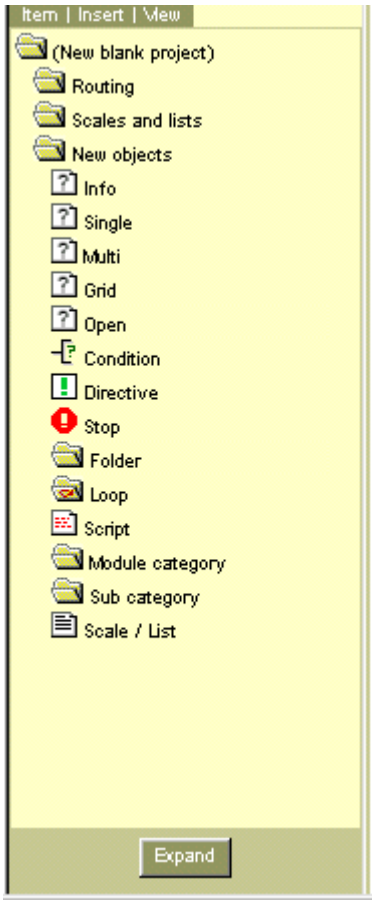


Figure 12: Questionnaire view

5.2 Adding New Objects to the Questionnaire

You design your questionnaire by dragging objects from *the New objects* folder, and dropping them in the *Routing* folder. For instance, dragging and dropping a *Single* object from *New objects* into the *Routing* folder, will insert a new single punch form as the first question in the routing. Another way of inserting an object is to select the preferred destination of the object, then select the corresponding object-type from the *Insert* menu.

After you have inserted an object into the routing, it will appear in the routing as a new icon. Double click the object for editing. Below you will find a description of the different available routing-objects.

5.2.1 Forms

There are currently five standard types of forms available:

Form type	Description
Info	Used for information pages in the survey.
Single	Single-punch question. The respondent can select one answer from a list.
Multi	Multi-punch question. The respondent can select many answers from a list.
Grid	A question type that lets the respondent rate a list along a scale.
Open	An open-ended question.

***Table 5:* Standard form types**

To create and edit a form, insert it into the routing, and then double click it. This will bring up a *details view* of the form on the right hand side of the routing. This view consists of several separate pages, described below. Click the buttons along the top of the frame to access the different pages. Information is saved whenever you move to a new page. Not all pages are available for all form types.

5.2.1.1 Text

On this page, you supply the text of the form, along with other related information. If you work in two or more languages at the same time, there will be a separate set of fields for each active language (see 5.2.1.7)

Field	Description
Title	The title of the form. The title appears in the routing-tree. It also functions as a heading for the form when used in a web-survey, and as a descriptive label for the form when used in online reporting. The length of the title can be maximum 255 characters. When a title is not provided, Text will be used as a title. (In templates, you may use the primitive ^FTITLE^ where you want to insert this text. See 3.6.1.1)
Text	The form text. (In templates, you may use the primitive ^FTEXT^ where you want to insert this text. See 3.6.1.1)
Instruction	Use this field to give descriptive instructions or comments. (In templates, you may use the primitive ^FCOMMENT^ where you want to insert this text. See 3.6.1.1)
Scratchpad	See 5.4

Table 6: Text page

5.2.1.2 Answers

[Not available for Info-objects and Open-objects]

Use this page to define the list of answers, or categories, for a question. The main characteristics of an answer are its *text*, which is what the respondent or interviewer will see on the screen, and its *precode*, which is the underlying code stored in the database.

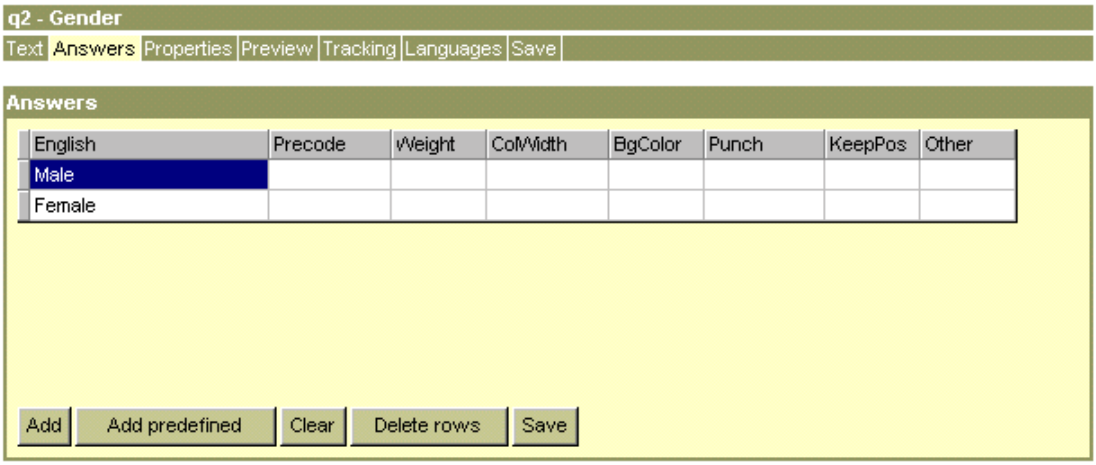


Figure 13: Answers page

The list (see figure) consists of one row for each answer or category, and each row consists of the following columns:

Field	**Description**
Language fields	The first columns in the list contain the text of the answer. There is one column for each active language (see 5.2.1.7). This makes it easy to translate between several languages.
Precode	Defines the precode of the answer. If a precode is not supplied, the precode will be equal to the position in the list, starting from 1. If you change the answer list while a survey is running, make sure that the precodes follow the original answers. Precodes should only contain alphanumeric characters, with no white space. The maximum length is 32 characters.
ColWidth	[Only relevant for grids] Defines the specific minimum width, in pixels, of a scale-element in grids. Applied primarily to eliminate width-differences between scale-elements, so to create an unbiased scale.
Weight	A numeric value for the answer. Use this field to define the individual value/weight of each category. You need to define values/weights to be able to calculate averages for the question in Online reporting. The average will only be calculated from answers with assigned values/weights. If you e.g. have “Do not know” as a possible answer, but want the averages to be calculated only from the other answers, you may leave the value/weight of “Do not know” blank.
Punch	[Applies to Multi questions only] Gives you the ability to define one or more elements in a multi-punch list as single-punch (e.g.: “None of the above”).
Background color property	This property makes it possible to have color variations inside a grid, i.e. a red to green scale. The list elements take a standard color string as input. Example: White may be specified as "white" or "#FFFFFF". A list of color codes is to be found under ***Help/Quick cards/HTML Colors***

KeepPos	Set this property to “yes” if you want a variable keep its position when randomizing a list (e.g. “*none of the above*” always last in a list).
Other	Set this property to "yes" if you want an open-ended text box at the end. It is not possible to combine dropdown option (see 5.2.1.4 Properties) and Other in Grid questions. The Other option overrides the Dropdown option. If you combine Dropdown and Other in Single questions, the text box will not appear when Other is chosen. If you have checked off this option for several elements of a Grid question, these elements will automatically be marked as Not required (see 5.2.1.4 Properties), i.e. it will be sufficient to answer the other elements in the Grid.

Table 7: Answers page

When the question is newly created, the list of answers will be empty. Press the *Add*-button to create a new answer in the list. Click inside the first column in the list. This will activate the first column. When you start typing, the active column will be put in editing mode, and a cursor will appear. If there is previous text in the column, your new text will be appended to the end.

When in editing mode, you can move to the next field by using the tab-button, or by clicking in the column with the mouse. You can directly move to an adjacent row by using the up- and down-arrow buttons. If you are at the bottom of the list, the down-button will create a new row.

When you are not in editing mode (no cursor), you may use the left- and right-buttons to move horizontally around in the list. Press <F2> to enter editing mode.

You can copy rows from the list by pressing <CTRL-C>, and pasting rows by pressing <CTRL-V>. The checkbox *Only copy active button* decides whether you copy and entire row or not, and the checkbox *Overwrite* decides whether you insert new rows when you paste or not.

The button *Add predefined* allows you to add a predefined list or scale to the answer list (see 5.5 Lists). This is very powerful when reusing for instance brand lists or scales in the survey. In a predefined row, the first column will become a dropdown-list when activated, giving you the opportunity to choose between the different predefined scales and lists.

The scratchpad is explained in 5.4 Scratchpad.

5.2.1.3 Scale

[Only available for grids]

This page defines the *scale* dimension of a grid. The page is functionally equal to the *Answers* page.

5.2.1.4 Properties

This page lets you define several aspects of the form-object. In brackets we specify which kind of questions the properties apply for.

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>Question ID</td><td>The project-unique ID of the form (questions). An alphanumeric string with no white space, starting with a letter. This is automatically generated.

The data are stored in the database with variable names constructed in the following fashion:

<table>
<tr><td>Single</td><td>One column with a tag identical to question id, and with answer precodes as contents</td></tr>
<tr><td>Open</td><td>Similar to that of Single, but the answer text is the content.</td></tr>
<tr><td>Multi</td><td>A column per answer alternative. Tag is equal to: question id_precode of the answer alternative. Contents are either 0 (not selected) or 1 (selected).</td></tr>
<tr><td>Grid</td><td>A column for each sub question (answers). Tag is equal to: question id_precode of the answer alternative. Contents are the values answers have in Scales.</td></tr>
<tr><td>Other</td><td>“Other: specify” in single, multi or grid questions is represented in data files in the following way: Tag is equal to question id_precode_other. Contents will be the answer text.</td></tr>
</table>

NB: You do not have to change the default ID of a form, even if you introduce the form between other existing forms (e.g. at a late stage of the questionnaire construction). If you anyway do change the ID, be very careful to give it a unique ID! So if you introduce a new form between forms 5 and 6, being 5 a multiple question, do not label the new one 5_2, as this is the code for the second alternative in the original question 5!

These variable names will be used as column titles when survey data is exported to e.g. SPSS or Excel.

Constrain length of ID to 8 characters if you want to use the ID as column-name in SPSS when exporting data to this format. For multi-questions and grids, you should try to constrain length to 5 or 6 characters, because column names are constructed like this: question id_precode.</td></tr>
<tr><td>Precode mask</td><td>If you want to dynamically exclude one or more of the answers in the answer list, use this field to create a JScript-expression resulting in a set of precodes to include. See Section 12 for a description of expressions.</td></tr>
<tr><td>Field width</td><td>[All] Number of characters assigned to this question in the database and when exporting to fixed width ASCII files.

In one-row text boxes, it will not be possible to enter more characters than specified in Field width. If the text box has several rows, an error message will be generated, and the answer will be truncated.</td></tr>
<tr><td>Rows/Columns</td><td>[Open, Multi] You can control the size of Open question boxes. The open-ended questions may appear as text areas or as single line input boxes. The "open text" object has got two properties: "Rows" and "Cols". Rows = Height (number of lines), Cols = Width (number of characters). To produce a single line input, set "Rows" to 1.

In Multi questions with the OpenText option, you can specify the field width in Columns (the default is 30 characters). The Columns option does not appear until you select OpenText and after you save.</td></tr>
<tr><td>List Rows/Columns</td><td>[Single, Multi, Grids] Divides the answer list in several columns and/or rows. If you specify number of rows only, the answers will be distributed evenly across the columns and vice versa.

These options have no effect on Single dropdown questions.

If you have specified both options, Columns will be preferred as an option in cases where the number of elements is higher than specified.</td></tr>
</table>

	In Grids, if you have chosen Dropdown as an option, List Rows/Columns may be used to specify the number of rows and columns for the dropdowns.
Numeric	[Open, Multi with OpenText selected] If “Numeric” is selected and you click save, then input fields for “Precision” (integer 0<p<=28. Specifies the maximum total number of decimal digits that can be stored, both to the left and to the right of the decimal point), “Scale” (integer 0<=s<=p. Specifies the maximum number of decimal digits that can be stored to the right of the decimal point.), “Lower limit” (float) and “Upper Limit” (float) are available. For “Lower limit”, there is a dropdown containing the choices “>” and “>=”. For “Upper limit”, there is a dropdown containing “<” and “<=”. “Field width” is now not available. For a Multi with the OpenText selected, the same “Precision”, “Scale”, “Lower limit” and “Upper Limit” will be applied for all fields. Recoded variables are currently not supported.
Validation code	For unlimited flexibility in question-validation, you can write JScript-code to validate your answers. See 6 **confirmit** Scripts.

Table 8: Form properties

Flag	**Description**
Randomize	[Available in Single, Multi, Grid] When checked, the list of answers will be randomized.
Recoded var	[Single, Multi, Open] Turns this form into a recoded variable, which will not show up in the interview. You can predefine complex recoding rules in SQL or JScript (see 10 Recoding Data)
Background var	[Single, Multi, Open] Treat this form as a background variable. The form will not show up in the interview, but will fetch its value from the respondent list. Very useful when you have information on your respondents, which you want to apply in the skipping-logic of your questionnaire, or in the Online reporter. A column with a name equal to the form-ID is required in the respondent list.
Hidden	[Single, Multi, Open] A hidden form is hidden from the interviewer/respondent, but its value may be set on the fly from script-objects, making, e.g., “on-the-fly”-recoding possible.
Ordered	[Multi, Grid] Check this flag if you wish the generator to produce code that ensures that the inputs in Multi Ordered questions and Grid questions are unique and consecutive for each alternative in the list. In Multi questions, you can combine this choice with OpenText option. The data input boxes will be placed after the text.
OpenText	[Multi] This flag will turn a multi into a list of text-boxes, making it easy to create open-ended ratings.
Quota var	[Will be implemented later in WI. At present CATI only].
Dropdown	[Single, Grid] Singles and grids are normally represented as rows and columns of radio buttons. This flag will generate dropdown-boxes instead.
Not required	[All] Normally, all questions require an answer if you have checked off the option Answer Required in WI Generator (see 7.2.1). Checking this flag will turn off this validation for the particular question.

Deleted	[All] When you delete an object from the routing, it is not permanently deleted, but the “deleted”-flag is set. If you turn on *Show deleted nodes* (see 5.3 Editing the Routing), you can undelete the object by unchecking this flag.
Password	[Open] Makes it possible to collect data from the screen in a password style. The letters will appear as **** on screen. You may then want some validation of the password, or screening: q1 (*the password question*) `IF (f('q1') == "secret_password") THEN ...` For organizations that want to do internal surveys but don't want to give out their employees' email addresses, this may be a solution. The survey password must then be distributed within the organization. Rows are always forced to 1, no matter what you specify in rows.

Table 9: Form flags

5.2.1.5 Preview

The *preview* page will give you an idea of the respondent’s view of the question. The question is displayed using a standard web-interview template. Some of the features are not shown in preview yet, though, e.g. Dropdown Grid, Background Color, Size of text boxes, etc.

5.2.1.6 Tracking

The tracking functionality allows you to trace where and when a particular question has been used. This applies to questions that are saved in **confirmit** Library and reused in several projects over time. Questions are traceable as long as you do not change their Question ID. You may change titles and texts, though.

5.2.1.7 Languages

In a multilingual project, you may not want to work with all project-languages at the same time. This page lets you select a subset of languages to work in. Check or uncheck languages, and select another page.

5.2.2 Conditions

Using condition-objects, you can create *skip-logic* in your questionnaire. To create and edit a condition, insert it into the routing, and double click on it. You insert conditions wherever you want to alter the flow of a survey based on a Boolean expression, for instance based on the answer of a previous question. If the expression evaluates to *true*, the survey-mechanism will execute the routing-objects beneath the *THEN*-node. If the expression does not evaluate to *true*, the objects beneath the *ELSE*-node will be executed, if the *ELSE*-node exists.

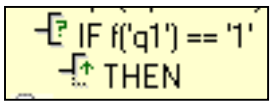


Figure 14: Conditional node

The expressions are written using the JScript-language. Expressions are described in Section 12 and to help building expressions, there is a c*ondition builder* (see 5.2.2.2 Condition Builder).

5.2.2.1 Description of Fields

Field	Description
Expression	A JScript-expression evaluating to *true* or *false*. Described in detail in 12 **confirmit** Scripts.
Branches	Select whether the condition should include an *ELSE*-branch or not.
Deleted	Used the same way as in forms (see 5.2.1.4 Properties)

Table 10: Conditional fields

5.2.2.2 Condition Builder

The condition builder helps you to build JScript expressions in no time.

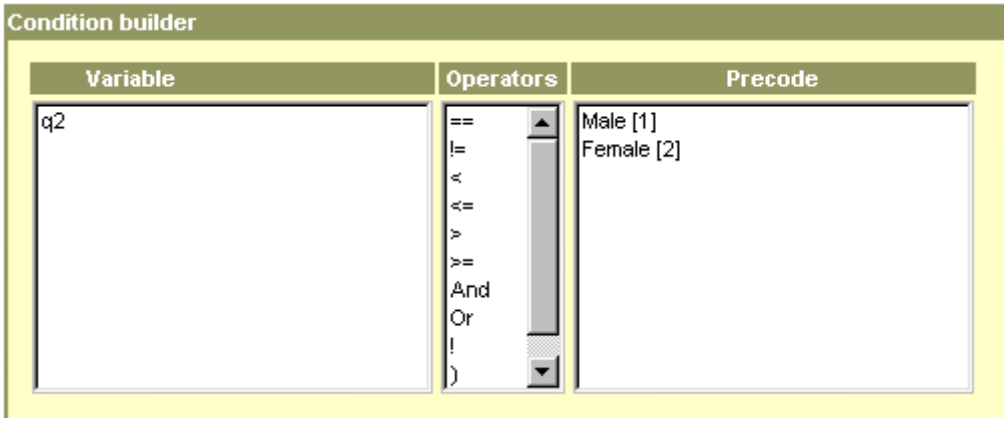


Figure 15: Condition builder

To use the builder, activate a condition-object by double clicking on it. **Single**-click on a form in the routing-tree you want to base your expression on (Double-clicking will activate form-edit mode). This will bring up a builder for the chosen form. The builder consists of three lists:

List	Description
Variable	This list will show the form-ID of the selected form. If the form is a grid, each of the answers in the grid (which are themselves single-punch questions) will be shown. Click on an item in the list to insert a reference to the form in the expression-field.
Operators and methods	This list includes most of the operators and methods you can apply to the form-object in focus. The list will be different for each type of form, because the various form-types support different methods. Click on an item in the list to insert the operator or method into the expression-field.
Precode	This list will show the answers of the selected form, with the precode in square brackets. Click on an item in the list to insert the precode into the expression-field.

Table 11: Condition builder

In combination, these lists make it easy to build complex expressions without remembering the JScript-syntax. You may at any time single-click another form in the routing to expand your expression to be based on more than one form.

Description	Symbol
Logical NOT	!
Less than	<
Greater than	>
Less than or equal to	<=
Greater than or equal to	>=
Equality	==
Inequality	!=
Logical AND	&&
Logical OR	||

Table 12: Single/grid operators

Description	Method/operator
Includes	`.inc(`
Union	`.union(`
Intersection	`.isect(`
Difference	`.diff(`
Logical AND	`&&`
Logical OR	`||`
Logical NOT	`!`
End parenthesis	`)`
Comma	`,`

Table 13: Multi operators/methods

Remember that each part in a condition has to be a complete statement.

Example: You have an age question with 4 answer alternatives. You want to include a condition which selects only respondents who have answered ‘1’ or ‘2’.

This is *not* the correct condition:

IF f(‘age’) == ‘1’ || ‘2’ THEN… Here ‘2’ is *not* a statement!

This is the *correct* syntax:

IF f(‘age’) == ‘1’ || f(‘age’) == ‘2’ THEN…

See 6 **confirmit** Scripts for a more thorough description of Jscript syntax in conditions.

5.2.3 Directives

Directives are instructions to the interview-generator that apply from where they are inserted in the routing and forward.

At the moment, there exist two directives:

Directive	Description
MULTIPLE_QUESTIONS	This directive will instruct the interview-generator to display the succeeding questions as a batch of questions on one page. There will be a break when a condition, precode mask or a new MULTIPLE_QUESTIONS- or SINGLE_QUESTIONS-directive is encountered. Example: If you want a web-survey to break for each ten forms, insert a MULTIPLE_QUESTIONS after every 10th form.
SINGLE_QUESTIONS	One question per page.

Table 14: Directives

5.2.4 Stop-objects

A Stop in the questionnaire will end the interview. This is mainly used for screening-purposes, and to end the interview various places in the routing. The status-code you set on the Stop-object is important:

Setting	Description
Complete	The web interview should be considered a complete interview. Status is set to complete and the end time of the interview is registered. Normally, online reporting will only show statistics for complete interviews.
Screened	Use this status-code to mark the interview as incomplete because of screening. Status is set to "screened" and end time is set.
Quota full	[Not in use yet]

Table 15: Stop node settings

5.2.5 Scripts

For very advanced routing, on-the-fly recoding, or other programming efforts, you can insert script-objects that will run your JScript-code when executed. When using script-objects, care must be taken to create codes that will run without errors. JScript is an interpreted programming language, which means that the code will only be checked at run-time. You should always perform thorough testing of questionnaires that include script-objects. (See 6 **confirmit** Scripts.)

5.2.6 Folders

To organize your questionnaire, create folders in the routing. For example, put all demographic-forms into one folder.

Field	Description
Name	The name of the folder (shows up in the routing-tree).
Description	A description of the folder.

Table 16: Folder fields

Folders make it easy to move a set of related routing-objects around in the questionnaire. Folders are also used in conjunction with the library (see 5.6 CONFIRMIT Library).

5.2.7 Loops

[**Login only:** CATI has different loop construct. See separate manual.]

A loop is a special type of folder, where the routing inside the loop will be repeated for a number of iterations. Let's say you want to repeat a block of questions for a list of car-brands. Instead of defining the same block of questions for each car-brand, only replacing the brand name, you can simply create a loop iterating over the list of cars, asking the same questions for each brand. You can also use nested loops (loops in loops).

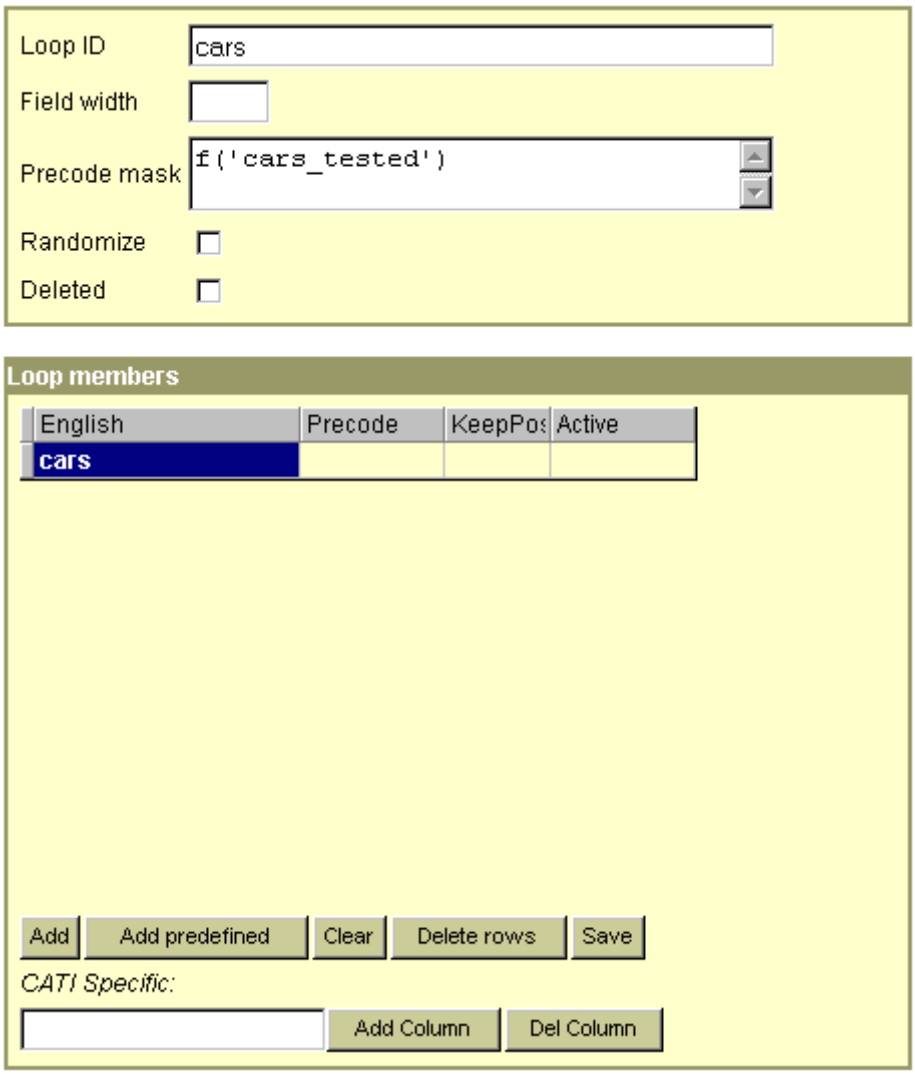


Figure 16: Loops

The initial definition of a loop consists of giving the loop a unique ID within the project, and creating a list of loop-iterations (created in the same way as you create an answer list in a form (see 5.2.1.2 Answers). The different parts of the loop-details view are:

- **Loop ID:** A unique ID of the loop within the project.
- **Field width: N**umber of characters assigned to the iteration precode in the database and when exporting to fixed width ASCII files.
- **Precode mask:** If you want to dynamically exclude one or more of the iterations in the list, use this field to create a JScript-expression resulting in a set of precodes to include. See APPENDIX A: **confirmit** Scripts for a description of expressions.
- **Randomized:** When checked, the list of iterations will be randomized.

- **Deleted:** When you delete a loop from the routing, it is not permanently deleted, but the "deleted"-flag is set. If you turn on *Show deleted nodes* (see 5.3 Editing the Routing), you can undelete the object by unchecking this flag.
- **Loop members:** A list of loop-iterations, edited the same way as the answer-list in forms. Definition of standard columns:

Column	Description
Language columns	The first columns in the list contain the text of the loop-iteration. There is one column for each active language (see 5.2.1.7 Languages).
Precode	Defines the precode of the iteration. If a precode is not supplied, the precode will be equal to the position in the list, starting from 1. Precodes should only contain alphanumeric characters, with no white space. The maximum length is 32 characters.
KeepPos	Set this property to "yes" if you want an item to keep its position when randomizing the list of iterations.
Active	Defines whether the iteration is active or not. For instance, in long-running trackers, you may want to ask the same set of questions once a week, but for different brands/concepts. By adding to the list, and activating/deactivating iterations, you can keep the same questionnaire.

Table 17: Standard loop columns

- **CATI Specific (login only):** See separate manual.

5.3 Editing the Routing

You reorganize the routing mainly by means of drag & drop in the routing-tree.

5.3.1 Moving an Object

To move an object (and its children, when moving folders, loops and conditions), left click the *text* of a routing object in the tree. This will make the object turn **blue**.

- Keep pressing the button, and start dragging. After moving the mouse a few pixels, a yellow drag-object will appear to the right of the mouse-pointer, giving you an indication of what you are dragging.
- At this point, releasing the mouse-button will drop your object on the object that is beneath the mouse-pointer. This destination object is always **red**.
- If you want to cancel the drop-operation while dragging, press the <ESC>-button. Do not just drop the object "out of sight".
- If the destination object is out of view because the routing is long, use the arrow-keys or PgUp/PgDown-keys to scroll up and down slow or fast. Keep the mouse-button pressed.
- When you drop the object, it will be moved to the new location, as long it is dropped on a legal destination. An example of an illegal destination is one of the objects in the *"New objects"*-folder.
- Dropping an object on top of another usually means to move the object to the position *behind* the destination object. If you drop an object on a folder-type object, it will end up *behind* the folder, not *inside*. **If you want the object to end up *inside* the folder, hold the <ALT>-key down before dropping. The object will end up as the first object in the folder.**

NB: If you should wish to change default form IDs, please read carefully the Question ID section in Table 8, 5.2.1.4 Properties.

5.3.2 Duplicating an Object

Duplicating behaves exactly like moving, but you create a copy of the source object. To duplicate, hold the <CTRL>-key down before dropping. An alternative is to select an object, and choosing *Duplicate* from the *Item* menu.

5.3.3 Selecting Multiple Objects

If you want to move or duplicate multiple objects, select the first object, hold <SHIFT> or <CTRL>, and press another object. <SHIFT> will multi-select consecutive objects, and <CTRL> will multi-select specific objects. To drag the selected objects, hold <SHIFT>, click the last selected object, and start dragging. You may release the <SHIFT>-button while dragging, and use the <ALT> and <CTRL> button to move inside or duplicate as described in 5.3.1 Moving an Object.

5.3.4 Deleting Objects

To delete one or more objects, select them, and press <DEL>. This will remove the selected objects from the routing, and set the *Deleted* flag on the objects to true.

You can undelete such objects by checking *Show deleted nodes* from the details-view of the top folder in the routing-tree (The root-folder with the name of the project. Double click to access details-view), and refreshing the tree. All deleted objects will appear as greyed-out in the routing-tree. Access the *Properties* page for the deleted object you want to recover (see 5.2.1.4 Properties), and uncheck the *Deleted* flag.

However, if the objects are already marked as deleted, and you delete the deleted objects, they are **permanently** deleted, and non-recoverable.

5.3.5 Pitfalls

Due to internet-browser limitations, some of the drag & drop operations may not be as intuitive as they are in normal Windows-based application:

- When picking up an object to drag, point to the **text** of a routing object, not the **icon**. You cannot drag the icon.
- If you drag too many objects, the drag-operation will fail with an error-message.
- After dropping an object, there will be a delay because of the round-trip to the server.

5.4 Scratchpad

The scratchpad is a text editor available in several screens. If you paste text into the scratchpad, for instance from a Word-file, you can easily copy and paste text into textboxes, grids, etc. The content of the scratchpad is stored locally in the web-browser, and is available as long as the web-browser is open, or until the session times out.

5.5 Lists

Quite often, one or more scales, or one or more brand-lists, product-lists, etc. are reused several times in a questionnaire. **confirmit** lets you predefine such lists, and reuse them in forms (see 5.2.1.2 Answers), or even in other lists (nested lists). Whenever you change the contents of a list, this change will be reflected in all lists that use the changed list.

To create a list, either drag a *Scale /List* object from the *New Objects* folder to the *Scales and lists* folder, or select *Scale /List* from the *Insert* menu. This will create a new list in the *Scales and lists* folder. Double-click the list to edit.

Field	Description
Name	A descriptive name for the list. The list-name is the text that shows up in the drop-down list when inserting the predefined-list in another form or list.
Answers	The answer-list behaves like the list in forms and loops. See 5.2.1.2 Answers for description of usage.
Scratchpad	See 5.4 Scratchpad.

Table 18: List fields

5.6 CONFIRMIT Library

The CONFIRMIT library makes it easy to reuse previously created questions, modules or even complete questionnaires including skip logic and reporting. The users may only access questions/modules for which they have at least R*ead* permissions.

1) Open the Questionnaire you are working on.

2) To enter the Library you go to the upper menu *View* > *Library*. Another frame beneath the questionnaire will open.

3) You go to the new objects folder in the questionnaire tree and drag a folder named *Module category* into the folder named *Modules* in the Library. You can double-click on it to give it a name.

4) You mark your module category, then you go to upper menu *Insert* > *Module* and a tree structure alike node will appear in your folder. You can have as many of them there as you wish.

5) When you double-click on the module, a third frame, looking exactly like the Questionnaire frame, will open. You double-click on the first folder, and the following window will open:

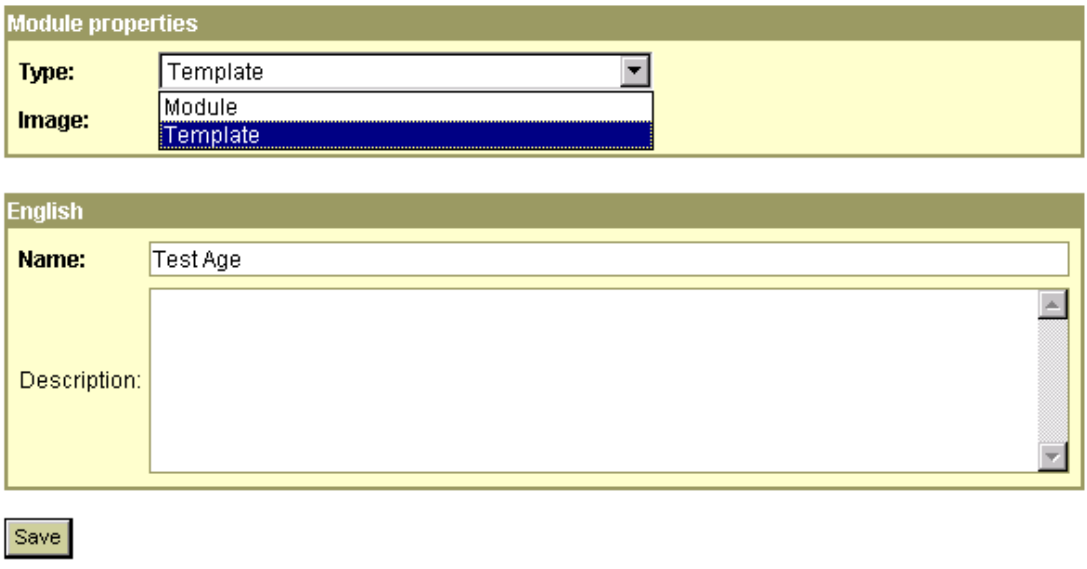


Figure 17: Module properties

Field	Description
Type	In this field you can choose whether you wish the question/questions/questionnaire in the module to be saved only as a module in the library, or also as a template in the

	the module to be saved only as a module in the library, or also as a template in the *Main menu > Initiate a new project.* See 5.6.1Templates in Library.
Image	In this field you can refer to an image that will appear as an icon on the *Initiate a new project* page next to the title of the template. You simply enter the URL of the image.
Name	In this field you enter the name of the folder, which will also be the title of the module.
Description	You can enter a more detailed description of the module.

Table 19: Module properties

In those modules you can simply drag and drop the questions or whole questionnaires you plan to reuse. You just drag the questions into the Routing folder.

The same procedure goes for Forms folder.

When you start working on a new project and wish to use some of the questions from the library, you simply open the library and the module where the particular question or questionnaire is stored, and by drag and drop method move them upwards to the questionnaire. The items are copied in this process, so a copy will always remain in the library.

5.6.1 Templates in Library

When you work on a module you wish to store in **confirmit** Library, you are given option to store it either as a module or as a template. Ordinary modules are accessible only through **confirmit** Library. If you store a module as template, it will be possible to access it through the Library, and as a link on the *Initiate New Project* page, as shown below. When you click a template on the Initiate New Project page, you will be brought straight to the Questionnaire builder containing the questions from the template you chose.

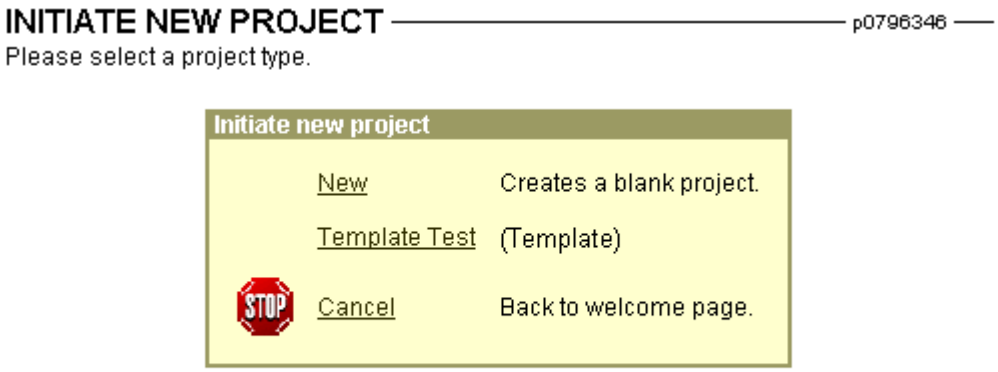


Figure 18: Templates in Initiate New Project

6 CONFIRMIT Scripts

6.1 Introduction

Most questionnaires contain some amount of script code. Scripts are used:

- in conditionals for controlling the flow through the questionnaire,
- for filtering the lists displayed in questions and the iterations in loops based on previous answers (precode masks)
- in form elements for text substitution (piping),
- for custom validation of user input and
- in general-purpose code contained in script nodes.

confirmit uses Microsoft's JScript[1] scripting engine to evaluate all questionnaire expressions and to execute scripts. The runtime environment of the interview engine (or simply *the runtime*) supplies a number of functions and objects that provide references to and let you manipulate survey variables.

JScript is an interpreted programming language, which means that the code will only be checked at run-time. So when you use scripts you should always test your questionnaires thoroughly before going "live".

This chapter describes briefly how to refer to and manipulate survey variables.

NB! The functions used here are not described in detail. They are described in the context they are used most often, (e.g. union() in precode masks section), but that does not mean that they are in any way restricted to being used in just that context.

For a more thorough reference manual to scripting in **confirmit**, see Appendix A where the functions are described in detail.

6.2 Accessing Survey Variables – the f Function

The f function is used to access survey variable values, i.e. the answers given to a question. It will return values or sets depending of the type of variable it is used on. It will return *null* if the question has not been asked (e.g. because of skipping logic).

[1] JScript is Microsoft's adaptation of ECMAScript, a computer language standard based on several originating implementations, most notably Netscape Communication's Javascript and JScript itself.

6.2.1 f("qID")

Its basic syntax is `f("qID")`, where qID is the question ID specified in Properties (see 5.2.1.4 Properties). The values returned are different for different question types, but they are always strings or sets of strings:

Question type	**Values returned**
Single	The precode of the answer.
Open	A string with the text answered.
Multi	A set with the precodes of the answers selected.
Multi with Ordered or OpenText checked (see 5.2.1.4 Properties)	A set with the precodes of the elements in the answer list that are answered.
Grid	A set with the precodes of the elements in the answer list that are answered.
"Other" from answer lists	A string with the text answered.

Table 20: Values returned from f("qID") for different question types

6.2.1.1 Single

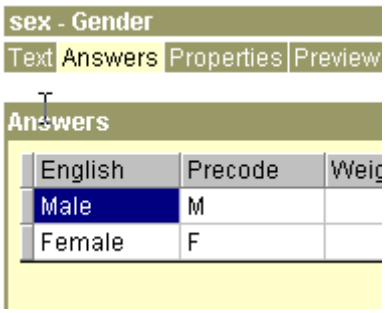


Figure 19: Answer list of single question "sex"

If the respondent answers Male, `f("sex")` will return the value *M*.

6.2.1.2 Open Text

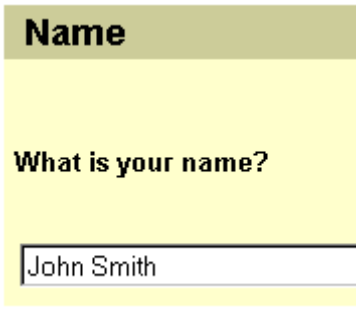


Figure 20: Open Text question "name"

`f("name")` will return *John Smith*

6.2.1.3 Multi

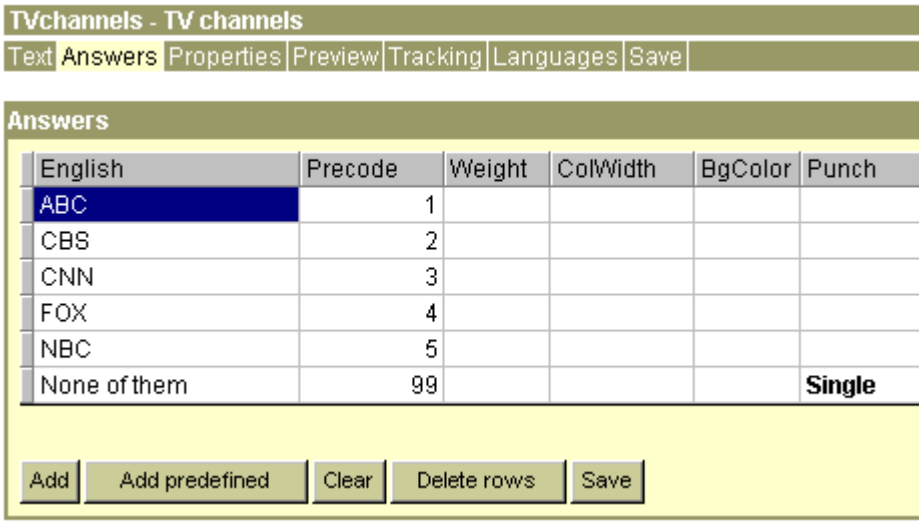


Figure 21: Answer list of multi question "TVchannels"

If the respondent answers ABC, CNN and NBC, `f("TVchannels")` will return a set containing *1,3,5*.

This answer list (and its precodes) will be used in the next examples as well.

6.2.1.4 Multi with OpenText or Ordered Property Checked

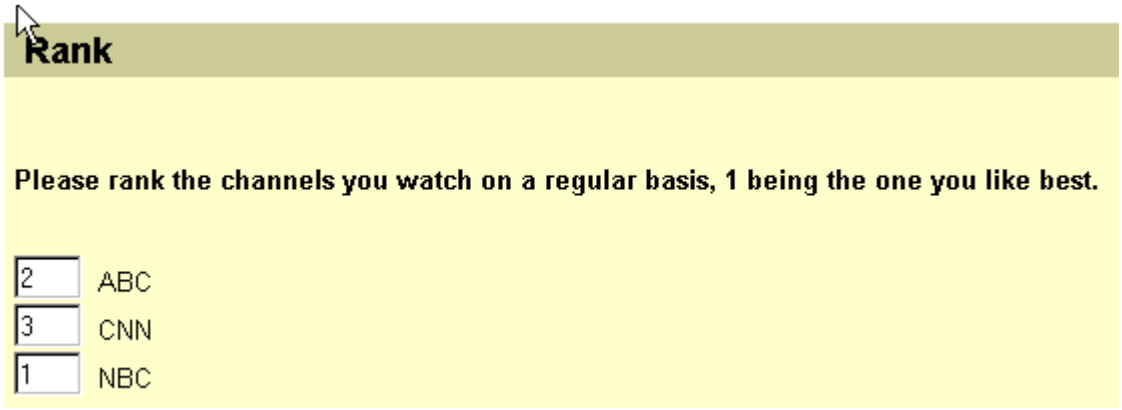


Figure 22: Multi question "rank"

For an Ordered multi question the respondent has to provide consecutive answers in the range from 1 to the number of items in the list (3). `f("rank")` will return the precodes of the items answered in the list, *1,3,5*. When used in a script node you have to use f("rank").get() (see 6.8.5.1.5 for more details).

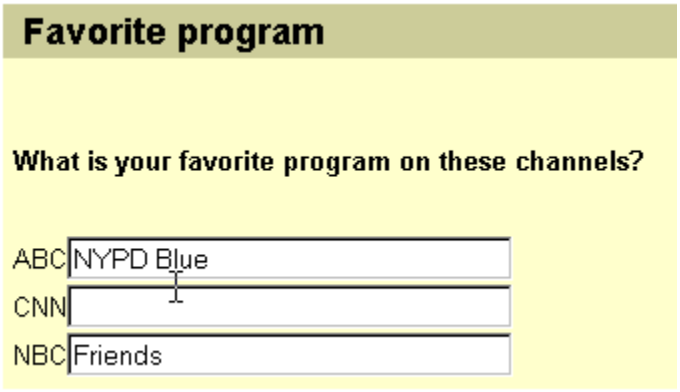


Figure 23: Multi question "program"

A multi question with the OpenText property checked will present the answer list If "Not required" is checked as well, the respondent will not have to answer the question. If the question is answered like this, `f("program")` will return a set containing *1,5*.

(Notice that the answer lists of these two questions as well as the grid in next question are filtered based on the answers given on the multi question *TVchannels*. More on filtering with "precode masks" in 6.4)

6.2.1.5 Grid

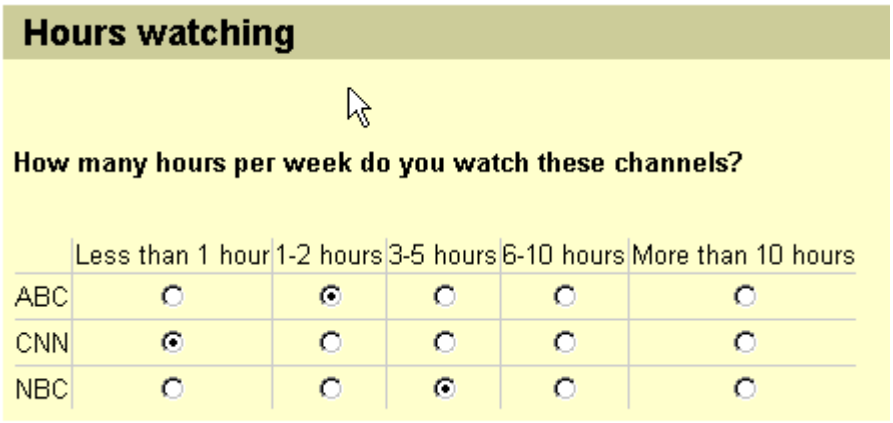


Figure 24: Grid "hours"

`f("hours")` will return a set of precodes from the answer list that has an answer, here it would be 1,3,5.

6.2.1.6 Other, Specify

favorite - Favorite TVchannel

Text | Answers | Properties | Preview | Tracking | Languages | Save

Answers

English	Precode	Weight	ColWidth	BgColor	Punch	KeepPos	Other
ABC	1						
CBS	2						
CNN	3						
FOX	4						
NBC	5						
Other, specify	6						**Yes**
Don't know	99						

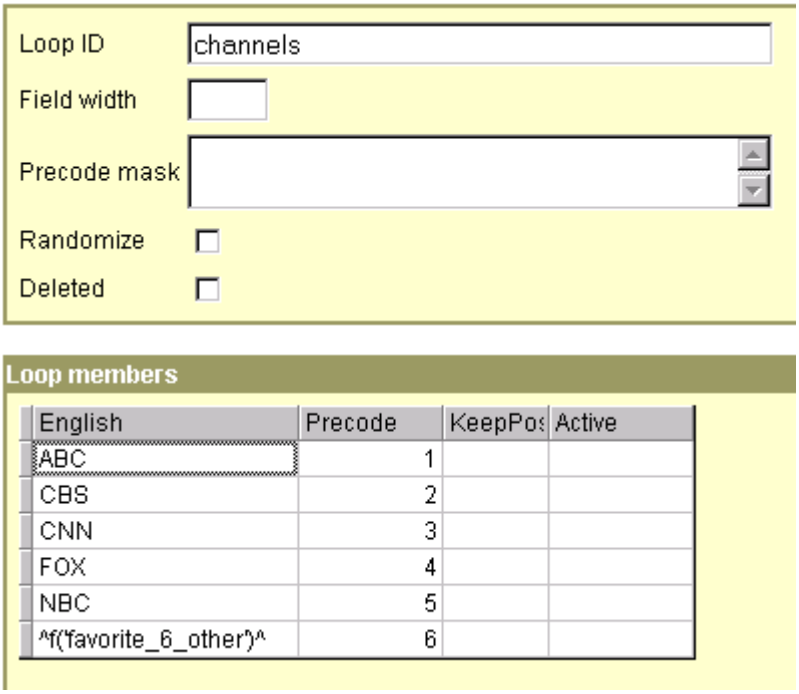
Add | Add predefined | Clear | Delete rows | Save

Figure 25: Answer list of single question "favorite"

Figure 26: Single question "favorite"

An Other text box will have "question id"+_+"precode"+_other as variable name, here it would be *favorite_6_other*. `f("favorite_6_other")` will return the string *MTV*.

This syntax is useful when you want to include the value entered in the Other text box in answer alternatives in later questions or in loops.



In this case *MTV* will appear in the loop, not *Other, specify*.

6.2.2 f("qID")["precode"]

Multi and grids group one or more variables in the same form. The answers are stored in consecutive columns in the database (see 5.2.1.4 Properties for an explanation of how they are stored). To get the answer of an item of a Multi or a Grid question, use the precode of the item you want as identifier: `f("qID")["precode"]`

Question type	**Values returned**
Multi	1 if the item with this *precode* is checked off, 0 if it is not.
Multi with Ordered checked	A string with the value (rank) given to the item with this *precode*.
Multi with OpenText checked	A string with the text answered for the item with this *precode*
Grid	The precode (from scale) of the answer to the item with this *precode.*

Table 21: Values returned from f("qID")["precode"] for different question types

6.2.2.1 Multi

If we use the same example as in 6.2.1.3, the multi question *TVchannels* where ABC, CNN and NBC are selected, `f("TVchannels")["1"]` will return 1 (since ABC is checked), `f("TVchannels")["2"]` will return 0 (since CBS is not selected).

6.2.2.2 Multi with Ordered Property Checked

Using the same example as in 6.2.1.4, the multi question *rank* where ABC was ranked as number 2, CNN as 3 and NBC as number 1, `f("rank")["1"]` will return 2 (since ABC was ranked as number 2), `f("rank")["3"]` will return 3 (CNN) and `f("rank")["5"]` will return 1 (NBC).

6.2.2.3 Multi with OpenText Property Checked

Using the multi question *program* from 6.2.1.4, `f("program")["1"]` will return *NYPD Blue*, `f("program")["3"]` will return an empty string (since the respondent did not answer anything for CNN) and `f("program")["5"]` will return *Friends*.

6.2.2.4 Grids

See the example in 6.2.1.5, the grid *hours*. Using the answers from that example, `f("hours")["1"]` will return 2, `f("hours")["3"]` will return 1 and `f("hours")["5"]` will return 3 provided that the scale of the grid is defined as in Figure 27.

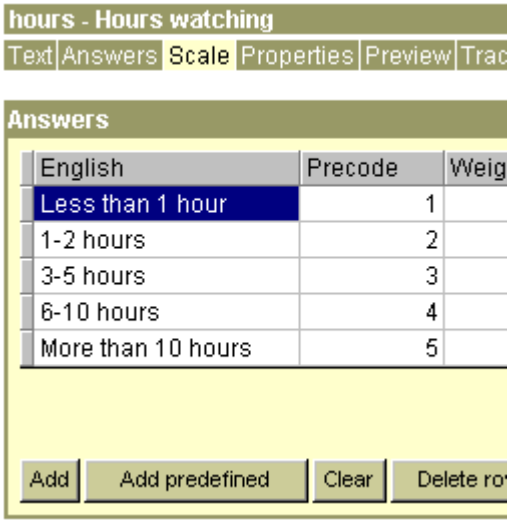


Figure 27: Scale of grid "hours"

6.2.3 Loops - f("qID","iteration_precode","iteration_precode",…..)

You refer to questions inside loops as described above, but you may want to be able to refer to the value in a special iteration within a loop.

Say we have a loop that iterates through the list of channels in the question *TVchannels* from 6.2.1.3. For each of them, a multi question *types* is asked and the respondent checks off different types of programs he or she likes to watch on the channel. Then there is a loop inside the other loop iterating through the types of programs the respondent has checked off, and inside that loop the respondent is asked to rate the channel's programs of that type, and if he/she gives it a score 4 or 5 he/she is asked to give the reasons for giving that score.

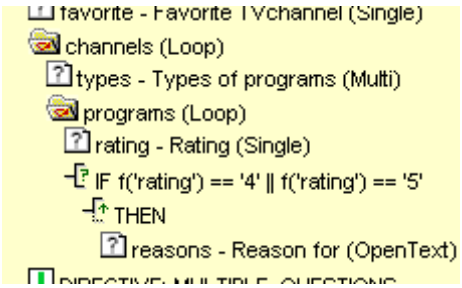


Figure 28: Nested loops

As you see from the IF-condition, within the innermost loop, *rating* may be referenced as a normal single question, using `f('rating').` This expression will give the value of *rating* within the current iteration. But if you wanted the value of *rating* in another iteration, the "precode" of that iteration would have to be specified. `f("rating","1")` would be the value of *rating* of "Sit-coms" (see Figure 29).

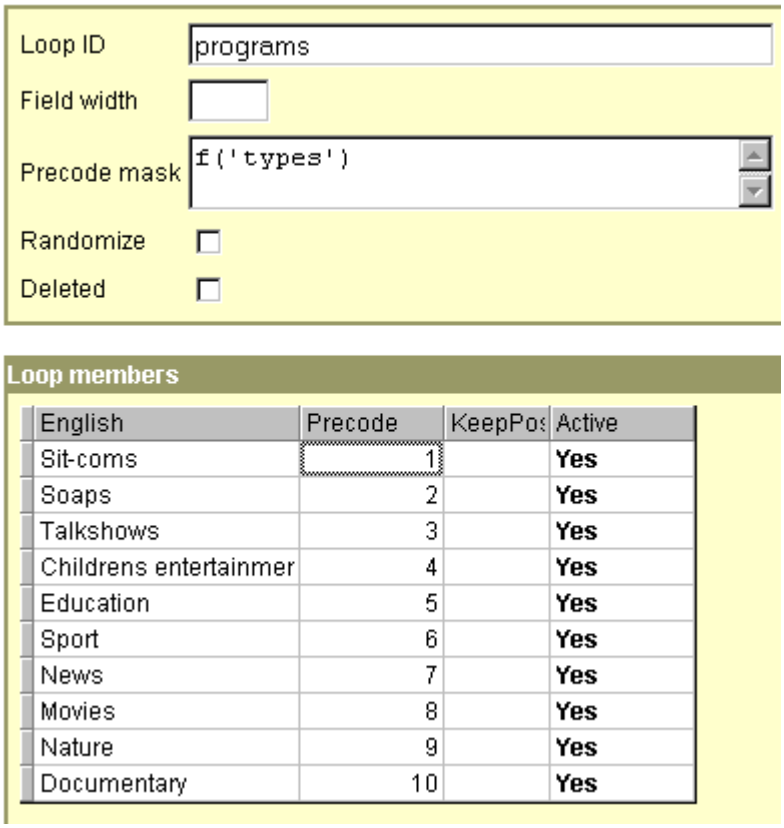


Figure 29: The inner loop, "programs"

This expression may be used within both loops. For the outer loop the current value for "channels" will be used. But you may specify the iteration of the outer loop as well (Outside the loops you would have too!). `f("rating","1","5")` would give the rating of NBC's "Sit-coms" (see Figure 30). Notice the order: The innermost loop first. Outside the loops an expression like f("rating") would not make much sense.

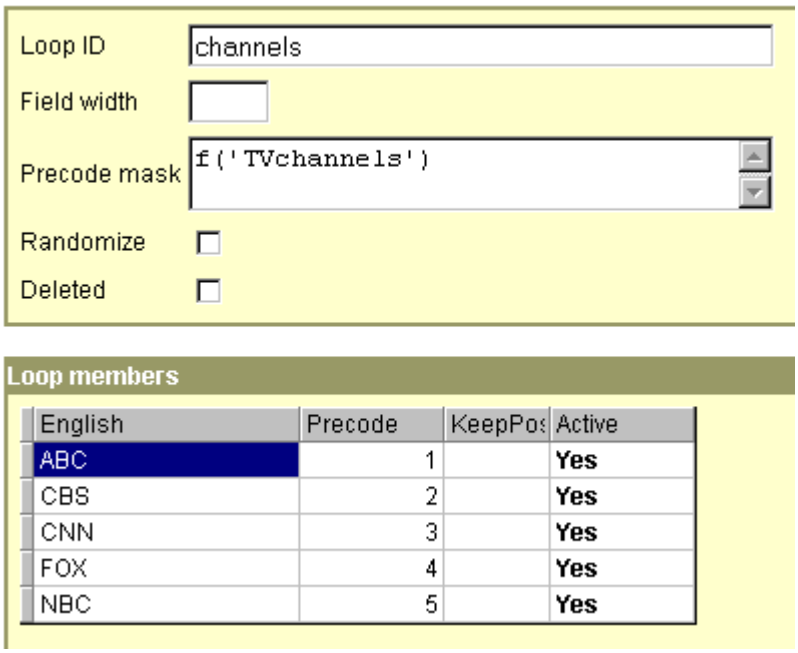


Figure 30: The outer loop, "channels"

Inside the loops you may refer to the current iteration with the syntax `f("loopID")`, e.g. `f("channels")` or `f("programs")`. The value returned will be the precode of the current iteration.

6.3 Conditions

When making conditions, the following standard Jscript logical operators are useful:

Description	**Symbol**
Logical NOT	!
Less than	<
Greater than	>
Less than or equal to	<=
Greater than or equal to	>=
Equality	==
Inequality	!=
Logical AND	&&
Logical OR	||

Table 22: Jscript logical operators

6.3.1 .inc()

As described in 6.2.1, f("qID") of multi question returns a set of the precodes answered to question qID. If you want to test for a particular value, use .inc (includes). l.inc(r) returns TRUE if r is a member of the set l. E.g. `f("qID").inc("precode")` will return TRUE if "precode" is one of the answers to qID.

6.3.2 .size()

If you need a test on how many answers were selected on a multi question, you may use .size(). `l.size()` will return the number of items in the set l. `f("qID").size() > 1` will return TRUE if qID has more than one answer.

6.3.3 .toNumber()

The .toNumber() method converts the variable to a number.

Example: The f function returns strings or sets of strings. If you e.g. had a single question with an answer consisting of 11 elements with the precodes “1”, “2”, …, “11” the expression

f("qID") <= "4" would evaluate to TRUE for the following answers: “1”, “10”, “11”, “2”, “3” and “4”.

.toNumber() converts the variable to a number. So f("qID").toNumber() <= “4” would evaluate to TRUE for the following answers: “1”, “2”, “3” and “4”.

You may also use f("qID").toNumber() <= 4

6.4 Precode Masks

Precode masks are used to filter answer lists of grids, single and multi questions and to select a set of iterations for a loop. In the precode mask field in Properties of a question (see 5.2.1.4) or in the loop construct (see 5.2.7) is a Jscript expression that evaluates to a set of precodes. The answer list (or loop member list) will be filtered based on the set of precodes in the Precode mask field.

We have already seen how the f function evaluates to a set of precodes when used on multi questions. These are other functions that are useful when constructing sets of precodes:

6.4.1 a("qID")

`a("qID")` returns the complete set of precodes defined in the answer list belonging to qID, no matter if they are answered or not.

6.4.2 set(), nset(), nnset()

Sometimes you need to create sets yourself. These three functions simplify this.

`set("precode","precode",…)` returns a set consisting of the precodes you specify.

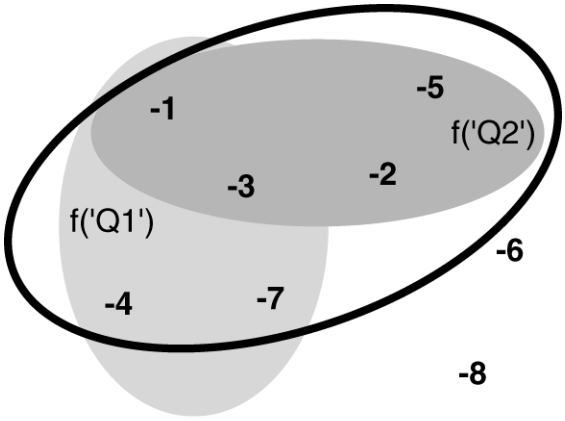
`nset(n)` returns a set populated with the members 1, 2, 3,…,n.

`nnset(m,n)` returns a set with values from m to n: m, m+1,…, n-1, n.

Example: You use a general predefined list as your answer alternatives in several questions, but in a particular question you only want to return answer alternatives 8,9,10,11 and 12. Then all you have to do is to put nnset(8,12) in the Precode mask property of that question.

6.4.3 .union()

`l.union(r)` returns a new set that is the union of l and r, that is a set consisting of all precodes in either l or r. If you want to add precodes to a set in a precode mask, then use .union()

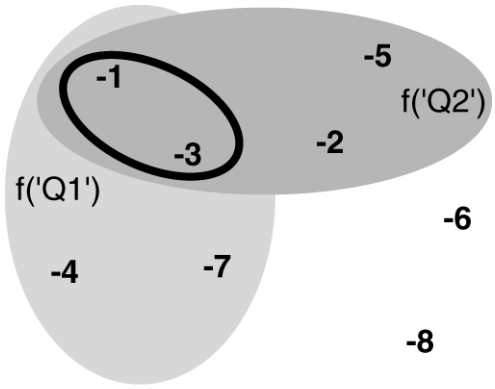


f('Q1').union(f('Q2'))

Example: f(‘qID’).union(set(‘99’))Here you will always include the answer alternative with precode ‘99’

6.4.4 .isect()

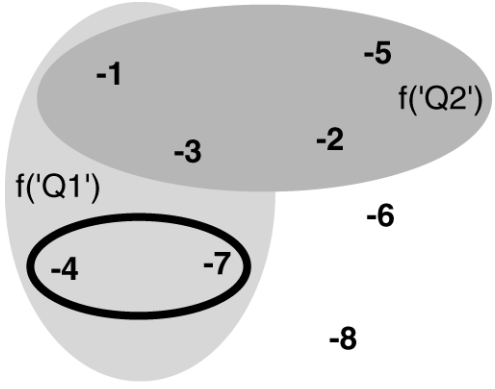
`l.isect(r)` returns a new set that is the intersection of l and r, that is a set consisting of all precodes both in l and r.



f('Q1').isect(f('Q2'))

6.4.5 .diff()

`l.diff(r)` returns a new set that is the difference between l and r, that is a set consisting of all precodes in l and not in r. If you want to remove precodes from a set in a precode mask, then use .diff()



f('Q1').diff(f('Q2'))

Example: f(‘qID’).diff(set(‘1’)) Here ‘1’ is never included.

6.4.6 Precodes

Remember that the precode mask gives a set of precodes that are used to filter the answer list/iterations. Make sure that you use exactly the same precodes on the items of the answer lists in the different answer lists, or else the masks will be wrong. Tip: Use Predefined scales/lists (see 5.2) as much as possible when using precode masks, because then you are sure that the precodes are exactly the same.

6.5 Text Substitution/Response Piping

In order to retrieve text or values from a question and insert it in the question wording of another question, you may use caret (^) in front of and after a Jscript expression.

Example:

In the *TVchannels* question from 6.2.1.3 `^f("name")^` is inserted in the text field. This is referring to the *name* question of 6.2.1.2.

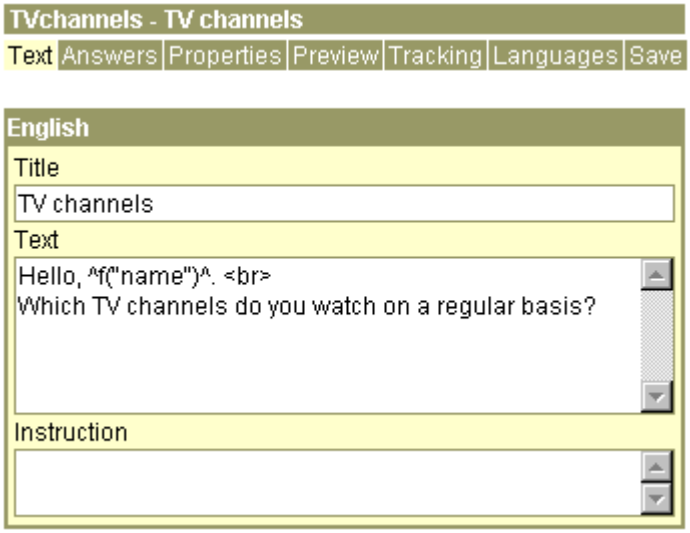


Figure 31: TVchannels

When displayed to the respondent, the question will look like this (of course provided that the respondent answered "John Smith" to the *name* question):

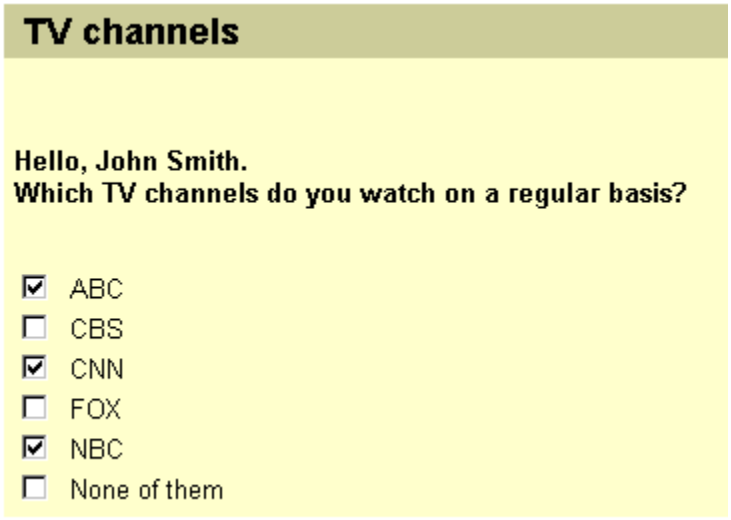


Figure 32: "TVchannels" question with text substitution.

Question type	**Returned from ^f("qID")^**
Open text	The text answered in the text box.
Single	The text from the answer list corresponding to the precode returned from f("qID") in the current language.
Multi	The text(s) from the answer list corresponding to the precode(s) returned from f("qID") in the current language. If more than one, the answer will be separated by comma and with "and" between the last two items.
Other property of answer lists	The text answered in the text box. (The question ID is qID_precode_other (see 5.2.1.4))
Loop	The text of the loop member corresponding to the iteration with precode qID in the current language.

Table 23: Texts returned from ^f("qID")^ for different question types

Question type	**Returned from ^f("qID")["precode"]^**
Multi with OpenText or Ordered property	The text answered in the text box.
Grid	The answer from the scale corresponding to the precode returned from f("qID")["precode"] in the current language.

Table 24: Texts returned from ^f("qID")["precode"]^ for different question types.

The same rules will apply to loop questions, but there you may need to specify iteration(s) as well (see 6.2.3).

6.5.1 Other methods

If you want other than the standard results described in Table 23 and Table 24 to be piped into the question text, use one of the following methods:

6.5.1.1 Coded Variables (Single Questions and the Elements of Grids)

.value() – Returns the value (precode)

.valueLabel() – Returns the label (text from answer list) in the current language

.domainValues() – Returns an array of all valid values (precodes)

.domainLabels() – Returns an array of all labels (texts from answer list) in the current language

6.5.1.2 Compounds (Multi Questions and Grids)

.values() – Returns an array of the values of each item in the answer list.

.categories() – Returns an array of all precodes (from the answer list) with an answer (checked in multi/answered in grid)

.categoryLabels() - Returns an array of labels (text from Answer list) associated with each category in the current language

.domainValues() – Returns an array of all categories (the precodes from the answer list)

.domainLabels() – Returns an array of all labels (texts from answer list) in the current language.

6.5.1.3 Some practical use

.value(), .values() and .categories() are used to ensure that you get the values stored in the database returned. In most contexts you wouldn't need to use it, because f("qID") will give you exactly what you want, but you may e.g. sometimes want the precodes or the values answered to be piped in instead of the labels (in ^ mode in the question text).

When you use ^f("qID")^, by default you will get the labels piped in for all question types other than open text questions, where you get the value answered. But perhaps (for some reason) you for example want the precode (or precodes) to be piped in instead. Then you may use these functions.

.value() gives the value stored for that field. For f("qID").value() on a single question it will be the precode of the answer, for an open text question it will be the text answered, for f("qID")["precodeID"].value() on a multi question it will be 1 or 0 (or nothing) depending if the answer with precode "precodeID" has been selected or not(or the question has been asked), and on a grid it will be the precode of the answer to the item from the answer list with precode "precodeID".

f("qID").values() gives an array of the values of question qID. If it is a multi question it will be an array with 1's and 0's depending on which items is selected, if it is a grid it will be the precodes of the answers.

f("qID").categories() gives an array of precodes of items that are answered on question qID (where .toBoolean() returns TRUE). For a multi this will be all items that are selected. For a grid it will be all items that are answered (normally in a grid all items are required, so this will return an array with all precodes from the answer list).

f("qID").domainValues() gives an array of all the precodes in the answer list (all possible answers/precodes). This may be very useful in scripting when you want to do some checking on all elements in a multi or a grid (e.g. to set a precode mask based on the answers of a grid or to copy a multi question into a hidden multi variable - or something like that.)

Example: Use this function to copy a multi into another:

```
function copymulti(from,to)
 {
  var precodes=f(from).domainValues()
  var i;
  for(i in precodes)
  {
   f(to)[precodes[i]].set(f(from)[precodes[i]])
  }
 }
```

.valueLabel(), .domainLabels(), categoryLabels(), .domainLabels() are all functions to return the variable label(s) on functions, either the label of a specific answer (.valueLabel()) or the labels of all possible answers (.domainLabels() on a single question, .categoryLabels() on a multi or grid for the answer list and .domainLabels() on a grid for the labels of the scale). They are used if you need the labels outside the ^ mode, in script nodes and validation code.

6.6 Validation Code

When you add a form to a questionnaire you actually add four stages of operations to the interview program:

1. **User interaction:** Display one or more forms to the person entering responses and wait for input.
2. **Check size:** Verify that open-ended answers comply with the field width. If not, truncate answers and raise error flag. Note that for performance reasons, no such checks are added for coded variables. The system assumes that the survey designer makes sure that the code lengths do not exceed the field width.
3. **Store:** Store the value of entered responses.
4. **Validate:** Check that answers stored comply with the specified data validation rules. If an error has been raised then repeat from 1.

This section describes the default validation rules and instructs you on how you can add your own code to a form in order to perform custom validation operations.

6.6.1 Default Validation Rules

The system provides these types of built-in error checking:

1. Required answer testing.
2. Exclusivity checks.
3. Other-Specify verification.
4. Rank order testing.
5. Answer size tests for fixed-width fields.

If you do not like the default handling of 1–4, or prefer to use a less stringent quality scheme for the data you collect, each type of error handling can be turned off individually when generating WI (see 7.2.1). The system always performs size checks, because failing to do so will cause database errors. A description of each test is provided below:

Required answer testing

Required answer testing exists in order to ensure that you get complete responses to your surveys.

- At least one alternative must be checked/selected for a coded variable defined in a **single** or **grid** form.
- Textboxes used for input of open responses cannot be left blank.

Note that for plain **multi** forms, an answer where no alternatives are checked are considered a valid answer, unless it contains “Single punch” alternatives which are subject to the exclusivity testing described next. Required answer testing may be turned off for separate questions by selecting "Not Required" in the question's properties (see 5.2.1.4).

Exclusivity testing

Exclusivity testing is used for **multi** questions that have one or more “Single punch” alternatives. A “single punch” element in a **multi** form declares a variable that is exclusive within the group of variables declared in the form. A positive response to an exclusive value eliminates the possibility that any other variables in its group can have a positive value, e.g. a “None of the above” alternative.

The exclusivity test assumes that the alternatives given exhaust all possible answer combinations and at least one positive answer is required.

In consequence, the response will only be considered valid if the respondent checks at least one of the none-exclusive or exactly one of the exclusive options.

Other-Specify validation

Other-Specify validation verifies that:

- The respondent provides a specification to an alternative that has a specify-field if that alternative is checked.
- That the alternative is checked/answered if a specification is provided.

Rank order testing

The system applies rank order testing when the “Ordered” option is selected for a form. **Multi** and **grid** answers must then constitute a set of consecutive integers starting at 1.

6.6.2 Adding Your Own Validation Code

One of the form properties is a text box where you may enter arbitrary script code to validate answers. Depending on the complexity of the validation problem, crafting this code may require a more intimate knowledge of JScript than can be presented in this document. This presentation of validation code is limited to a description of the interface provided by the runtime for signaling errors and setting error messages. It also provides a few examples.

6.6.2.1 A typical Validation Code

A typical validation code would consist of a condition with an expression that evaluates to TRUE for some type of error situation where you want the respondent to be unable to move to next page and you want an error message to be displayed:

```
if(expression)
 {
   RaiseError();
   <some function(s)setting the text of the error message(s)>
 }
```

6.6.2.2 RaiseError()

The RaiseError function signals to the interview runtime that the input entered in a form is invalid and that the form should be redisplayed with an error message.

6.6.2.3 Error Messages

When one or more questions on a page have an error, an error message will be displayed on top of the screen informing the respondent that one or more of the questions on the page has errors (Default: "Please answer all questions properly. Please review all questions on this page."), and more detailed error messages will be displayed above the question(s) where the error(s) occurred (see Figure 33).

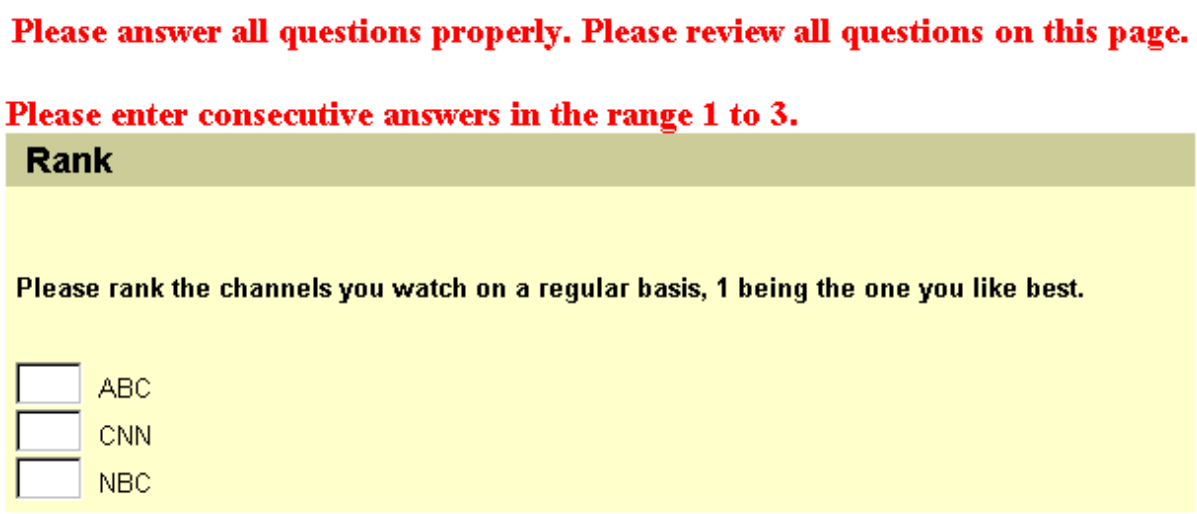


Figure 33: Error messages

A number of functions are provided to manipulate the error messages:

ClearErrorMessage() – clear the error message at the top of the page (will remove the default error message).

SetErrorMessage(langID, message) – set the error message at the top of the page (will replace the default error message).

ClearQuestionErrorMessage() – clear the error message for the current question (will remove the default question error message).

AppendErrorMessage(langID, message) – appends *message* to the current page's error message.

SetQuestionErrorMessage(langID, message) – sets the error message for the current question.

AppendQuestionErrorMessage(langId, message) – appends *message* to the current question's error message.

langID is a language identification. When conducting multilingual surveys you should provide one message for each of the languages used for interviewing. The runtime provides an object `LangIDs` which maps language codes to the internal values of **confirmit**. The codes used are specified in the international standard ISO 639, and are listed in Appendix B. E.g. LangIDs.en is English, LangIDs.fr is French and so on.

message is a string value with the error message you want to display. This could be a text inside quotes, a Jscript string variable or you may use the ErrorTemplate function (below).

6.6.2.4 Template Based Error Messages

The system allows you to use templates for your error messages in custom validation. For each error type, various elements are filled in that can improve the information content of the messages you report

to the end users. Instead of simply using fixed strings for error messages, the `ErrorTemplate` function can be used to “plug in” information about the current question.

The syntax of ErrorTemplate is `ErrorTemplate(spec)` where spec is a string with the template specification. Inside the spec you may use different elements that are singled out with caret (^) as pre- and suffix, e.g. ^MISSING^.

Example:

```
AppendQuestionErrorMessage(LangIDs.en, ErrorTemplate("^MISSING^ has
not been answered."))
```

will append the text "*label(s)* has not been answered" to the error message.

Templates are available for all the standard validation. See Appendix A for a description of all the templates.

6.6.2.5 Some default functions for Conditions in Validation Code

The **confirmit** runtime provides a number of functions that may be used in the conditions of your validation code:

QuestionErrors() - returns TRUE if an error has been raised during validation, FALSE otherwise

MissingRequiredError() – returns TRUE if one or more required answers to a question are missing, FALSE otherwise

ExclusivityError() – returns TRUE if an exclusive answer ("single punch") in a multi has been selected together with any other alternative, FALSE otherwise

SizeError() – returns TRUE if one or more open-ended answers exceeded the maximum length specified in field width, FALSE otherwise

NotSpecifiedError() – returns TRUE if an alternative in an Other-Specify construct has been selected/answered without filling in the associated "Specify" text box, FALSE otherwise.

NotSelectedError() – returns TRUE if a "Specify" value has been entered in the text box of an Other-Specify construct without selecting/answering the associated alternative, FALSE otherwise.

RankError() – returns TRUE if the "Ordered" property has been selected for the form and the answers to the questions are non-numeric, do not start at 1 or are non-consecutive, FALSE otherwise.

NumericError() – returns TRUE if the “Numeric” property has been selected and the answer is not numeric, FALSE otherwise.

PrecisionError() – returns TRUE if the “Numeric” property has been selected and the answer can not be stored within the defined precision, FALSE otherwise.

ScaleError() – returns TRUE if the “Numeric” property has been selected and the answer contains more decimals than in the defined scale, FALSE otherwise.

RangeError() – returns TRUE if the “Numeric” property has been selected and the answer is outside the defined range, FALSE otherwise.

Example:

```
if (ExclusivityError())
 {
      AppendQuestionErrorMessage(LangIDs.en,
ErrorTemplate("^DEF_EXCL^ cannot be selected with any other
alternative."))
}
```

(Notice that you do not need to use RaiseError(). When ExclusivityError() is TRUE this situation is already marked as a error situation by standard validation).

6.7 Script Nodes

Script nodes typically contain code for

- internal programming purposes
  Example: Say for instance, you have the multi question *TVchannels* from 6.2.1.3. You want a single question *most* after that one where you ask something like "And which of these channels do you watch most?" If the respondent answers just one in *TVchannels*, you would not want them to get this question. That can be done by adding a condition (see 5.2.2) with the code f('TVchannels').size() > 1, and move the single question *most* to the then-branch of the condition. But you would want to have the one channel they answered in *TVchannels* in the data of the *most*. This can be accomplished by making an ELSE-branch to the condition, and add a script node in it.

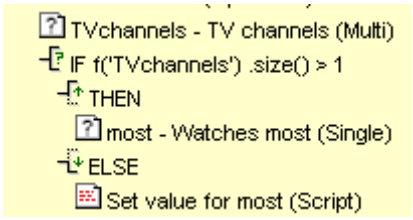


Figure 34: If-then-else construction

  The script node should contain this simple code:

```
f("most").set(f("TVchannels"))
```

- defining functions used in precode masks, conditions, other script nodes or validation code
  Example: You have a grid in one question and want the next question only to include the answer alternatives which have been given the highest score, i.e “6” on a scale. You then write a function in a script node:

```
function onlyinclude6(){
    var s = new Set()
    var precodes=f("qID").domainValues();
    for(var i=0; i<precodes.length; ++i)
     {
      var precode=precodes[i];
      if(f("qID")[precode]=="6")
          s.add(precode)
     }
    return s}
```

  In the precode mask property of the next question you call this function by using it’s name. In this case onlyinclude6*()*

- setting the values to hidden variables (see 5.2.1.4)
  Example: Instead af asking a respondent which browser they use, you may include a hidden question Browser and set it’s value by using the BrowserType() and BrowserVersion() functions in a script node.

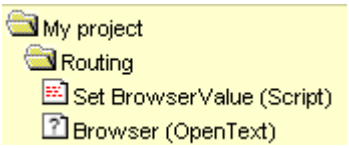


Figure 35: Setting a value for a hidden question

  The only code in the BrowserValue script should be:

```
f('Browser').set(BrowserType()+BrowserValue())
```

6.8 Useful functions and methods

This section describes some functions and methods that are useful when you are building questionnaires. The functions and methods can be used in validation codes, in precode masks, in conditions or in script elements. Remember that the examples only show you one possible use of the function.

6.8.1.1 Arithmetic functions

6.8.1.1.1 Sum()

Returns the sum of its arguments. Valid argument types are numbers, arrays or any kind of variable object.

Example: You have a multi question with OpenText selected and want the respondents to enter a percentage that sums up to 100%. Include this script in the Validation code property:

```
if (Sum(f('qID'))!=100){
  RaiseError();
  SetErrorMessage(LangIDs.en, "Your responses added up to " +
Sum(f('qID')) + ", they must add up to 100");
}
```

6.8.1.1.2 Count()

Returns the number of arguments.

6.8.1.1.3 Average()

Returns the average of its arguments.

6.8.1.1.4 Max()

Returns the maximum of its arguments.

Example: You ask the respondents how much time in hours they spend watching some TV channels a week in a multi question with OpenText selected. You want to ask a follow-up question if some TV channel(s) is(are) watched for more than 15 hours a week.

Include this condition:

```
IF Max(f('qID')) > 15 THEN….
```

6.8.1.1.5 Min()

Returns the minimum of its arguments.

6.8.2 Conversion

6.8.2.1.1 .toString()

The .toString() method converts the variable to a string.

6.8.2.1.2 .toNumber()

The .toNumber() method converts the variable to a number (see 6.3.3 for details).

6.8.2.1.3 .toBoolean()

The .toBoolean() method converts the variable to a Boolean (TRUE/FALSE). If you use an expression like this in a validation code:

```
if(f("qID").toBoolean())
```

-and qID is a single question, it would be TRUE if it is answered, FALSE otherwise
-and qID is a open text question, it would be TRUE if it is answered, FALSE otherwise
-and qID is a multi question, it would be TRUE if one or more of the checkboxes is selected, FALSE otherwise
-and qID is a grid, it would be TRUE if one or more of the elements in the answer list has an answer, FALSE otherwise (so there might be some answers missing in some lines)

If you use
```
if(f("qID ")["1"].toBoolean())
```

-and qID is a multi question, it would be TRUE if the element with precode "1" is selected, FALSE otherwise
-and qID is a grid question, it would be TRUE if the element from the answer list with precode "1" is answered, FALSE otherwise

Example: You have a multi question (q1) and want the respondents to select at least one answer. By default you don't have to select any answers in a multi question, even with the Answer Required test selected (see 7.2.3.3). You have to include a script like this in q1's validation property:

```
if (!f('q1').toBoolean()){
  RaiseError();
  SetErrorMessage(LangIDs.en, "You must provide at least one
answer!");
}
```

6.8.2.2 Classification functions

6.8.2.2.1 .size()

The size method returns the number of selected precodes in a multi question.

As an alternative to the size method you can use the Sum() function (see earlier in this chapter for a description).

Example: You have a multi question and want to force the respondent to select exactly 3 answers. Then select the not required property for that multi question and add this script in the validation code property:

```
if(f("qID").size()!="3")
 {
  RaiseError();
  SetQuestionErrorMessage(LangIDs.en,"You must select exactly 3
answers!");
 }
```

6.8.2.2.2 .inc()

The include method is used to check if a set includes some specified precodes.

Example: The example shows a condition node where we check if a question does *not* include the precode “99”. (could be an answer like “None of the above”).

```
IF !f('qID').inc(['99']) THEN…
```

This is a work around for the lack of a GOTO functionality in confirmit. The example is an alternative to `IF f('qID').inc(['99']) THEN GOTO…` You have to refrase your GOTO logic.

6.8.2.2.3 IsNumeric(arg)

Returns true if the argument is numeric.

Example: You have a multi question with the OpenText property selected and want the first answer input to be numeric.

```
if((!IsNumeric(f("qID")["1"])) && (f("qID")["1"] !="")){
  RaiseError();
  SetQuestionErrorMessage(LangIDs.en,"Not a valid number! Please
correct!");
 }
```

In this example you don’t want the user to get an error message when the first answer is left empty. That’s why we have included the second comparison.

Note that you could use the property Numeric as an alternative (see 5.2.1.4 for details)

6.8.2.2.4 IsInteger(arg)

Returns true if the argument is an integer.

6.8.2.2.5 IsDateFmt(arg, format)

Determine whether the provided argument is a valid date according to format.

Within a date format, the following character sequences have special significance:

- Y: Year, four digits.
- YY: Year, two digits.
- YYYY: Year, four digits.
- M: Month number, one or two digits.
- MM: Month number, exactly two digits.
- D: Day number, one or two digits.
- DD: Day number, exactly two digits.

All other characters are treated as separators.

All parts are optional. If omitted, then the system date is used to determine month and year and the number 1 is used for the day.

Century interpolation for 2-digit years is handled with a break-point value of 10: values between 0-10 are interpreted as the years 2000-2010, while 11-99 are treated as 1911-1999.

IsDateFmt returns an object with the properties day, month, year if the date is valid, null otherwise.

6.8.2.2.6 IsDate(day, month, year)

Determines whether the provided year, month, day combination constitutes a valid Gregorian date.

The month and year arguments are optional. If they are not provided then the current month and/or year is used.

Century interpolation for 2-digit years is handled with a break-point value of 10: values between 0-10 are interpreted as the years 2000-2010, while 11-99 are treated as 1911-1999.

IsDate returns an object with the properties day, month, year if the date is valid, null otherwise.

6.8.2.2.7 IsEmail(arg)

Checks whether arg has the format of a valid e-mail address.

Note: the function does not attempt any name lookups or address verification.

6.8.2.2.8 IsNet(quadIP, quadNet, quadMask)

Determines whether an IP address belongs to an Internet network.

Arguments:

quadIP:	The IP address, on quad form (X.X.X.X)
quadNet:	The network number, on quad form.
quadMask:	An optional mask, on quad form, if subnetting is used.

If no mask is provided then the number of bits that constitute the network part of the address is determined from the address class type. In this case the function returns true if the IP address' network number is identical to the supplied network number and it has a non-zero local address. Otherwise it returns false

If a mask is provided then the function returns true if the IP address' network and subnet parts are identical to the one supplied and the host number part is non-zero. Otherwise it returns false.

6.8.3 Context information

6.8.3.1.1 CurrentID()

Returns the current respondent ID, may be used to arbitrate between blocks of questions, etc.

6.8.3.1.2 CurrentLang()

Returns current language used.

6.8.3.1.3 InterviewStart()

Returns the respondent’s interview start time.

6.8.3.1.4 InterviewEnd()

Returns the respondent’s interview end time.

6.8.3.1.5 GetStatus()

Returns the current interview status.

6.8.3.1.6 SetStatus()

Sets the current interview status.

6.8.3.1.7 Forward()

Returns true if moving forward through the questionnaire.

6.8.4 Browser information

6.8.4.1.1 BrowserType()

Returns the browser type, as detected by the MSWC.BrowserType component.

Example: Together with the function BrowserVersion()(see under), this information can be included as hidden variables instead of having the respondents to enter this information manually in a question themselves (see 6.7 for an example).

6.8.4.1.2 BrowserVersion()

Returns the browser version, as detected by the MSWC.BrowserType component (see 6.7 for an example).

6.8.4.1.3 RequestIP()

Returns the IP address which the request originated from as a string on quad IP form.

6.8.5 Handling sets

6.8.5.1.1 a('qID')

`a("qID")` returns the complete set of precodes defined in the answer list belonging to qID, no matter if they are answered or not.

6.8.5.1.2 set()

`set("precode","precode",…)` returns a set consisting of the precodes you specify (see 6.7 for an example). Notice that the set() function is *not* the same as the .set() method (see 6.8.5.1.7 for a description of the .set() method).

6.8.5.1.3 nset()

`nset(n)` returns a set populated with the members 1, 2, 3,…,n.

6.8.5.1.4 nnset()

`nnset(m,n)` returns a set with values from m to n: m, m+1,…, n-1, n (see 6.4.2 for an example).

6.8.5.1.5 .get()

Returns the associated value (precode). Most often the f ('qID') function will return exactly the same as f('qID').get(). However, for ordered multi questions it is necessary to use the .get() method.

Example: You want to ask your respondents a question about the alternative they have ranked as number one in an earlier ordered multi question. Then you may add this generic function in a script node:

```
function rankedfirst(qid){
     var strfirst;
     strfirst = "";

     var precodes=f(qid).domainValues();
     for(var i=0; i<precodes.length; ++i) {
     var precode=precodes[i];
     if (f(qid)[precode].get() == "1"){
       strfirst = f(qid)[precode];
       }
 }
 return strfirst;
}
```

In the precode mask property you call this function.

6.8.5.1.6 .label()

Returns the label in the current language.

6.8.5.1.7 .set(v)

The .set() method sets one object equal to another object. Notice that the .set() method is *not* the same as the set() function, described in 6.8.5.1.2.

Example:

```
f("qID1").set(f("qID2"))
```

6.8.5.1.8 .union()

`l.union(r)` returns a new set that is the union of l and r (see 6.4.3 for details).

6.8.5.1.9 .isect()

`l.isect(r)` returns a new set that is the intersection of l and r (see 6.4.4 for details).

6.8.5.1.10 .diff()

`l.diff(r)` returns a new set that is the difference between l and r (see 6.4.5 for details).

6.8.5.1.11 .add()

The .add() method is used to add elements to a set (see 6.7 for an example). When working with the f('qID')-function you should use the .union(set(…)) method instead (see 6.4.3 for details).

6.8.5.1.12 .remove()

The .remove() method is used to remove elements from a set. When working with the f('qID')-function you should use the .diff(set(…)) method instead (see 6.4.5 for details).

6.8.5.2 General utilities

6.8.5.2.1 SendMail(from, to, subject, body)

SendMail can be used to send an email-message using CDONTS

Arguments:

from:	Message sender
to :	Recipent's address.
subject:	Message subject line
body:	Message body

Example: This is an example used in a script node at the end of a questionnaire for registering to a seminar.

```
SendMail ("marketing@confirmit.com",f("contact")["e-mail"],
"Confirmation - Expo Media 2000",
"Dear "+f("contact")["name"]+","+"\n"+"\n"
+"Thank you for visiting our microsite and registering for the Expo Media 2000. This is a confirmation that you wish to attend this seminar:"+"\n"+"\n"
+f("seminar").label()+"\n"+"\n"
+"Best Regards,"+"\n"
+"Marketing Executive"+"\n"
+"Future Information Research Management (FIRM) AS"+"\n"+"\n"
+"--------------------------------"+"\n"
+"-- Email powered by confirmit --"+"\n"
+"--------------------------------"
)
```

In the example information about the recipient is collected from a multi question with the open text property selected and information about the seminar is collected from a single question. "\n" means a line break

6.8.5.2.2 Filter(qID,prefix)

Returns a set of precodes whose labels are prefixed by the supplied prefix. The Filter(qID,prefix) is useful when you have long answer lists.

Example: You ask the respondent to enter the first few characters in an Open question (q1). Then you include a precode mask in the next question (q2):

```
Filter('q2',f('q1'))
```

Now, the answer alternatives in q2 will only include the ones with the supplied prefix, entered by the respondent in q1.

7 Preparing for Data Collection

7.1 Generating Response Databases

When you are done with a questionnaire, or you want to test the questionnaire, you must prepare a response-database for your survey. Every time you make changes in the questionnaire, you have to update the response-database. This process is called compilation.

NB: The system keeps two separate databases for your survey, **one test-database**, and **one production-database**. You generate the test-database from *Testing → Compile database.* The production-database is generated from *Production → Compile prod. env.*

When clicking *"Compile ..."* for the first time, you will be presented with a dropdown dialog, asking you to confirm the creation of the initial database. If you click *OK*, a new database will be created and compiled. This may take some time, depending on the length of your questionnaire.

During compilation, the system performs some checks to ensure the integrity of the questionnaire. For instance, if you have non-unique form ID's in the questionnaire, the system will stop the compilation with an error message. The system also performs a check on whether the precodes of the Single and Grid questions do not occupy more space than allotted to them in Field Width. (See 5.2.1.4 Properties).

If the database already exists because of a previous compilation, you are presented with a slightly different dropdown dialog. You can choose between:

- Update existing database (incremental) – the system will update only the most recently made changes.
- Update existing database (rebuild) – the system will update the whole database, i.e. it will run through the whole questionnaire.
- Create new database – the system will create a completely new database. All existing old data will be deleted.

.

Most of the time, you should only update the database. This will preserve all data (responses, respondents, quotas, etc.) in the database, and is quicker. In a test-phase, however, it may be useful to start from scratch by recreating the database. **WARNING: This will delete all data in the database. You should NEVER do this in a running production-database.**

Login only:

To avoid unnecessary storing, FIRM does not guarantee that test databases are saved on the server for more than two weeks.

7.2 Generating Interview Files

Before you can start interviewing, you must have generated a production response database as described above.

Login only:

For generation procedure for CATI files, check separate manual.

7.2.1 Web Interview Files

The web interview generator produces a file capable of

1. Displaying forms for user input over the web,
2. Storing the respondent's answers, and
3. Controlling questionnaire flow based on background data (if available) and answers received.

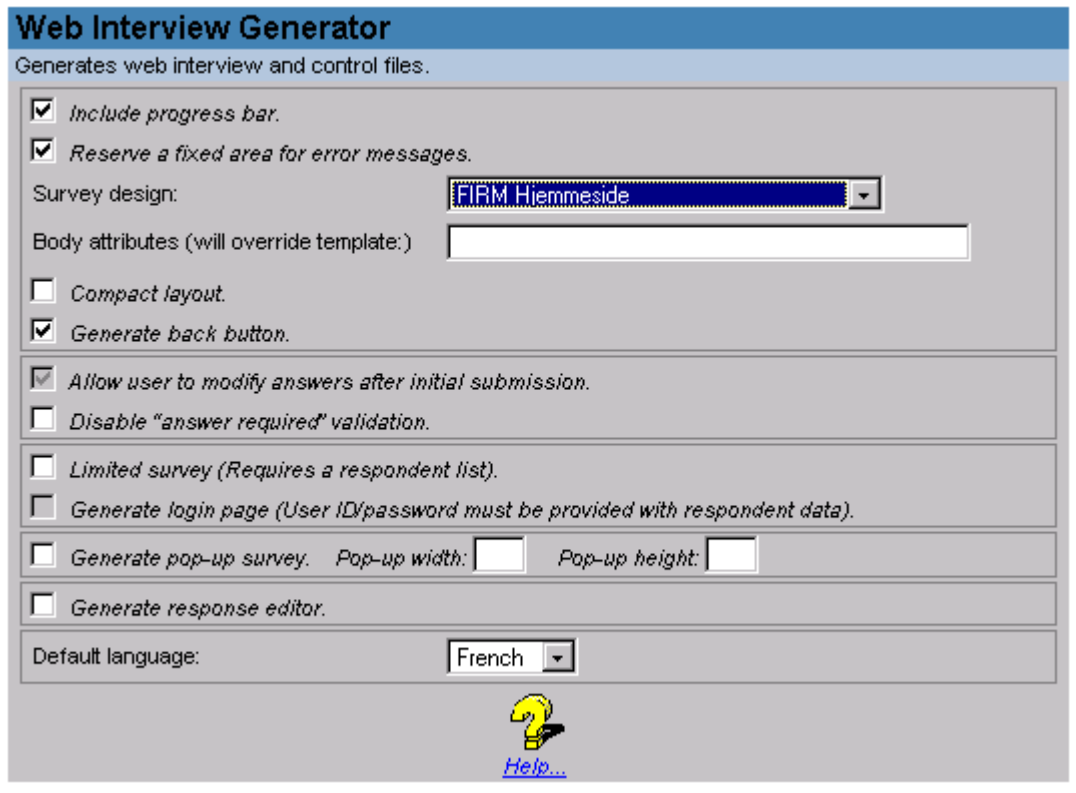
A description of the available options is provided below.

The URL to the survey will always be **http://www.yourconfirmitserver.com/wi/pXXXXXXX/i.asp**

Login only: The URL to the survey will always be **http://www.confirmit.com/wi/pXXXXXXX/i.asp**

If you need to generate new WI files for projects created before the 21st of February 2000, we recommend that you use the Old WI Generator.

For projects initiated after the above-mentioned date, you may use the New WI Generator.



Login only: Figure 35: Web Interview Generator options (Old)

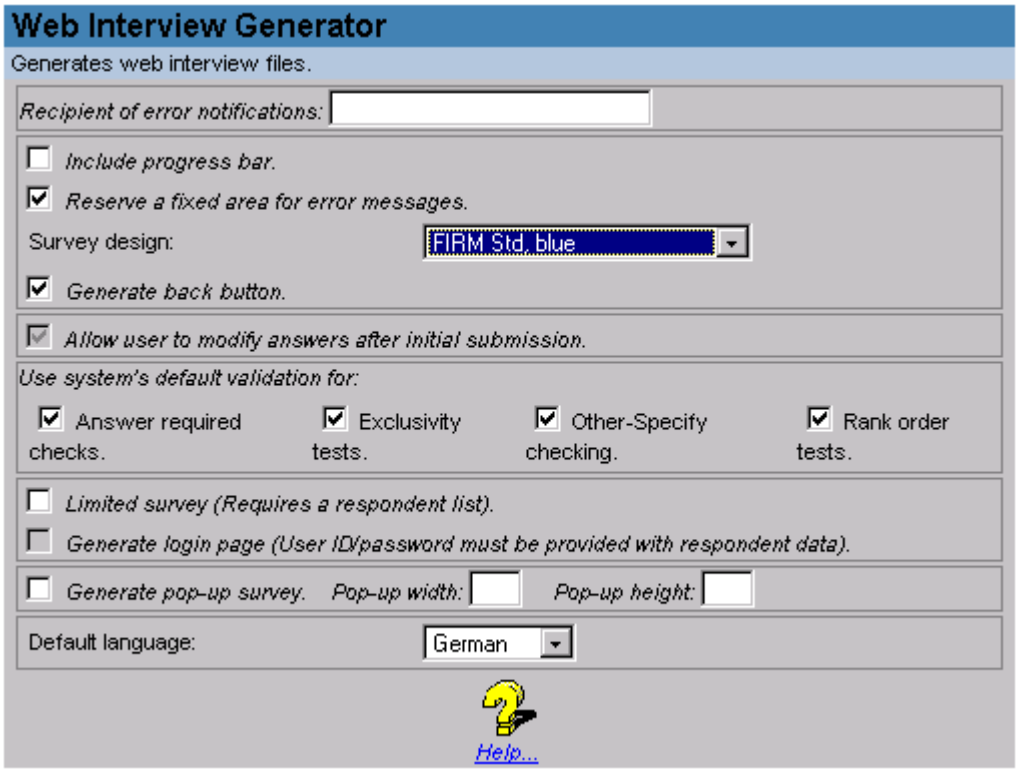


Figure 36: Web Interview Generator options (New)

7.2.2 Options Controlling Appearance

7.2.2.1 Recipient of Error Notifications

In this field you can enter an email address you wish to receive survey error notifications to.

Whenever there is an error in the survey, the respondent will get a general error message:

“Sorry, but the system has encountered an internal error and cannot go on with the interview at this stage. Please check back later.”

The addressee of the email address entered in the WI Generator, on the other hand, will receive a detailed description of the error and its whereabouts, e.g.:

**** Error info ****

Date : Fri Apr 28 01:06:58 UTC+0200 2000

Project : pXXXXXXX (The Name of the Project)

Respondent: 1

**** Error description ****

Problem encountered while processing question 'q22':

Error in mask:

f("q21").add("99")

Object doesn't support this property or method

The error message specifies the project ID, the title of the project, the question, and the erroneous script, precode mask, condition, and etc. expressions.

7.2.2.2 Include Progress Bar

This option instructs the generator to include a progress bar on the interview pages. This bar shows the respondent how much of the questionnaire is remaining (in percentage).

7.2.2.3 Reserve a Fixed Area for Error Messages

Select this option if you wish to reserve a small fixed area for error messages just below the page header. This prevents the rest of the page from jumping up and down if the interview program displays an error message. Because this area consumes space, you may want to uncheck this option, especially if you are creating a pop-up survey where the window size is limited.

7.2.2.4 Survey Design

In this field you may choose among WI templates you have predefined (see 3.6 Templates) and apply them to your project.

Login only:

7.2.2.5 Body Attributes.

This option lets you override the HTML body attributes specified in the chosen template (to change the background color etc.)

Use of this option is deprecated. The preferred method of changing the survey's design is to create a new template with the template editor.

7.2.2.6 Compact Layout

This option controls the way in which the questions are displayed. By checking this, they are displayed one after the other using the entire page width instead of one per line.

7.2.3 Options Affecting Interview Behavior

7.2.3.1 Generate Back Button

Back buttons allow the user to step backwards in the questionnaire and review/modify previous answers. This option is selected as default. It also forces the *Allow user to modify previous answers...* option described below to true.

7.2.3.2 Allow Modification of Previous Answers

This option controls the way in which the generated interview handles reentry to a questionnaire page. Reentry can occur if a respondent uses the browser's back button, or if he/she returns to the questionnaire through using the URL received from FIRM requesting participation in the survey.

If the survey should be "one-shot", then do not select this option. The generator then modifies the questionnaire flow according to the following scheme:

- If a respondent has completed the questionnaire and attempts to return to it, then the interview program displays a "Survey already completed" message.
- If the respondent returns to a partially completed interview, then a "You have already completed parts of the questionnaire. Press OK to continue" message is displayed.
- If it is detected that the browser's back-button has been used, then a "Previous answers may not be modified. Press OK to continue" message is displayed.

7.2.3.3 Answer Required Checks

The generator produces code that ensures that the respondent answers a question unless it has the "Not required" property set. This option applies to all Open, Single and Grid questions, and Multi questions with an exclusive answer alternative defined as single punch.

7.2.3.4 Exclusivity Tests

The generator produces code that ensures that the respondent has answered the exclusive answer alternative defined as single punch in a Multi question, without checking additional answer alternatives in a list.

7.2.3.5 Other – Specify Checking

The generator produces code that confirms the consistency of user input, i.e. that the respondent does both, checks off for Other and writes in the text box,- and vice versa.

7.2.3.6 Rank Order Tests

The generator produces code that ensures that the inputs in Multi Ordered questions and Grid questions are unique for each alternative in the list and consecutive. The questions must be marked as Ordered in Properties (see 5.2.1.4 Properties).

Login only:

7.2.3.7 Disable "Answer Required" Validation

By default, the generator produces code that ensures that the respondent answers a question unless it has the "Not required" property set. This option is actually a "dirty" shortcut, which turns off validation for the entire interview, without consideration of this form property.

7.2.4 Options for Generating Limited Surveys

7.2.4.1 Limited Survey

The generator supports two kinds of surveys: limited and open. Limited surveys can only be accessed by respondents identified in a respondent list. Each respondent may automatically be provided with a unique link to the survey, which includes identification credentials. Alternatively, he has to identify himself on a login page.

Open surveys may be answered by anyone who desires to do so. As such, they are suitable for surveys where the respondents' identities cannot be established in advance, e.g. Web site evaluations.

7.2.4.2 Generate Login Page

As mentioned, **confirmit** can provide a unique link for each respondent in a limited survey. As an alternative, the survey may include a login page where the user enters a user name and a password to participate. Check this option if you want this mode of operation. Remember to include user name/password information in the respondent lists. See 8.1Preparing the Respondent List.

7.2.5 Pop-Ups

7.2.5.1 Generate Pop-up Survey

Normally, an interview is ended by displaying a "Thank you" box that brings the user to a specified page when the OK button is pressed. This option alters the behavior of the generated interview to close the browser window when the interview is complete.

7.2.5.2 An Example of a Script for Pop-up Surveys

If you want to do a pop-up on a website, a script will have to be included in the code of the page where you want the pop-up to be opened from. There are many different solutions for this, depending on when you want the pop-up to come up etc. Below is an example of a script you could use.

```
<script>
if (document.cookie.indexOf ("FIRMpXXXXXXX=true")==-1){
      if ( Math.random() < 0.2){
            document.cookie = "FIRMpXXXXXXX=true;path=/;Expires=Mon,
1 Jan 2001 12:00:00 UTC"
      window.open("http://www.yourconfirmitserver.com/wi/pXXXXXXX/i.a
sp", "windowname",
"toolbar=no,location=no,directories=no,status=no,menubar=no,resizable
=yes,copyhistory=no,scrollbars=yes,height=360,width=300")
      }
} </script>
```

Change pXXXXXXX to the actual project number.

The script first checks if the respondent has a cookie already (then he/she has already gotten the pop-up and it should not be opened again). Math.random gives a random number between 0 and 1. Math.random() < 0.2 will mean that 20% of those who open the web page (for the first time) will get the pop-up. document.cookie will send a cookie. Expires: Lifetime of the cookie. Set this to a date some time after the survey is finished.

Window.open opens the pop-up window. Change the url to the url of your survey. windowname may be changed into a name of a specific window if there already is a window opened that you want the pop-up in, or you may use it to refer to the window from other scripts. You may leave this empty ("").

The other parameters control the pop-up window:

toolbar=no- allow the window to have or not have a toolbar.

location=no - allow the window to display its location or not.

directories=no - allow the window to display directory buttons or not.

status=no - allow the window to have a status bar at the bottom or not.

menubar=no - allow the window to display a menu bar or not.

resizable=yes - make the window resizable, boarders are dragable.

copyhistory=no - allow for tracking of links activates the back or forward buttons.

scrollbars=yes - automatically include scrollbars on the window if needed.

height=360 - height of the window in pixels.

width=300 - width of the window in pixels

7.2.5.3 .Pop-up Script Wizard

In Test modus, after having generated a web interview, you are presented a link, e.g. English, to the test survey. When you enter the link to a survey that has been generated as a Pop-up, you will immediately be presented with a Pop-up Script Wizard. The Wizard includes an example of a Pop-up script, very similar to the one included in 7.2.5.2, as well as it gives the user an opportunity to adjust several of the Pop-up parameters: window appearance, display frequency, and cookie.

The changes you make to the Pop-up script will be applied the next time you generate web interview in the Test mode.

7.2.6 Default Language

This option is only displayed for multi-language surveys. The language selected here becomes the default used when no language information is available in the request used to access the interview or in the respondent list.

7.3 Project Status

You can set the project status on this page.

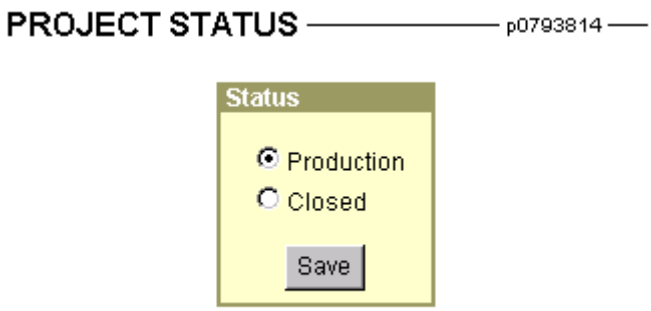


Figure 37: Project status

Production: The survey is running.

Closed: The survey has been closed.

7.4 Project Log

The system records every action executed on a project basis. In *Production* > *Project Log* you will have an overview of all changes that have been made in the project. The system records the exact time when changes are made, the user who makes the changes, and specifies every change.

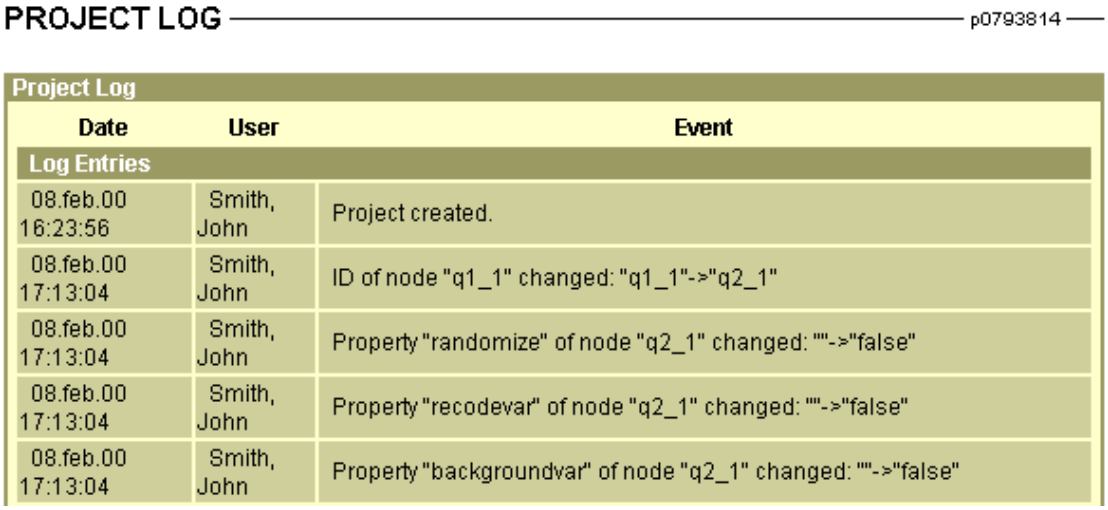


Figure 38: Project log

7.5 URL Setup

You have a possibility to control which website the respondents will come to, when they have finished the survey. When you click *Project* > *Production* > *URL Setup*, you will be brought to this page.

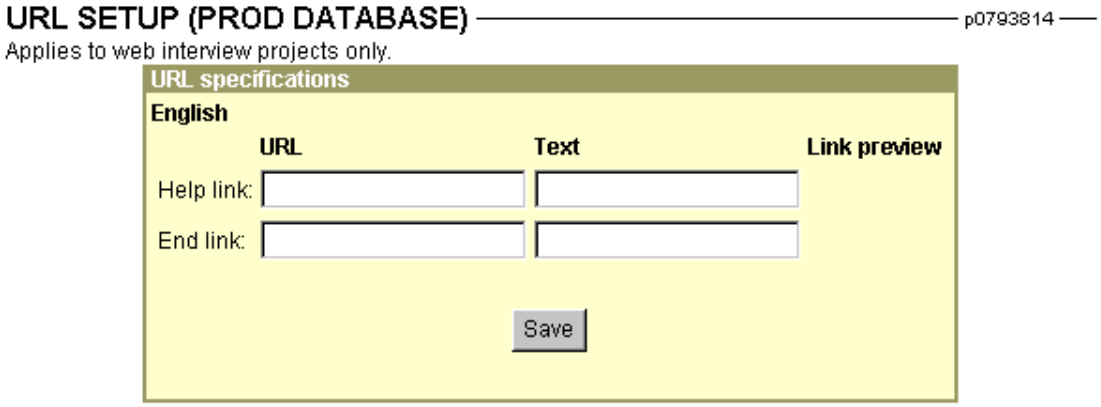


Figure 39: URL setup

Field	Description
Help link	In this field you enter a URL address to a website you wish to use either for additional information or for another purposes in the questionnaire. In this case you

	have to have included primitive ^ HELP^ in the WI template you will apply to this survey.
End link	In this field you enter the URL address of the website you wish the respondents to be brought to after they have finished the survey.
Text	In this field you may control the appearance of the link, i.e. you may substitute the address with another text.
Link preview	In this column you will get a preview of the link.

Table 25: URL setup

Usually, the End link must be defined between the processes of Compilation and Generation of WI: *Compile prod. Env. > URL Setup > Generate WI.*

You will have to Generate WI anew in case you make any changes in *URL Setup*.

8 Handling Respondents in Limited Surveys

For targeted, non-public web interview surveys, it is necessary to prepare and upload a list of respondents. This chapter describes the necessary steps needed to prepare the respondent list and upload it. Chapter 8.3 also describes how to automatically send e-mails with invitations and reminders for participants of web surveys.

8.1 Preparing the Respondent List

The respondent list must be in the form of a tab delimited text file, with column headings in the first row. This file can for instance be exported from a spreadsheet application such as Microsoft® Excel™ by using the *Save As* function and choosing the format *Text (Tab delimited - .txt).*

This data file is prepared on the user's computer and is subsequently uploaded into **confirmit**.

The respondent data file must contain some or all of the following columns, depending on the type of survey. **Make sure that the spelling is exactly the same as in the table below.**

Column name	Description	Limited WI
name	The full name of the respondent	opt
language (for multilingual projects)	The respondent's preferred language. The survey will be presented in this language (see appendix for language codes)	opt
email	The respondent's e-mail address	req
userid	The respondent's userid if logon is required	opt
password	The respondent's password if logon is required	opt

req = required opt = optional

Table 26: Fields in respondent lists

In addition to these columns, any background variables defined for the survey must be represented as columns in the respondent data file. **The column names for these background variables must exactly match the form IDs given to these background variables in the questionnaire designer. See 5.2.1.4 Properties.**

8.2 Uploading the Respondent List

Once the respondent data file has been prepared and saved as a tab delimited text file, it is ready to be uploaded into **confirmit**. This is accomplished by selecting the menu *Respondents/Uploading.* Simply press the *Browse…* button, which will invoke the familiar Windows™ File Open dialog, allowing the user to browse through the file system and select the appropriate respondent data file and press the *Open* button in the file dialog.

Once the correct data file has been selected press the *Upload…* button to start uploading the respondent list to the **confirmit** system. After the file has been successfully uploaded, the user is presented with three options:

- **Yes, append:** This option will append the newly uploaded respondents to any existing respondents already in the database.
- **Yes, overwrite:** This option will delete any existing respondents in the database before adding the newly uploaded respondents. **Use this option with caution.**
- **No:** This option will abort uploading respondents and return to the *Project overview* page.

If uploading is confirmed, a status screen will appear after a few seconds, summarizing the number of respondents that was uploaded.

8.3 Sending E-mail

confirmit provides a powerful wizard for sending e-mails to web survey respondents. On the basis of several selection criteria, e-mails can be sent to all or a subset of the respondents in order to invite them to participate in a web survey or to remind them of a previous invitation. It is even possible to send "thank you" e-mails to respondents that have completed a web interview.

The e-mail wizard is accessed through the *Respondents > Emailing* menu. The wizard will take the user through the following steps:

1. **Respondent selection:** Select which respondents should receive the e-mail and specify the content of the e-mail.
2. **Preview and confirmation:** Preview which respondents will be mailed and how the e-mail will look.
3. **Sending e-mail:** Send the e-mails to the selected respondents.

More on these steps in the following sections.

8.3.1 Step 1: Respondent Selection

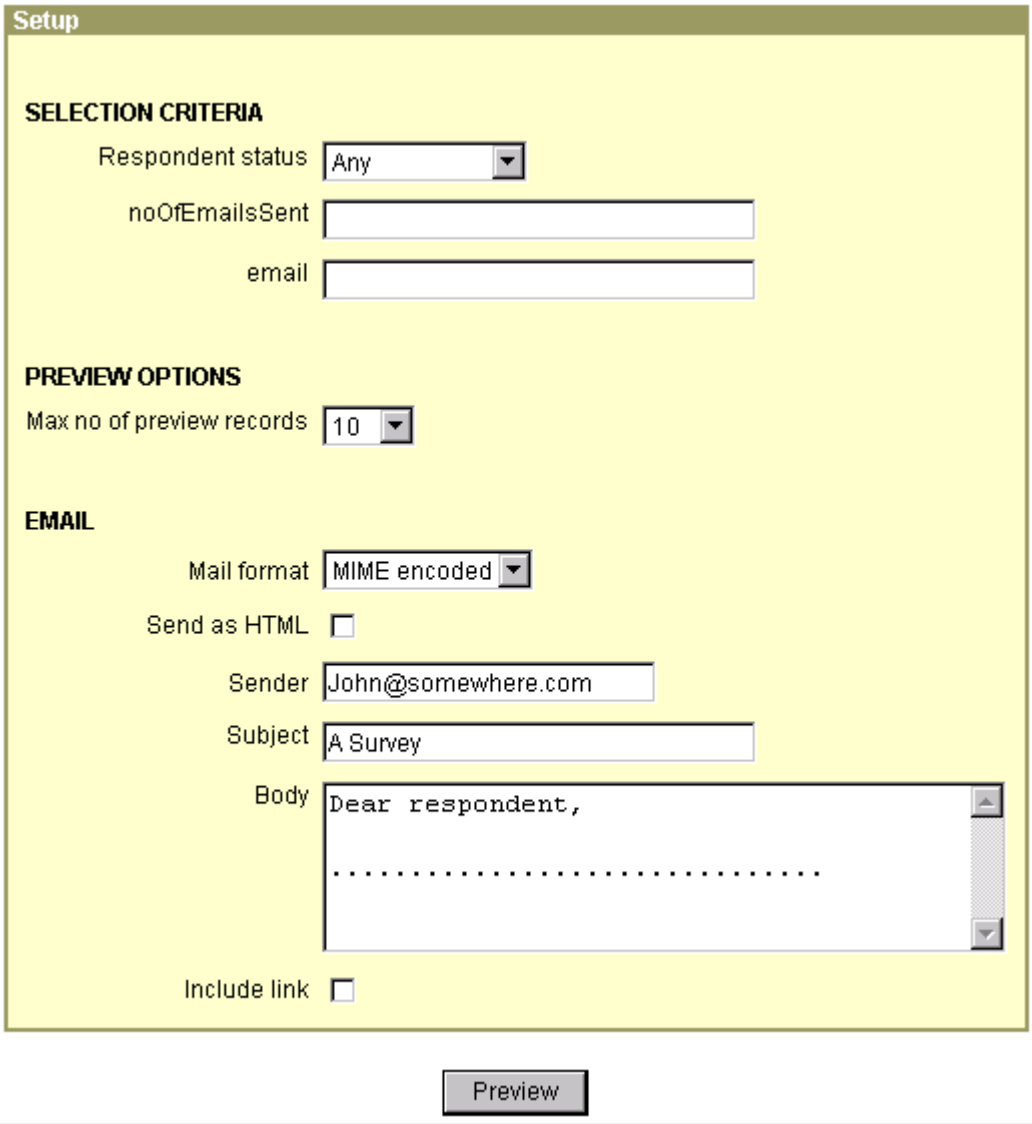


Figure 40: E-mail setup

Please note that this screen may vary depending on the fields you have included in the respondent file that you are using. All fields those will be presented under *selection criteria.*

In this step, the user enters the selection criteria that will determine which respondents will receive the e-mail. The criteria are:

Criterium	**Description**
Respondent status	Chooses respondents based on the status of the interview, see table below.
Number of emails sent	The number of e-mails sent is recorded for each respondent, and can be used as a selection criterion.
Other fields: eg "email"	The wizard will display any additional fields from the respondent file that may be used for selection, including sample categories and quota variables.
Status	**Description**
(Any)	All respondents
Not answered	All respondents who have not entered the survey
Complete	All respondents who have completed the survey. (Useful for sending "Thank you" e-mails).
Incomplete	All respondents who have started answering the survey, but have not completed it.
Screened	All respondents who have screened status (See 5.2.4 Stop-objects).
Quota full	Not in use yet.

Table 27: Respondent selection criteria/status

To make the selection abilities as flexible as possible, the selection criteria must be input in the *Structured Query Language (SQL)* syntax known from relational database queries.

Examples:

Assume that the respondent data includes gender as a background variable with precodes 'M' and 'F' for male and female respondents, respectively. The e-mail wizard would then list *gender* as one of the possible selection criteria. To select only male respondents one would enter

```
= 'M'
```

in the *gender* input field. Selecting based on a set of values would be done with the *IN* keyword. Selecting both male and female respondents (for the example's sake) is done with the criteria

```
IN ('M', 'F')
```

in the *gender* input field. Selecting respondent that have already received more that 2 e-mails could be accomplished by entering

```
> 2
```

in the *noOfEmailsSent* input field. Similarly, all standard SQL keywords and operators can be used, such as *BETWEEN*, <, <=, >, >=, <>, *AND*, *OR*, *LIKE* etc. For instance, to find all respondents whose names start with 'B' (assuming the respondent list includes name as a background variable):

```
LIKE 'B%'
```

Under *Preview option* the user can enter the maximum number of respondents that will be displayed in the preview list in step 2 of the e-mail wizard. This is merely a precaution when working with large numbers of respondents.

Under the *E-mail* heading the user can enter details about the e-mail he wishes to send to the selected subset of respondents:

Field	Description
Mail format	Choose between sending a MIME encoded (default) or plain text e-mail. Normally MIME will work best on most e-mail programs.
Send as HTML	Emails may be sent as HTML codes. Check the "Send as HTML" box to send HTML coded email invitations. Comment: Only a few mail systems will display such emails correctly. Use with caution. When used, "MIME" format should also be selected.
Sender	Enter the e-mail address that is to appear in the *From* header field in the e-mail.
Subject	Enter the subject header of the e-mail.
Body	Enter the text that is to appear in the body of the e-mail message.
Include link	Select this checkbox to automatically have a link (URL) to the web interview at the bottom of the e-mail (default).

Table 28: Email setup fields

8.3.2 Mail Merging Functionality

To merge a background data field into the body text, for instance the respondents' e-mail address, write:

^email^, referring to the uploaded email column (must be written exactly as the column label).

It is possible to display the link to the survey within the e-mail text by simply referring to it in the following way:

- ^slink^

Make sure that:

- "slink" is not used as background data;

The reference ^slink^ should always be written on a separate line in order to avoid breaking the link over two lines.

8.3.3 Avoiding Unnecessary Respondent Inquiries

In order to avoid unnecessary failure messages and inquiries from respondents when running limited Web-surveys, we strongly recommend including the following information in the e-mail body text:

Please, read the following before you enter the survey!

- Cut (Copy) – and – Paste the link into the Internet address window in case you cannot access the survey by simply clicking on the link. Make sure it's on one single line and without any spacing.
- Navigate forth and back by clicking on the >> and << buttons, or just >> if the back button is disabled. Do not use the browser's "back" and "forward" buttons, or your answers may not be saved!
- If you are not able to switch to the next page, you might have forgot to answer all questions on the page. Look for an error message.
- Use the link in the e-mail message when you wish to continue responding to the survey after a pause. Do not bookmark the survey page!
- The link is unique! Please, do not send it to other respondents.

By pressing the *Preview* button, the user is taken to step 2 of the e-mail wizard.

8.3.4 Step 2: Preview and Confirmation

In this step, a preview list of respondents shows which respondents will receive an e-mail, based on the selection criteria entered in step 1. The maximum number specified under Preview options in step 1 limits this list.

In addition, a preview of the e-mail that is to be sent is shown.

At this point, the user can choose to cancel the e-mail wizard altogether and return to the *Project overview* page; he can choose to go back to step 1 to edit the selection criteria; or he can choose to proceed with sending e-mails and go on to step 3 of the wizard.

8.3.5 Sending E-mails

This is the final step of the e-mail wizard. At this point the e-mails will be queued for immediate delivery to the selected respondents.

When sending emails, a standard check will be run on the email list. The message will not be sent to invalid addresses. The **confirmit** user will receive a mail from the system on how many emails have been sent, and how many mails it has failed to send.

A confirmation from the system will be sent to the address you have specified in your own User Settings, even if you have specified another sender address in the *sender* field.

8.4 Handling Respondents in Multilingual Projects

In multilingual projects it is necessary to ensure that the respondents get the survey in correct language. There are several methods for controlling this for Limited and Open surveys.

8.4.1 Open Multilingual Surveys

There are two ways of securing that respondents get the survey in a particular language:

- In open multilingual surveys you may have a Single question with the question ID ***l*** (as in language) as the first question of the survey. You will have to use the actual languages as answers, and the language codes (see Appendix B: **confirmit** Language Codes) as precodes.

  The respondents will get the next question in the language they chose in the Language question.

- Another option is to distribute survey links specifying language, to the countries the survey will be conducted in, e.g.:

  http://www.yourconfirmitserver.com/wi/pXXXXXXX/i.asp?l=0 (for English)

  http://www.yourconfirmitserver.com/wi/pXXXXXXX/i.asp?l=6 (for Spanish)

  etc.

8.4.2 Limited Multilingual Surveys

When it comes to limited surveys, you may:

- Include a column with a tag *language*, which specifies the language each respondent will get the survey in (see 8.1 Preparing the Respondent List). The contents of the column must be **confirmit** language codes (see Appendix B: **confirmit** Language Codes).
- The other option is to include a Single question with the question ID ***l*** in the survey, as mentioned above (see 8.4.1 Open Multilingual Surveys).
- In multilingual surveys with login-page (see 7.2.4.2) the respondents are presented with a drop-down where they can choose language.

9 Reporting

9.1 Introduction

All reporting functionality in **confirmit** is accessed by clicking *Reports* in the main menu. Click on the *Online Reporting* and you will enter this screen:

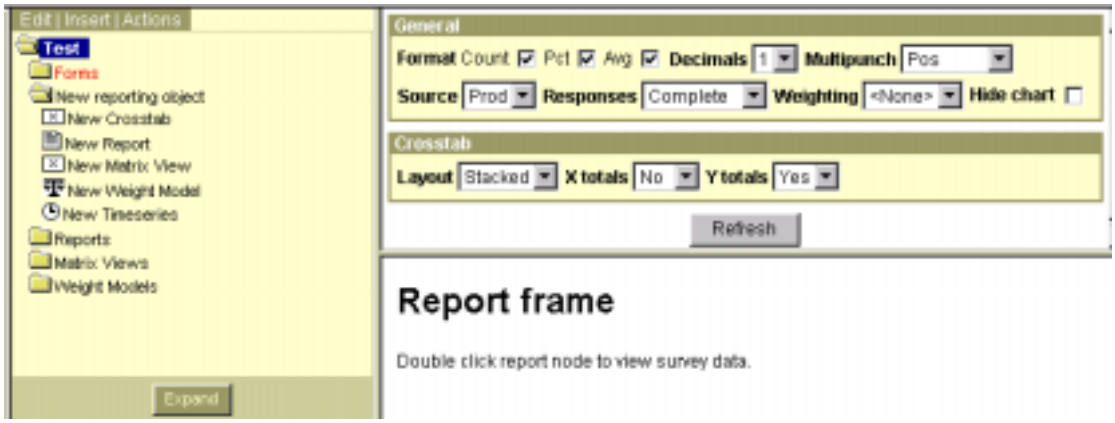


Figure 41: Initial reporting screen

This screen consists of three frames:

- **The reporting tree.** A tree containing all your reporting objects. That is forms, new reporting objects, reports and matrix views. The menu on top of the tree is called the report tree menu.
- **The settings frame.** A frame with General settings for the reports. These General settings can be overridden by setting specific properties for the report nodes. To get more space on the screen for tables and charts, the general option frame can be toggled. Toggling is done from the reporting tree menu: 'Edit->Toggle options'.
- **The report frame.** The frame where all report tables and charts are displayed, or the properties of the report node when chosen.

You may expand all trees by clicking the *Expand* button.

9.2 Checking a Single Question

The easiest way of checking the answers of a single question is by entering the *"Form"*-folder in the reporting tree:

1. **Open the *"Form"*-folder.** Single-click on the folder icon. This will show you the routing the same way as the *"Routing"*-folder in the questionnaire.
2. **Locate the question.** Since the subfolders, condition trees and loops are all initially closed, it is crucial to know something about the questionnaire routing to be able to find the question of interest.
3. **Double click the question.** When the question is double clicked, the results are shown in the bottom right frame as shown in the Figure 42.
4. **Expanding the Other-specify fields.** By clicking the icon of a form containing 'Other' answer alternatives, the form will be expanded. The sub-tree of the form will contain the open-text fields. These can be reported and dragged into reports just as any other open-text forms.
5. **Expanding Grids.** Single-click on the icon of a grid question to expand it. All the grid elements will be shown as single questions.

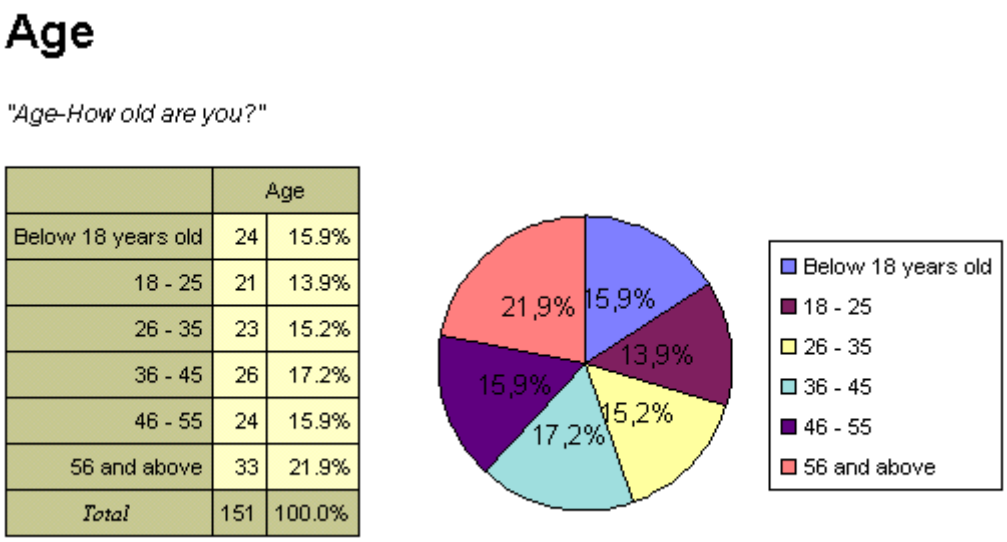


Figure 42: Reporting a single question with Excel Chart Component

When checking results directly from the *"Form"*-folder, all settings are either default settings or the settings from the *"Settings"*-frame.

9.3 Building Reports

Within a project, all the reports are collected in the report folder. The idea by having more than one report is that you can prepare reports for different target groups and order exports (ppt/Excel files) separately. The following actions are possible to take at the report level:

- **Create a new report.** A new report is created by dragging a *New Report* object from the *New reporting objects* folder into the *Reports* folder.
- **Change name.** Select *Edit → Properties* in the report tree menu, and change the report name in the report frame.
- **Answer mask for reports.** You can select the answers (for a single question) and the answers and scales (for a grid question) you want to be shown in the report. This masking can be executed for specified question forms in a report. You just select a question in a report you want to mask and go to *Edit→Properties* in the report tree menu. In the report settings in *Answer mask/Scale mask* field you may select the answers. See table *Report elements settings* in chapter 9.8.2 for details.
- **Adding questions to the report.** Drag the form from the *Forms* folder and drop it at the right location within the report.
- **Add cross tabulation to the report.** Drag a *"New crosstab"*-object from the *New reporting objects* folder and drop it at the right location within the report. See chapter 9.4 for details on building crosstabs.
- **Add time series to the report.** Drag a *New timeseries* object into the report, e.g. within a crosstab. Select the *New timeseries*, and then go to *Edit → Properties* from the report tree menu. Afterwards configure the time series with granularity and start/end points.
- **Delete.** Select *Edit → Delete* in the report tree menu. Can be applied to both, individual questions/crosstabs and to whole report folders.
- **Duplicate.** Select *Edit → Duplicate* in the report tree menu.
- **Showing the whole report.** Select *Actions → Show Report* from the report tree menu. This will display the whole report in the report frame.
- **Order report.** Select *Actions-Order Report* from the report tree menu. By specifying a PowerPoint template and email address in the report frame, Excel sheets and/or PowerPoint slides

will be sent to the email recipient. In User Settings you may specify in which MS Office version you wish the report to be sent.

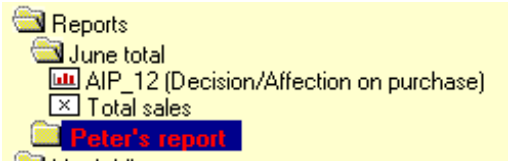


Figure 43: Project with two reports.

The charts in PowerPoint reports are saved as pictures. This means that it is not possible to make any changes to ordered PowerPoint reports. If you wish to make changes to the charts in ordered reports, we recommend that you order Excel report, make your changes and prepare a PowerPoint presentation afterwards.

Figure 43 shows a report, *June total*, with one crosstab and one single punch question.

9.4 Building Crosstabs

When a new crosstab is created within a report, the crosstab object includes two subfolders:

- The *"Crosstab..."*-folder where the vertical questions of the crosstab can be included. This is done by dropping questions within the folder.
- The *"With..."*-folder where the horizontal questions of the crosstab can be included. This is done by dropping questions within the folder.

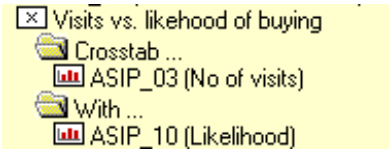


Figure 44: Crosstab: No of visits vs. likelihood

9.5 Reporting on Loops

The easiest way of checking the answers in a Loop is by entering the *"Form"*-folder in the reporting tree:

- **Open the *"Form"*-folder.** Single-click on the folder icon. This will show you the routing the same way as the *"Routing"*-folder in the questionnaire.
- **Locate the Loop.** Since the subfolders, condition trees and loops are all initially closed, you should know something about the questionnaire routing to be able to find the question of interest.
- **Single-click the Loop folder.** When the Loop is single-clicked, the questions the Loop contains will appear. If you single-click on the form node, all iterations for the loop question will appear. When you double-click on each iteration, the results will be shown for only the particular iteration (see Figure 45).

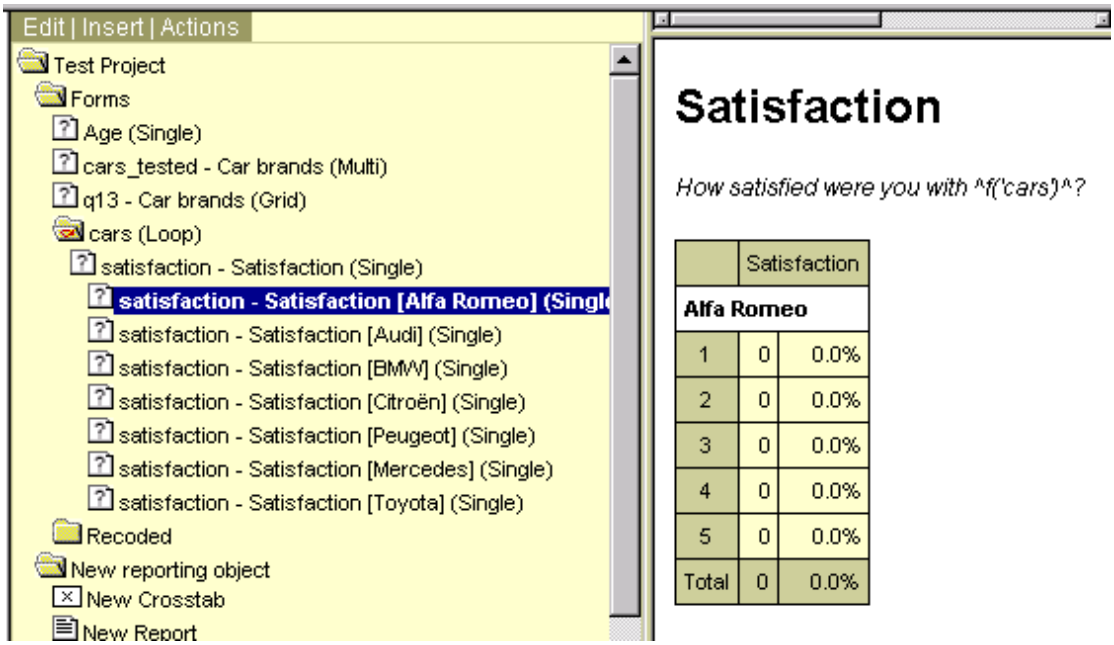


Figure 45: Showing results for single iteration

- **Adding loops to reports.** You can add a loop question to the report by drag and drop method. You single-click the loop folder in *Forms*, and all the questions in the loop will appear. Select the question you wish to report on, and drop it into the *Report* folder.
- **Loops in crosstabs.** You may cross tabulate the questions in the loop with the loop. Insert a *New crosstab* into the *Report.* Insert the loop questions into the *Crosstab…* folder if you wish them to appear vertically, and the loop folder into the *With…* folder if you wish the iterations to appear horizontally; or you may do it vice versa (see Figure 46).

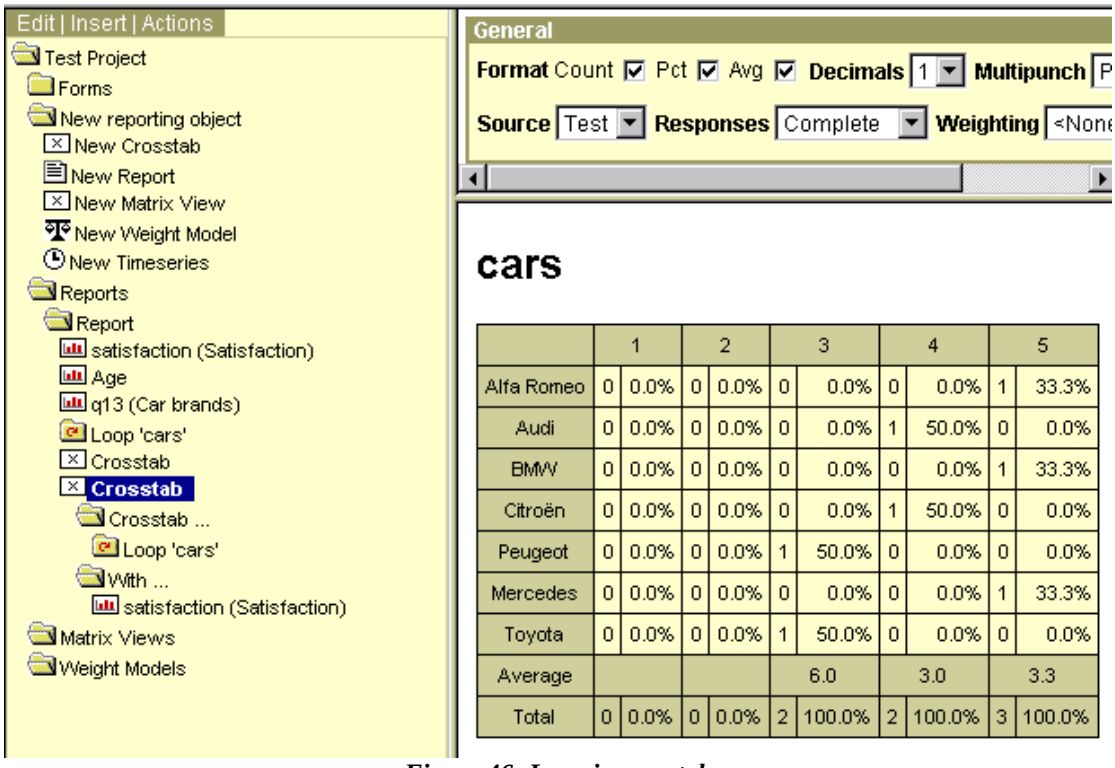


Figure 46: Loop in crosstab

For more detailed information on how to edit reporting elements of loops, see chapter 9.8.2 Report Elements Settings.

9.6 Weight Model

CONFIRMIT allows weight models to be applied to your reports. Three separate types of weight models may be used.

Weight Model Type	**Description**
Sample balancing	This method is used for weighing along multiple dimensions when only the marginal reference percentages along each dimension are known. E.g: It is known that the reference population is 30% male and 70% female and that 60% of the population own a car while 40% do not. From these facts the actual weights are calculated according to the standard sample balancing algorithm. Max iterations and deviation limit for this algorithm decide how accurate calculations are required.
Cell matching - percentage	This method is used for weighing along multiple dimensions when all the matrix cell percentages are known. E.g: It is known that: • 20% are male with cars • 10% are male without cars • 40% are female with cars • 30% are female without cars From these facts, the weights can be calculated by dividing the sample percentages by the reference percentages.
Cell matching – weight	With this method all the weights are entered directly and not calculated by comparing a reference with the sample. The method may for example be used if the weights are calculated by another system.

Create a new weight model by dragging the "New weight model" object into the "Weight models" folder. Double click the new object to edit its properties. The percentages for the cell matching methods are entered from this property screen.

Drag and drop the associated questions into the new weight model. For "sample balancing" (default), double click the individual questions in the weight model folder to set their percentages.

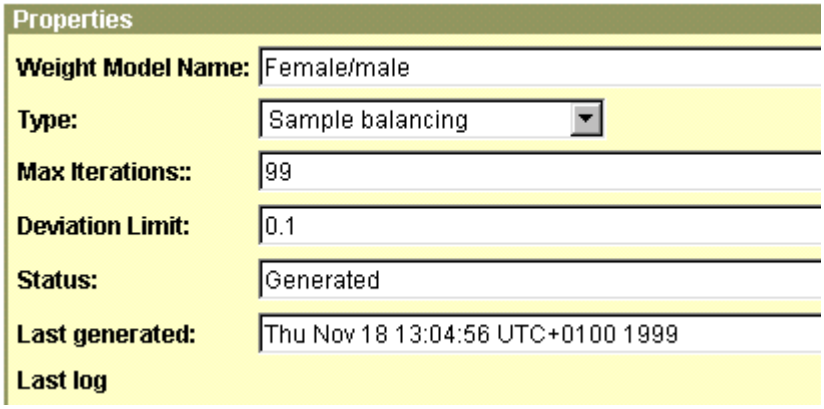
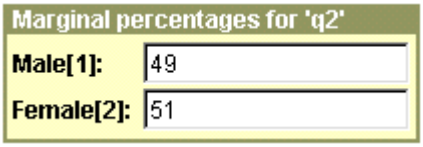


Figure 47: Weight model properties



Respondent weights are calculated when the user selects [Action => Calculate weights] after first having activated the weight model folder. Weight models may be applied to report folders and/or as a general report option. Report folder model overrides the generally selected model.

It is not possible to apply weighing on questions that were answered only by part of the respondents (filtered questions).

9.7 Time Series

Use time series to create trend reports.

1. Drag a 'New timeseries' object into the report, e.g. within a crosstab.
   2. Select the new time series and select 'Edit->Properties' from the report menu.
   3. Configure the time series with granularity and start/end points (day, week, month, quarter, year).

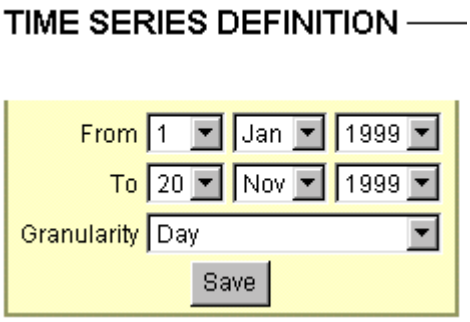


Figure 48: Time series properties

Time series may be used in cross tabulations or as a stand-alone report item.

9.8 Settings

9.8.1 General Settings

The settings frame in the reporting screen includes the following general settings for the whole reporting environment. Every time you make any changes in the General Settings you have to click the *Refresh* button in order to update the settings.

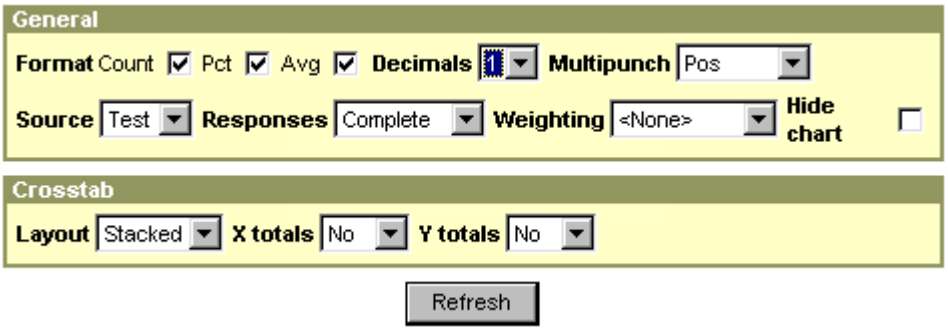


Figure 49: General settings frame

Setting	Setting description
General	
Source	Indicates whether test or production data should be reported.
Format	Indicates whether count, percent, and/or average should be displayed. Average can only be displayed for weighted single punch or grid questions. See chapter 5.2.1.2 for how to give values/weights to a question.
Decimals	Number of decimals to use in average and percent format.
Responses	By default set to show only complete interviews, but can also show only incomplete interviews or both complete and incomplete.
Multipunch	**Pos:** Shows only numbers of positive answers for a multi punch. **Pos + Neg:** Show numbers of positive and negative answers for a Multi question (answered/not answered) **Ordered:** Shows an ordered multi punch split in ranking.
Weighing	<None> or one of the defined Weight models
Hide chart	Use this check box to show only tables in the report frame.
Crosstab	
Layout	**Stacked:** The questions in the *"With..."*-folder are stacked. **Nested:** The questions in the *"With..."*-folder are nested.
X totals	Set this to *" Yes"* to get the totals for a question to the right of the rest of the crosstab.
Y totals	Set this to *" Yes"* to get the totals for a question under answer alternatives.

Table 29: General settings options

9.8.2 Report Elements Settings

All of the general settings can be overridden by setting properties for each single element within a report. The element would be a crosstab or a simple question within the report. This is done by selecting a report element, single-clicking on it, and then choosing *Edit → Properties* in the above the tree structure menu. From this menu point, you can also set the following properties that are not available from the general settings:

9.8.2.1 Excel Chart Component

The following screenshots will be available for you if you choose Excel as the chart component in user settings (see 3.2 User Settings).

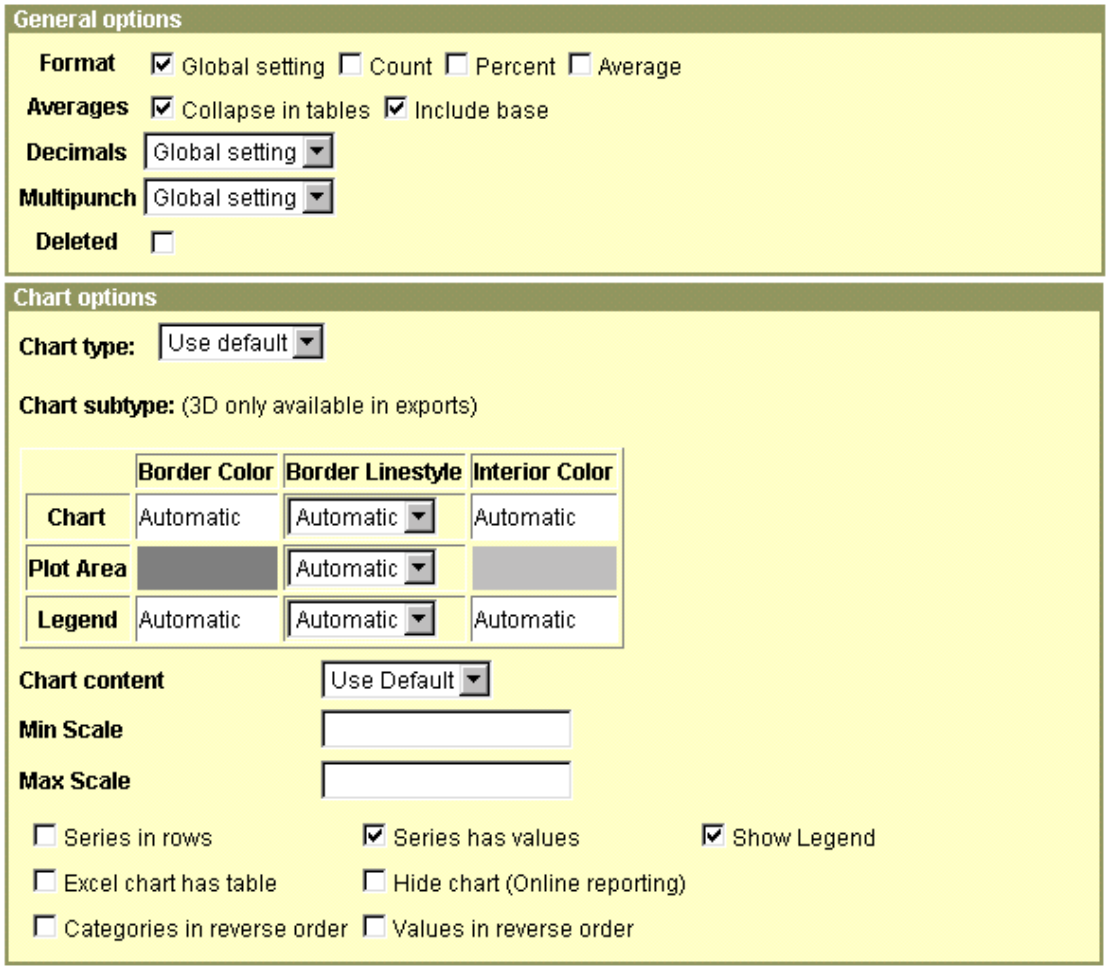


Figure 50: Chart options for Excel Chart Component

Chart Options Excel Component	
Setting	**Description**
Type	Defines the chart type to be used. The most normal chart types like bars, pies, lines and areas are supported among some other.
Subtype:	
Border Color	(chart and legend) The default border color of charts and legends is set to be black. The changes are invisible in **confirmit**. Nevertheless, they will be activated when ordering the report. Double-click on the fields in order to activate the Color Selection.
Border Linestyle	(chart, plot area, legend) The changes are invisible in **confirmit**. Nevertheless, they will be activated when ordering the report.
Interior Color	(chart and legend) Double-click on the fields in order to activate the Color Selection.
Chart content	Indicates whether count, percent, and/or average should be displayed. Average can only be displayed for weighted single punch or grid questions. See chapter 5.2.1.2 for how to weigh a question.
Min Scale	Indicates the minimum value on a scale.
Max Scale	Indicates the maximum value on a scale.
Series in rows	The graphical representations of the categories, e.g. bars or columns are placed next to each other, each category having its own legend.
Excel chart has table	The changes are invisible in **confirmit**. Nevertheless, they will be activated when ordering the report.
Categories in reverse order	If you check off for this option, the categories in the chart will be shown in reverse order as compared to the table, e.g. male – female in the table, but female – male in the chart.
Series have values	Table values of each category are explicitly specified in the chart.
Hide chart (Online reporting)	Use this check box to hide the charts in the report frame.
Values in reverse order	The values in the chart, i.e. count, percent or average, will be shown downwards and not upwards.
Show legend	You may choose whether you wish the legend to be displayed in the chart.

Table 30: Excel chart component options

9.8.2.2 First Impression Chart Component

The following options are available for you if you have chosen First Impression in user settings (see 3.2 User Settings).

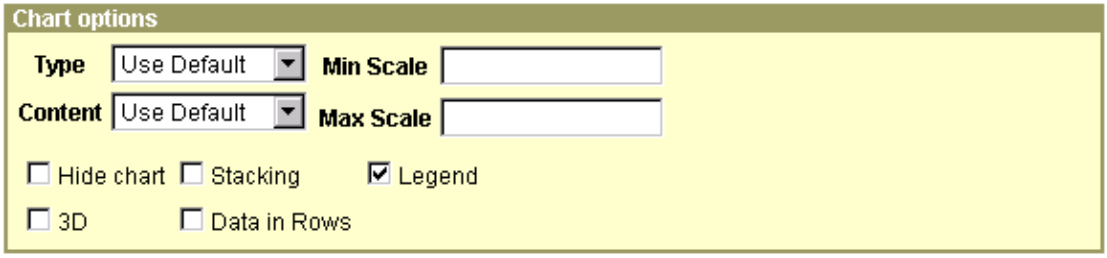


Figure 51: First Impression Chart options

Setting	Description
Chart options First Impression Component	
Type	Defines the chart type to be used. The most normal chart types like bars, pies, lines and areas are supported among some other.
Content	Indicates whether count, percent, and/or average should be displayed. Average can only be displayed for weighted single punch or grid questions. See chapter 5.2.1.2 for how to weigh a question.
Min Scale	Indicates the minimum value on a scale.
Max Scale	Indicates the maximum value on a scale.
Hide Chart	Use this check box to hide the charts in the report frame.
3D	The chart can be viewed in 3D.
Stacking	The chart is stacked.
Data in rows	Uses data in rows instead of data in columns. This is the terminology of the chart component used. Excel uses the opposite terminology: Series in columns and series in rows.

Table 31: First impression component options

9.8.2.3 PowerPoint Options

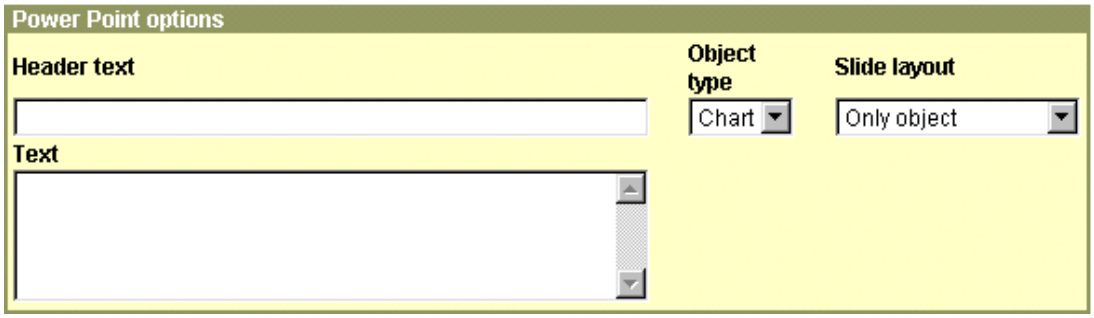


Figure 52: PowerPoint options

PowerPoint options	
Settings	**Description**
Header text	The header text in a PowerPoint slide. This text can also be set to give a name to a crosstab in the reporting tree.
Text	The text in text frames in PowerPoint slides.
Object type	Used to choose between table and chart as the PowerPoint object.
Slide layout	**Object left/Text right:** The object (table or chart) is in the left of the slide and the text is in the right of the slide. **Text left/Object right:** The text is in the left of the slide and the object is in the right of the slide. **Only text:** Only text is shown in the PowerPoint slide. **Only object:** Only object is shown in the PowerPoint slide.

Table 32: PowerPoint options

9.8.2.4 Filter Options

See 9.9 Filters.

Figure 53: Filter options

Filter options	
Setting	**Description**
Active filter	It is possible to set any of the defined filters active for one report element.

Table 33: Filter options

9.8.2.5 Answer Mask/Scale Mask

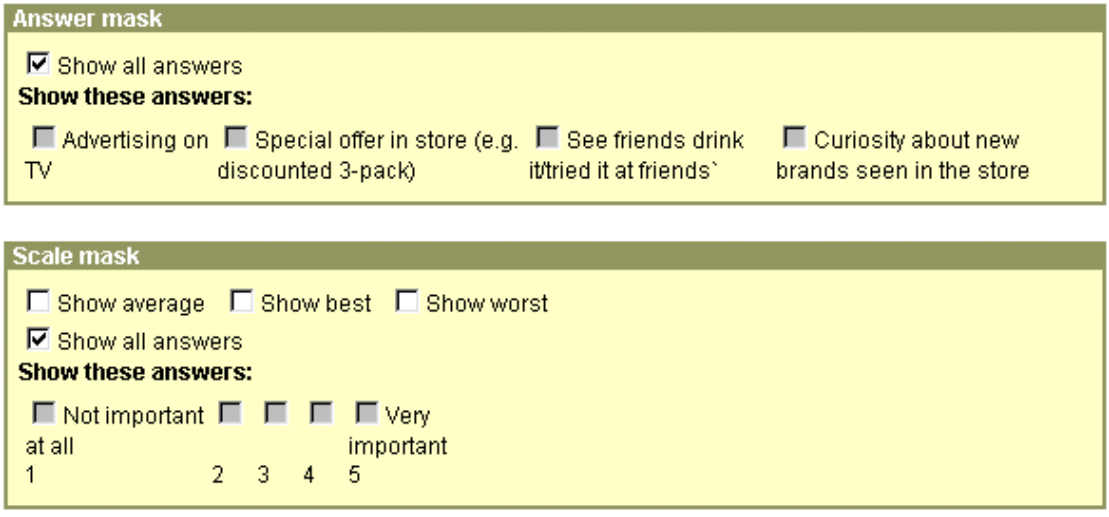


Figure 54: Answer Mask/Scale Mask

Answer mask/Scale mask	
Setting	**Description**
Show average	Choose this to show the average values of the variables. The answers/scales have to be assigned values to as described in 5.2.1.2.
Show best	The best average value is shown.
Show worst	The worst average value is shown.
Show all answers	All answers/scales are shown in the table/chart.
Show these answers	You may check off specific answers/scales you want to be shown in the table/chart.

Table 34: Answer mask/Scale mask options

9.8.2.6 Crosstab Options

Figure 55: Crosstab options

Crosstab options	
Setting	**Description**
Layout	**Stacked:** The questions in the *"With..."*-folder are stacked. **Nested:** The questions in the *"With..."*-folder are nested.
X totals	Set this to *" Yes"* to get the totals for a question to the right of the rest of the crosstab table.
Y totals	Set this to *" Yes"* to get the totals for a question under answer alternatives.
Table content is average of first node	The contents are the average values of the first question in the *Crosstab….* folder. The variable must be weighed to get average values.

Table 35: Crosstab options

9.8.2.7 Loop Options

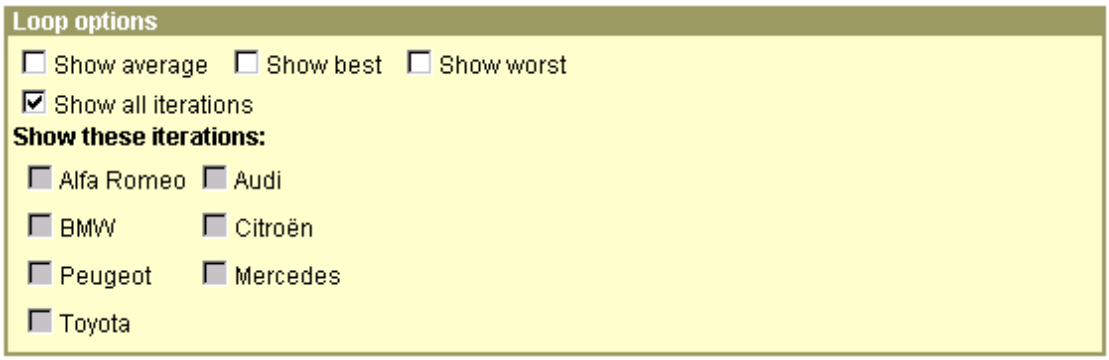


Figure 56: Loop options

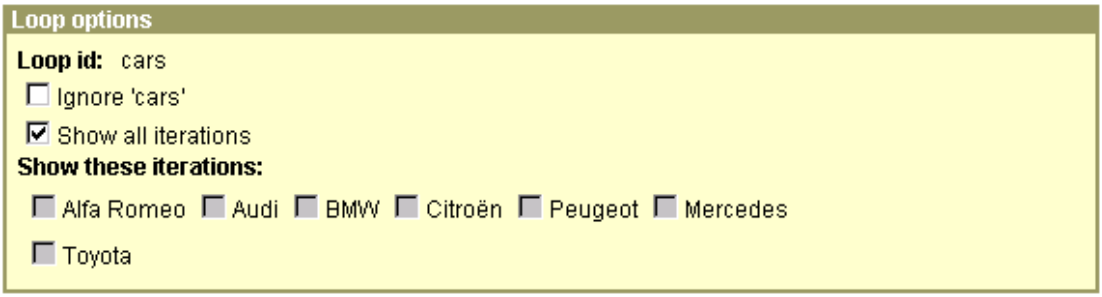


Figure 57:Loop Question options

Setting	Description
Show average	Choose this to show the average values of the variables. The answers/scales have to be assigned values to (weighed).
Show best	The best average value is shown.
Show worst	The worst average value is shown.
Show all iterations	Check this box if you wish all iterations to be shown. Otherwise, uncheck this box, and mark the iterations you wish to report upon instead.
Loop id	Denotes the loop you are editing.
Ignore ‘loop id’	This choice is available only for the questions in loops. If you check this box, the iterations will not be shown. Therefore, we would recommend unchecking it when reporting on the loop questions alone. On the other hand, you may check it when cross tabulating loops and loop questions, in order to prevent iterations from appearing both vertically and horizontally.

Table 36: Loop options

9.9 Filters

9.9.1 Defining Filters

Filters are defined like this:

1. Select the question to be used as filter within the *"Form"*-folder.
2. *Actions → Add Filter* from the menu above the reporting tree.
3. Enter the filter name (e.g. Male).
4. Select the precodes to let through the filter (e. g. the male precode if only male respondents are to be counted).

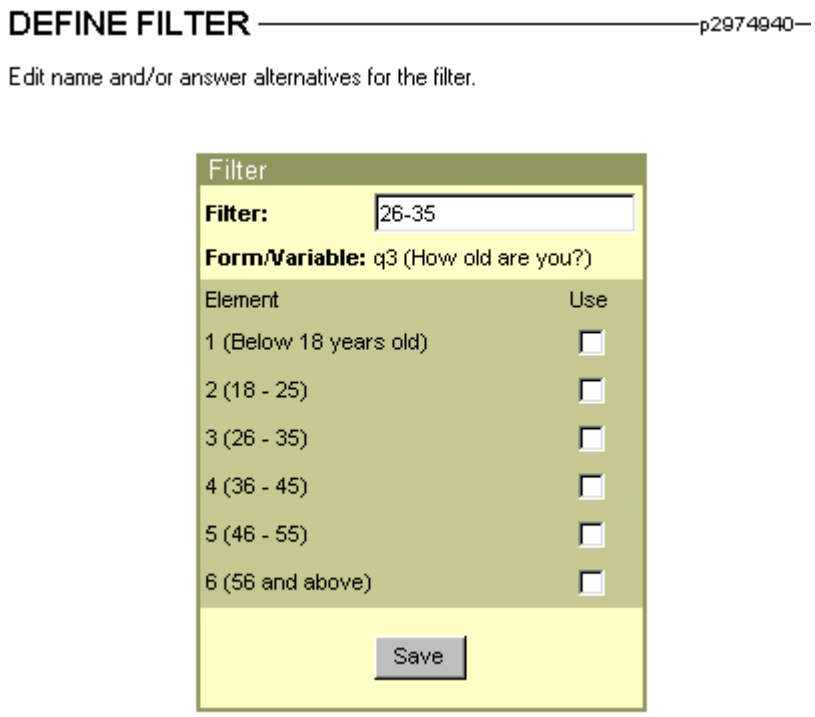


Figure 58: Filter definition

9.9.2 Applying Filters

Filters can be added globally or at the report element level.

- **Global filters:** Select *Online Reporting → Define filters* from the main **confirmit** menu. Then check the correct filter in the list.
- **Element level filters:** See Filter options in 9.8.2.

9.10 Report Publisher

9.10.1 Publishing a Report

The **confirmit** Publisher allows you to make your reports available to anyone you want. You select the data/graphs to display, and **confirmit** updates results according to your settings.

The reports may be published on the WEB (either openly available or password protected), or attached as an HTML file to an e-mail. If your company has installed **confirmit** internally, the location where reports are published (either your intranet or the WEB) has been decided by your company.

To be able to publish a report, you must create one as described in chapter 9 Reporting. The settings you choose for the report are also applied to the Publisher. However, note that the Publisher engine does not support 3D graphs. Also check that you have Excel as your *Chart Component* in *User Settings*, or you will get a different result published than what you see in the report.

To edit graphs to be published, open the relevant report form the *Reports* folder, and single-click on the slide you want to edit (as shown in the picture below). Choose *Properties* from the *Edit* menu, and do the adjustments you need. For more information about the options available to you, see User Manual chapter 9 Reporting.

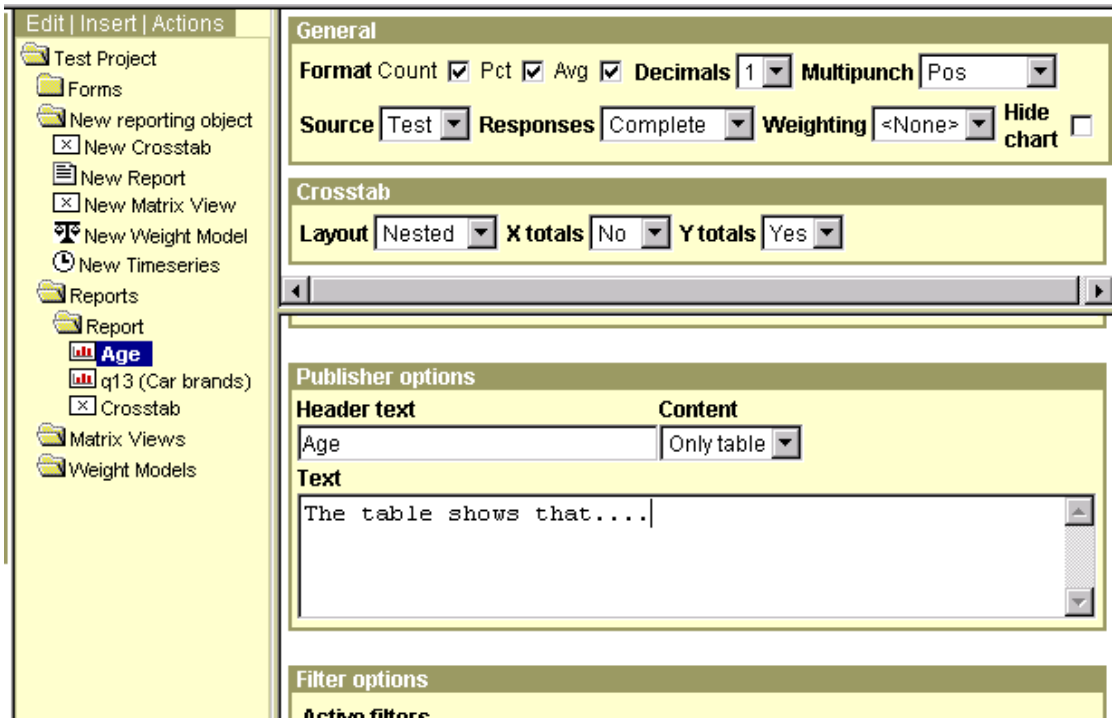


Figure 59: Editing a slide to be published

Settings that are unique for Publisher, are found in the *Publisher Options*-window (picture above). You can insert *Header text* and / or *Text*. They will both be placed above the tables/graphs being published. Your text fonts and colors can be defined by standard HTML commands. You can also choose whether to show tables, graphs or both.

After having prepared the report, you publish it by highlighting the report name (in the example above: R1), and choose *Publish Report* from the *Actions* menu. The following screenshot will appear:

Figure 60: The Publisher screenshot

The settings available to you are:

Setting	**Description**
Active	Tick this to make the report available to users
Run Now	By activating this, the report will be updated on the next engine run, even if the frequency setting requirement is not met (the FIRM engine normally runs every 5 minutes)
Report	The name of the report being published
Database	Can be either Test or Production
Lang	Language (based on your *Personal Settings* at the time of publishing)
Creator	Creator's name
Resp	Number of respondents of latest publishing
Failed	Error notification
Frequency	Allows you to choose how often the data is to be updated. See below
Targets	Settings for how the report is to be published
Log	Log of publishings
Users	Users allowed to view the report when publishing on a *protected link*
Template	Report layout (see 3.6 Templates).
One page	All reporting on one page
Last run	Time of latest update of data
Delete	Allows to delete a published report
Remove	The report is being removed from the web.

Table 37: Report publisher options

You must always define *Frequency* and *Targets* (and Users if you publish on a *protected link*) before a report is available for publishing.

About **Frequency**: **confirmit** will update the published report every x respondents (count), or at every given percentage increase of the respondent base (percent) or at a given time interval (minutes). The procedure is: Click Add, then choose a parameter, click save, enter the alternative you wish, and click save again. You can also combine different parameters.

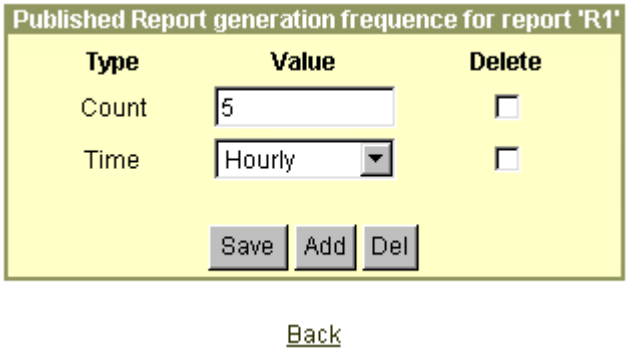


Figure 61: Frequency

About **Target**: Following the same add / save procedure as described above, you can choose three ways of publishing the report.

- The e-mail option sends the report as attachments to the mail-addresses you choose (each graph will be sent as separate gif).
- Cryptic link places the report on the address automatically generated. You can send the link to your users, or make it available from a home page. Anyone with knowledge of the link can view the report.
- Protected link places the report on the address automatically generated, but users need a *userid* and a *password* to access. You should have prepared a list of users as a tab delimited text document, with column headings *userid* and *password.* Click Users, Upload, Append or Overwrite, browse for the source file and complete by uploading. You must make login information available to the users.

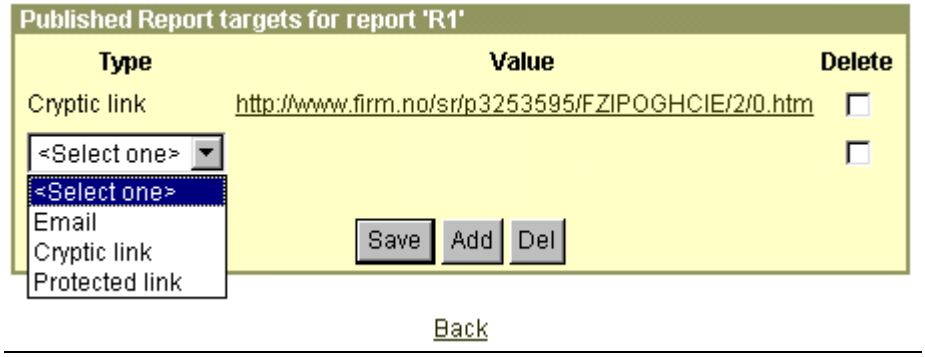


Figure 62: Targets

NB! Depending on the browser's settings of those entering the report, they might have to refresh (F5) each page of the report when entering a second time to get the updated data.

9.10.2 Changing a Report after Publishing

The report in the Report Folder is the source of the published report. So if you e.g. want to change chart type, or edit the Publisher Options, the changes will be applied the next time the Publisher engine runs (based on *Frequency settings* or *Run now* activation).

You cannot however change overall settings after publishing. So if you are publishing a report with Count Format (from *Format* in the *General*-window in figure 59), and later wish to change to Percent, you will have to publish the report once more after having changed the settings. The same applies for change in language, or if you want to remove or insert a graph after publishing. In all these cases you must publish once more. This, however, does not apply to change from Test data to Production data in *Source* in *General Settings.*

Outdated publishings are removed by clicking Delete and Save.

9.10.3 Publishing Different Data from One Survey to Separate Target Groups, or Same Report in Different Languages

You might want to make different results from a survey available to different target groups. You have to prepare the reports separately in *Online Reporting*. Thereafter you publish them one by one, selecting the wished publishing mode.

If you need the report to be in different languages, you just have to Duplicate the report in the Edit Menu. Enter information in the relevant language in the Publisher Options, change your User settings to the relevant language, and publish. Do these changes every time you publish a report in a new language.

9.11 Matrix Views

9.11.1 Introduction

A matrix view is mainly an end user interface, but can also be used by research analysts. The concept is to select three sets of questions.

Question set	**Description**
Questions	These questions are placed in the left part of the matrix. Only these questions can be reported as single questions within the matrix view.
Cross questions	These questions are placed in the bottom part of the matrix. The questions can be crosstabulated towards the cross questions.
Filters	The filter questions will be available in two filter dropdown lists at the top of the matrix. This means that results can be filtered on two questions simultaneously. In one filter you can chose one or more answer alternatives to let through the filter.

Table 38: Matrix view question sets

9.11.2 Configure Matrix Views

The matrix views are configured from the reporting tree. The following operations can be done on the matrix view:

- **Create a new matrix view.** A new matrix view is created by dragging a *"New Matrix View"*-object from the *"New reporting objects"*-folder into the *"Matrix Views"*-folder.
- **Change name.** Select *Edit → Properties* in the report tree menu, and change the matrix view name and id from the report frame.
- **Add question to a question set.** When a new matrix view is created, the matrix view element in the reporting tree includes three subfolders, one for each question set. Questions are added to the subfolders by dragging them from the *"Form"*-folder.

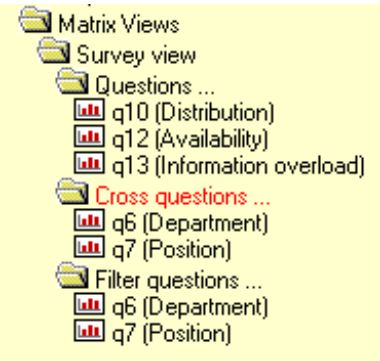


Figure 63: Matrix view configuration

9.11.3 Using Matrix Views

When the matrix view is initially entered, no charts or tables are displayed. The table below shows what options that are available in the matrix view.

NB! The chart or table must be refreshed every time the user changes options.

Option	Description
Select a question	By pressing a question part of the matrix, a question is selected. The question can be selected on and off by pressing them repeatedly. Dark color of the button indicates a selected question
Select a cross question	Cross-questions are operated the same way as the indicator questions. When a cross-question is selected, the selected indicators are crossed with this demography question.
Make filters available	Select the *"Filters"*-option
Apply a filter	The question to filter on is selected from a dropdown lists in the top of the matrix. When a question is selected, a row of answer alternatives appears right of the dropdown list. The alternatives to let through the filter should be selected.
Change display	The display can be switched between chart and table by changing the *"Display"*-option.
Change format	Format can be switch between count, percent and index (average) by changing the *"Format"*-option.
Change matrix size	The matrix size can be changed by changing the *"Screen"*-option. There are predefined matrix sizes defined to fit the following screen resolutions: 800x600, 1024x768 and 1280x1024
Make timeseries available	Select the *"Timeseries"*-option. This will add a time series row below the chart area.
Apply a timeseries	In the time series row, periods can be changed between weeks, months or years. When e. g. months are chosen, the months where there have been any respondents are displayed. When the trend button is selected, a line chart of the different weeks is displayed. When selecting a single week, this week is added as a filter.

Table 39: Matrix view operations

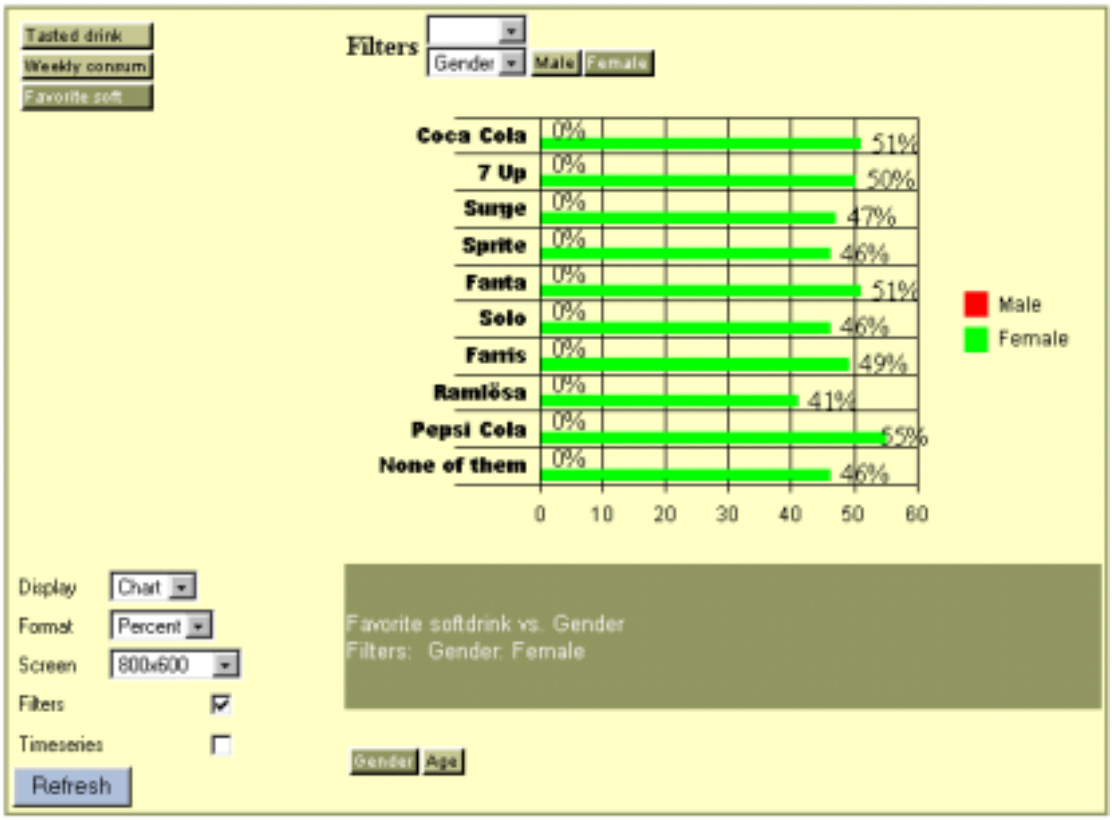


Figure 64: Matrix view

Here are some rules and guidelines for what can be done and what cannot be done within a matrix view.

- To be able to display a chart, at least one question has to be selected in the right question column.
- Index format can only be used with single punch and grid questions.
- Index format can only be used with weighted questions, see chapter 5.2.1.2 for how to weight a question.
- More than one question in the left question list can only be selected when dealing with indexes.
- A filter cannot be selected without selecting any of the answer alternatives.

Maximum 2 questions can be selected from the demography list. The crosstabs generated are nested by the demography questions. Nesting of more than two questions is rarely necessary, and would jeopardize the performance of the reporting server too much.

10 Recoding Data

Forms in the questionnaire can be turned into recoded variables by selecting [Properties => Recoded var] for the question. This causes an expression column to appear in the answer list of the question. These expressions should be programmed as a Boolean expression evaluating to true or false.

There are two kinds of recoded variables in **confirmit**, SQL based and JScript based variables. Calculation depends on what kind of variable is used. SQL based is the default variable type. Select [Properties => JScript based] to change the variable type.

The expressions are entered in the answer list of the recoded variable. To create a variable based on another specific question, it may be helpful to use the expression builder. This is activated by clicking the question you will base your variable upon when you are in the answer list of the variable.

10.1 Expression Syntax and Evaluation

Common for SQL and JScript based variable expressions is the way values of other questions are specified:

- **Single punch and open-ended questions:** 'Question id'
- **Multi punch and grid questions:** 'Question id' + '_' + 'Answer precode'

The values returned by such a reference is always a character. This is important to remember for rank order comparison. For strings, the value '11' is less that '2'.

10.1.1 SQL

For SQL based variables, any legal SQLServer primitives can be used within the expressions. Hence it is recommended to know some SQL to do create advanced expressions. To build advanced expressions correctly, it is necessary to know how the UPDATE statements of the expressions are created in CONFIRMIT. This is best illustrated by two examples:

Example 1:

A recoded variable 'v1' of type single. 'age' is an open ended question containing the age of the respondents. convert() is a SQL method for type convertion.

English	*Expression*	*Precode*
Young	*convert(int ,age)<40*	*Y*
Old	*convert(int ,age)>=40*	*O*

Generated UPDATE statement:

```
UPDATE [response_var0]
SET [v1]= CASE      WHEN convert(int,age)<40 THEN 'Y'
                    WHEN convert(int,age)>=40 THEN 'O'
                    ELSE NULL
          END
FROM      response_control INNER JOIN
          response_var0 INNER JOIN
          response0 ON
          response_var0.responseid = response0.responseid ON
          response_control.responseid = response_var0.responseid
```

Example 2:

A recoded variable 'v2' of type multi. 'car' is a multi format question

English	*Expression*	*Precode*
American car	*car_chevy<>'0' OR car.buick <> '0'*	*A*
European car	*car_opel<>'0' OR car_vovlo<>'0' OR car_saab<>'0'*	*E*

Generated SQL statements:

```
UPDATE [response_var0]
SET [v2_1]=CASE     WHEN car_chevy<>'0' THEN '1'
                    WHEN NOT (car_chevy<>'0') THEN '0'
                    ELSE NULL
            END
FROM       response_control INNER JOIN
           response_var0 INNER JOIN
           response0 ON
           response_var0.responseid = response0.responseid ON
           response_control.responseid = response_var0.responseid


UPDATE [response_var0]
SET [v2_2]=CASE     WHEN car_opel<>'0' OR car_volvo<>'0' THEN '1'
                    WHEN NOT (car_opel<>'0' OR car_volvo<>'0') THEN '0'
                    ELSE NULL
            END
FROM       response_control INNER JOIN
           response_var0 INNER JOIN response0 ON
           response_var0.responseid = response0.responseid ON
           response_control.responseid = response_var0.responseid
WHERE response_var0.responseid IS NOT NULL
```

In adition to normal SQL syntax the following convertsions are done by the CONFIRMIT recoding engine:

From	To
==	=
&&	AND
||	OR
!=	<>
qn.in('a', 'b')	qn IN ('a', 'b')
parseInt(q1)	CASE WHEN ISNUMERIC(REPLACE(CONVERT(varchar, q1),',',' '))=1 THEN CONVERT(int, REPLACE(REPLACE(CONVERT(varchar, q1),CHAR(10),''),CHAR(13),'')) ELSE NULL END

10.1.2 JScript

JScript based variables are recoded by looping through all the variables in a JScript environment. Hence all primitives of the JScript language are available from the expression.

When referencing other questions from the expressions, the calculation engine will create a JScript object based on the value in the database for the current respondent. This object has the following methods available:

Method	Return value
getValue() toString() valueOf()	The ADO value of the ADO field method value(). This is a string except for interview_start and interview_end that return date.
int()	ParseInt (getValue ())
str()	New String ((getValue ())
date()	New Date (getValue ())
float()	ParseFloat (getValue ())
int()	ParseInt (getValue ())

in(param1, param2,…)	True if getValue() equals at least one of the parameters.
weight()	The weight associated to the value of the field. This is the weight used to report averages.

In the evaluation context, the following functions are also available:

Function	Description
average(p1, p2, …,pN)	The averages of the parameters: (p1 + p2 + … + pN)/N All the parameters are converted by parseFloat() before added. Parameters equaling an empty string are not included in the average calculations at all. The parameter is also ignored in the parameter count N.

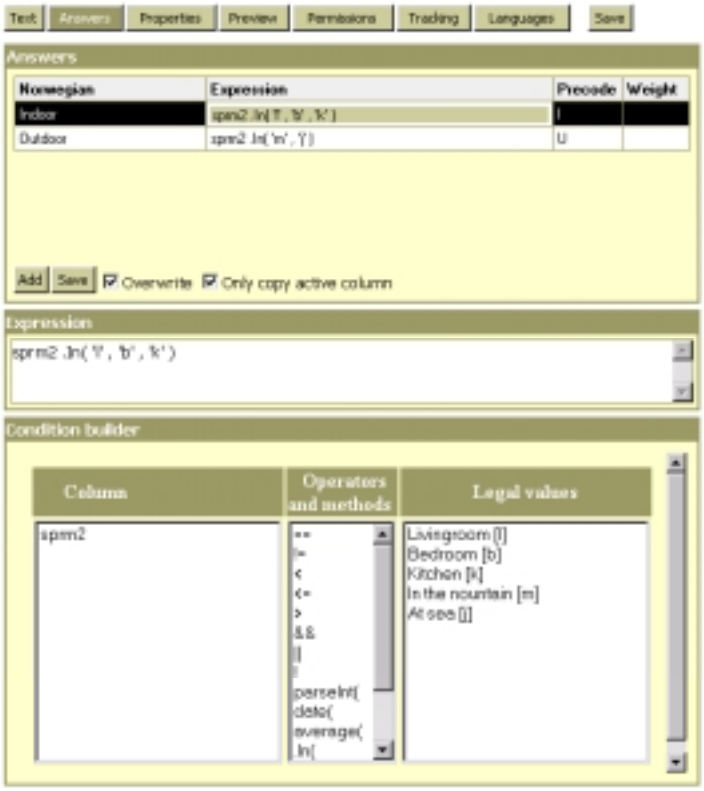


Figure 65: Variable definition with condition builder

11 Exports of Data Files

It is possible to export response data to SPSS, MS Excel and ASCII files. The export files will have the question IDs as labels (see 5.2.1.4 for an explanation of labels for different forms.)

For exports without loop variables the export files will have one row per respondent. If the export includes a loop variable, the export files will include one file for each loop. There will be one row for each loop iteration, so there will be several rows for each respondent.

All exports will be sent as compressed files with .zip extension ("zipped") and may be extracted by using standard software like WinZip®.

11.1 Template Editor

You may build templates to choose a subset of data fields for your exports. Instead of choosing all variables when preparing an export, you may choose one of these templates to get the subset you want.

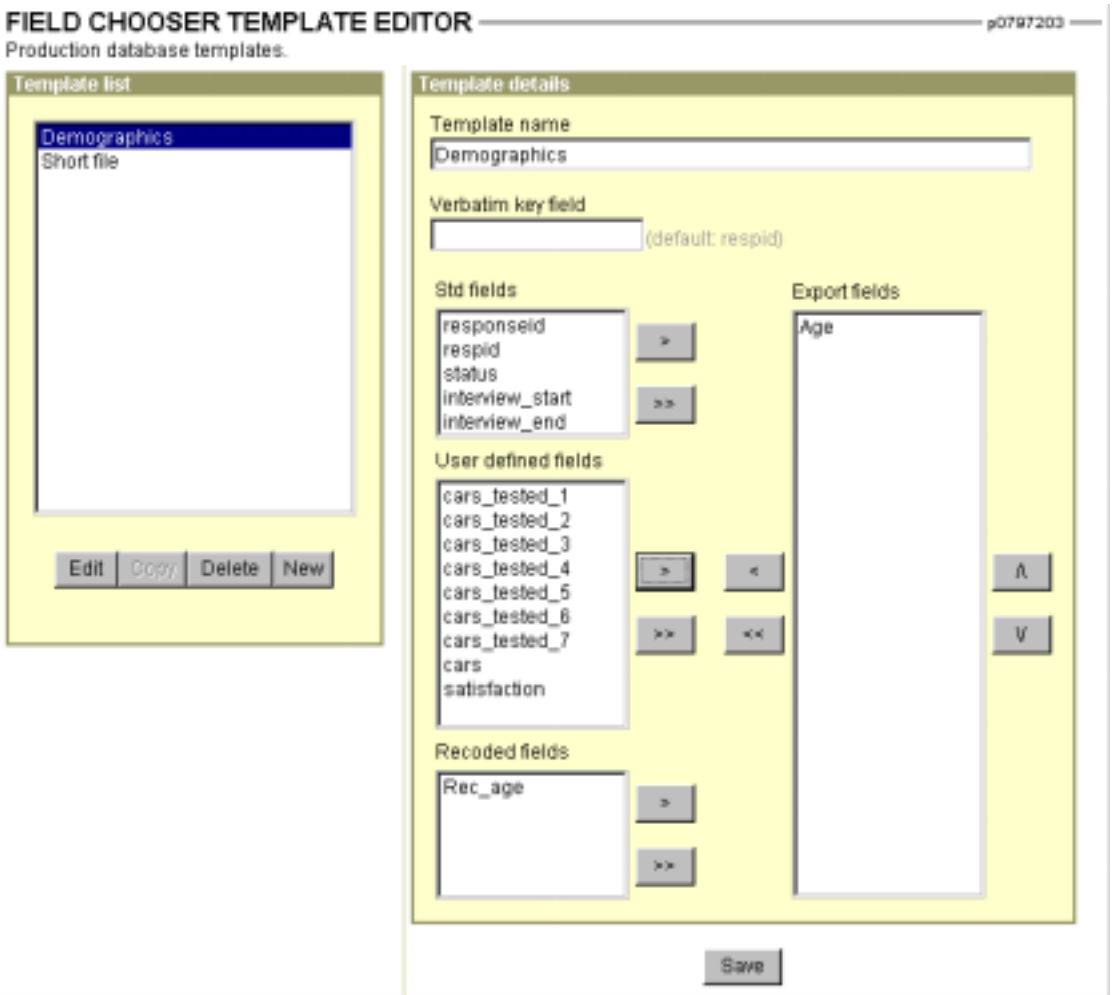


Figure 66: Field chooser template editor

From the template list you may choose to *Edit*, *Copy* or *Delete* an existing template (select template from the list and click the desired action), or make a new template by clicking *New.*

Field	Description

<table>
<tr><td>Template name</td><td>The name of the template.</td></tr>
<tr><td>Verbatim key field</td><td>Open Text questions that have no field width specified (see 5.2.1.4) will not be included in the export files when you export to fixed width ASCII. (Open Text questions initially have no field width limitations, so if it is not specifically applied to them, they could be infinite.) So when exporting to fixed width ASCII, a separate export has to be done for the Open Text questions (see ...). These files will by default have respid as identification of the respondents, but if the data files include another id you would rather use (e.g. a member id) this may be set in the “Verbatim key field”.</td></tr>
<tr><td>Std fields</td><td>These are standard system-generated fields.
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>responseid</td><td>Respondent identification given at the time the respondent starts answering the survey.</td></tr>
<tr><td>respid</td><td>Respondent identification given at the time the respondent is added to the respondent list.</td></tr>
<tr><td>status</td><td>Status of the interview. Complete if the respondent has reached the end of the questionnaire (or a STOP node where status is set to complete), screened if the respondent has reached a stop node where status is set to screened, quota_full if the respondent has reached a stop node where the status is set to quota full, and empty if the respondent has not finished the questionnaire.</td></tr>
<tr><td>interview_start</td><td>The time when the respondent entered the questionnaire. If the respondent is allowed to re-enter the questionnaire, it will be the last time the respondent entered the questionnaire.</td></tr>
<tr><td>interview_end</td><td>The time when the respondent finished the questionnaire, either from a STOP node or by reaching the end of the questionnaire.</td></tr>
</table></td></tr>
<tr><td>User defined fields</td><td>All the variables from the questionnaire, including hidden questions, but excluding recoded variables.</td></tr>
<tr><td>Recoded fields</td><td>All recoded variables from the questionnaire (see 10 Recoding Data).</td></tr>
<tr><td>Export fields</td><td>These are the fields that will be included in the exports done with this template.</td></tr>
</table>

Table 40: Template details

You move variables from *std fields*, *user defined fields* and *recoded fields* to *export* fields by selecting a variable and clicking > (or < to move it back) or by clicking >> to move all from the field. You may change the order of the variables by adjusting the list with ∧ and ∨. Click on *Save* when you are finished making changes to the template.

11.2 Data Export

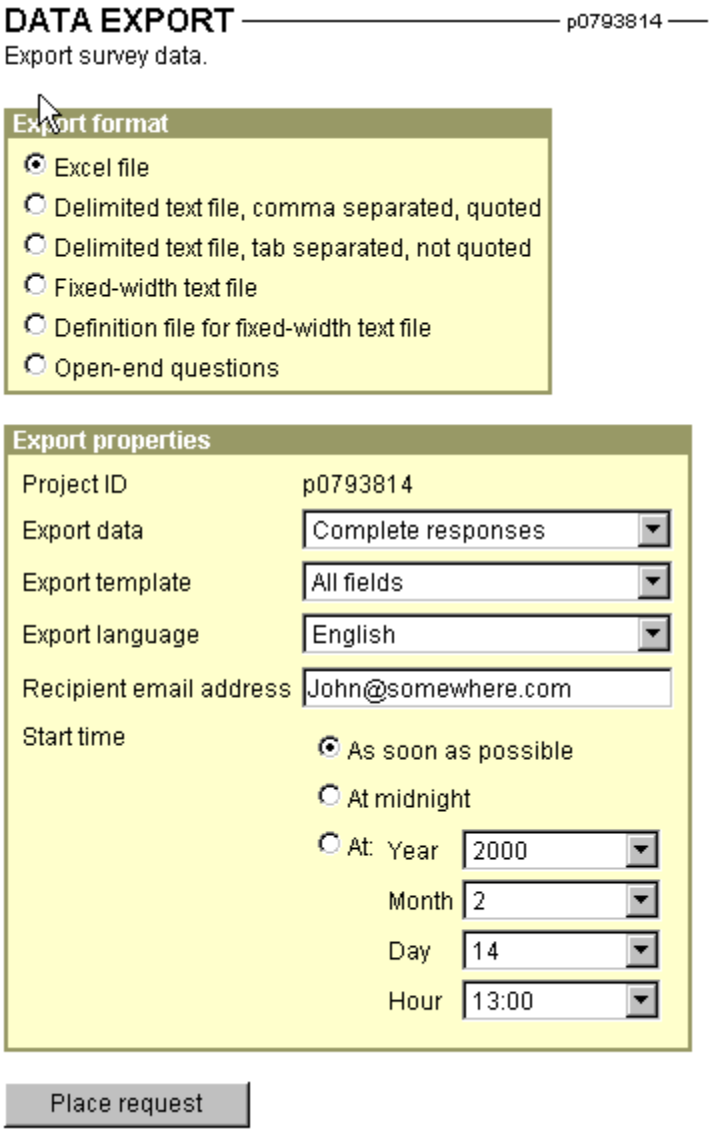


Figure 67: Data export

Export Format	**Description**
Excel file	MS Excel file in the Office version specified in User Settings (see 3.2). The variables may be split over several worksheets. Excel exports will never be made in Excel 4.0 and older, since these old versions do not support more than one worksheet for each workbook.
Delimited text file, comma separated, quoted	Comma (,) separated ASCII text file. All answers will be inside quotes ("").
Delimited text file, tab separated, not quoted	Tab separated ASCII text file. No quotes (")
Fixed-width text file	Fixed-width ASCII text file. Will give an export file of single, multi and grid questions. If Open Text questions have field width specified (see 5.2.1.4), they will be included as well. Exporting Open Text questions where the field width has not been specified can be done by selecting "Open Text Questions", The

<table>
<tr><td></td><td>field widths will be equal to the field widths from properties of the questions if provided or else to the default of the database. The fields with numeric responses will be zero-filled, so e.g. a precode 1 with a field width of 3 will be reported as "001".</td></tr>
<tr><td>Definition file for fixed-width text file</td><td>Will generate a MS Excel file with a description of the exported fixed-width text file. (The information provided may be useful for other exports as well.)<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Form ID</td><td>The Question (form) ID</td></tr>
<tr><td>Var ID</td><td>The variable name. Form ID and Var ID will be equal for Single an Open Text questions, but will differ for Grids and Multi questions. See 5.2.1.4 for an explanation of the variable names of multis and grids.</td></tr>
<tr><td>Start Col</td><td>The start position of this variable in the export file, counting characters from left to right.</td></tr>
<tr><td>End Col</td><td>The end position of this variable in the export file, counting characters from left to right.</td></tr>
<tr><td>Punch</td><td>Will list the possible punches (precodes) possible for these answers corresponding to the texts in the Answer Text field.</td></tr>
<tr><td>Question Text</td><td>The text of the question.</td></tr>
<tr><td>Answer Text</td><td>The text of the answer. (Corresponding to Punch).</td></tr>
<tr><td>Data Type</td><td>Question type (or system generated type).</td></tr>
<tr><td>Level ID</td><td>Loop level. Gives the id of the loop this question is inside.</td></tr>
</table></td></tr>
<tr><td>Open-end questions</td><td>Open-end questions are all open questions (Open Text questions, Other, specify from answer lists or Multi questions with Open Text property selected) that do not have a field width specified (see 5.2.1.4). They will be included in the normal export of the tab- and comma-separated text files and the Excel files, but not in the fixed width text files. If you choose "Open-end questions" you will receive a "fixed-width" ASCII text file with three columns: Respondent ID, Question ID and that respondents answer to that Open-end question. The respondent ID will by default be respid, but you may choose a different ID by selecting a different "Verbatim key field" in your template (see 11.1)</td></tr>
</table>

Table 41: Export formats

Export property	Description
Project ID	The system generated ID of the project
Export data	Choose if you want your export to include all responses, or just the ones with status "complete".
Export template	All fields or a user-defined template for a subset of the fields (see 11.1)
Export language	Select which language Question titles and labels should appear in. Choose only from active languages.
Recipient email address	The export file is sent to this mail address.
Start time	Specifies when the report shall be put in the task list. Either immediately, by midnight or at a specific time.

Table 42: Data export properties

Click *Place Request* to send the request to Task Management (see 3.4).

11.3 SPSS Export

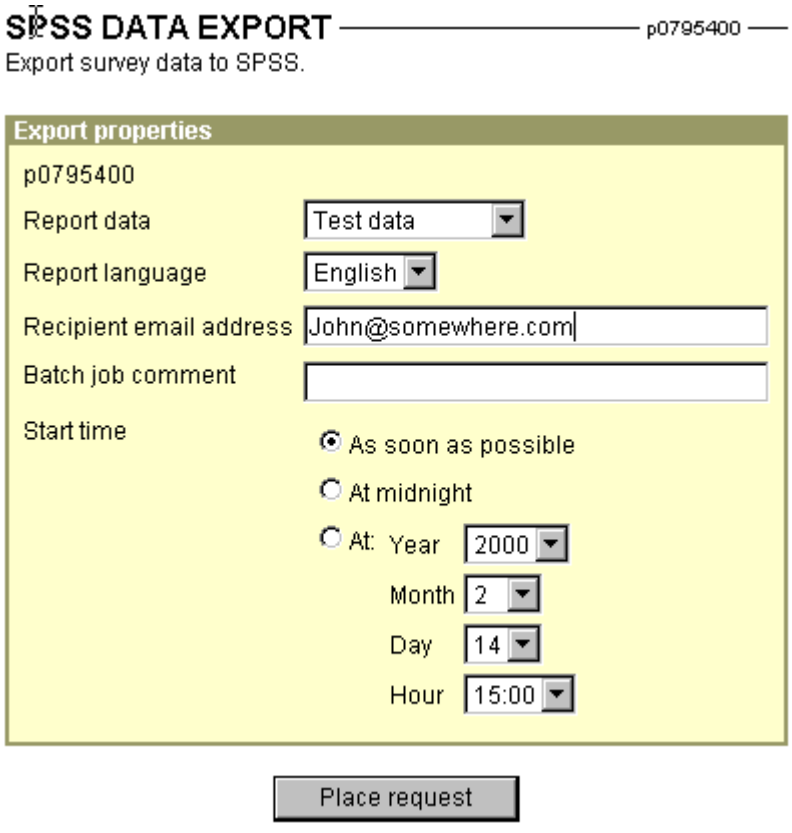


Figure 68: SPSS Data Export

SPSS Export will generate two files: A SPSS file (with the extension .sav) and an ASCII text file with a description of the Variable-ID mappings. SPSS variable IDs are restricted to a length of 8 characters. Question IDs that are longer than 8 characters will be replaced with IDs of the format "@n" where n is a number. The text file has a list of all the variables, their name in CONFIRMIT and their corresponding names in the SPSS file.

SPSS Export property	Description
Report data	The data set you want an export from. Either "Production data" or "Test data"
Report language	The SPSS file will include the question text and labels from the questionnaire in the language selected here. Choose only from active languages.
Recipient email address	The export is sent to this mail address.
Batch job comment	This comment will appear in the comment field of the Task Management lists. Use it to distinguish between reports.
Start time	Specifies when the report shall be put in the task list. Either immediately, by midnight or at a specific time.

Table 43: SPSS Export properties

Click *Place Request* to send the request to Task Management (see 3.4).

11.4 MS Word Export

You can also export your questionnaire to MS Word format. This is currently beta functionality, but it has been added because of the large number of requests. **This will export the last compiled version of the questionnaire.** It does not include any kind of skip logic.

Note that you have to have compiled a database as well as chosen the same language user settings (see 3.2) as in *Report language* in order to be able to export your questionnaire into MS Word format. Comment: This is unsupported functionality, meaning that we do not currently bug fix this feature.

Figure 69: MS Word Export

Click *Place Request* to send the request to Task Management (see 3.4).

12 APPENDIX A: confirmit Script Reference

12.1 Accessing Survey Variables

f function
Description Accesses survey variable values. **Declaration** `f(`*qID,* `[`*expr1*`[, `*expr2*`,...]])`

You may think of `f` as a mnemonic for *form*, which is place in the questionnaire definition where all variables are declared.

qID is the ID of the form, as specified in the form's properties, where the variable(s) in question are declared.

[expr1[, expr2,...]]: none or more expressions used to identify a specific iteration if the question is defined inside a loop. Within loops, tables of variables are associated with each question instead of the single item normally used. Each expression should evaluate to a valid loop iterator value, starting with the value of the innermost nested loop. If the number of expressions provided is less than the current loop nesting level, then the current values will be used for the loop iterators of the enclosing blocks.

Say that you have a question – q1 – placed inside a loop over the set {"1", "2", "3", ...}.

During execution of loops, the default values of [expr1[, expr2,...]] are set to the current values of each loop's iterator variable. In our case, this means that the default value for expr1 will be "1", "2", etc. If you want to refer to q1's value within the current iteration you simply use:

```
f("q1")
```

Call

```
f("q1", "1")
f("q1", "2")
```

etc..

if you need to refer to answers given in a specific iteration. Explicit qualification is only necessary if you need to refer to an answer given in a previous iteration or when execution flows past the loop where the question is defined.

The function returns different types of objects depending on the form type:

single: coded variable with a value domain equal to the "precodes" in the answer list of the form.

open: open variable

multi: set of dichotomy variables with member names corresponding to the "precodes" in the answer list of the form.

multi with OpenText checked**:** set of open variables with member names corresponding to the "precodes" in the answer list of the form.

grid: set of coded variables, each with a value domain equal to the "precodes" in the form's scale list, and with member names corresponding to the "precodes" in its answer list.

12.1.1 Variable Types

confirmit currently provides support for three types of variables:

- *Coded variables* are assigned distinct values from a pre-coded domain of alternatives. In **confirmit** users enter the responses to coded variables by use of groups of radio buttons or dropdown lists. **Single** forms declare one and **grid** forms a set of coded variables.
- *Dichotomies* may be assigned the integer values 1 or 0, which denote a positive or a negative response respectively. **confirmit** uses checkboxes for input of dichotomies. These variables are declared in **multi** forms.
- *Open variables* have an unbounded value domain. In expressions, they are treated as strings. You declare open variables in **open** forms or by checking the OpenText and/or Ordered property for a **multi** form.

Compounds group one or more variables that are defined in the same form. In **confirmit** a compound is defined for each multi or grid form.

12.1.2 Basic Variable Objects

In the interview system, JScript objects represent each variable. The basic interface shared by all types of variable objects is listed below. In the following table `x` denotes the variable object and `v` a JScript expression:

Expression	Result type	Description
`x.get()`	String	Returns the value of the associated field.
`x.set(v)`	String	Sets x equal to v.
`x.label()`	String	Returns the variable's label in the current language.
`x.toString()`	String	Converts the variable to a string.
`x.toNumber()`	Number	Converts the variable to a number using radix 10 and returns the result.
`x.toBoolean()`	Boolean	Converts the variable to a Boolean.

Table: Variable objects expressions

Type information is available at run-time by examining an object property. For each variable type a different property is defined:

Property	Description
x.CODED	Object represents a coded variable.
x.OPEN	Object represents an open variable.
x.DICHOTOMY	Object represents a dichotomy.
x.COMPOUND	Object represents a compound.

Example:

```
function foo(qID)
{
    var v = f(qID);
    if (v.CODED)
        return "" + v.value():
    else if (v.COMPOUND)
        return "" + v.categories();
}
```

12.1.2.1 Coded Variables

Expression	Result type	Description
x.value()	String	Returns the value of the associated field.
x.valueLabel()	String	Returns the label associated with the variable in the current language.
x.domainValues()	Array	Returns an array of all the valid values for x subject to masking.
x.domainLabels()	Array	Returns an array of all labels corresponding to the valid values of x, subject to masking and in the current language.

The objects used to represent coded variables also implement the Set interface described in Section 12.1.3.

.toBoolean() returns true if the question has been answered, false otherwise.

12.1.2.2 Open Variables

.toBoolean() returns true if the question has been answered and is different from an empty string, false otherwise.

12.1.2.3 Dichotomies

.toBoolean() returns false for no answer and the value “0”.

12.1.2.4 Compounds

Returned for grid/multis. Members are the elements in the answer list.

Member access:

`x[ident]`	Variable object	Returns the variable object representing the member identified by ident.

Implements:

Expression	Result type	Description
`x.values()`	Array	Returns an array of the values of each member in the compound.
`x.categories()`	Array	Returns an array of all precodes associated with the members whose toBoolean member returns true.
`x.categoryLabels()`	Array	Returns an array of labels associated with each category in the current language.
`x.domainValues()`	Array	Returns an array of all the valid categories for x subject to masking.
`x.domainLabels()`	Array	Returns an array of all labels corresponding to the valid categories of x, subject to masking and in the current language.

12.1.3 The Set Interface

Set arithmetic is primarily used in expressions that mask alternatives of a coded variable, prevent answers to certain variables in a multi-response form or to select a set of iterations for a loop.

The interface that must be implemented by objects used in set expressions are summarized in the following table, where `l` is an object that implements the set interface. `r` is a JScript expression:

Expression	Result type	Description
`l.members()`	Array	Returns a JScript array of all `l`'s members. Typically used for iteration over the set's members.
`l.union(r)`	Set	Returns a new set that is the union of l and r.
`l.isect(r)`	Set	Returns a new set that is the intersection of l and r.
`l.diff(r)`	Set	Returns a new set that is the difference between l and r.
`l.inc(r)`	Boolean	Test for set inclusion. Returns `true` if `r` is a member of `l`, `false` otherwise.
`l.size()`	Number	Returns the cardinality of l.

Please note that for each of the functions `union`, `isect`, and `diff`, `r` must evaluate to an object that implements a set interface, otherwise these functions will fail.

12.1.3.1 Basic Set Type

Set type implements the set interface and members for adding/removing members.

Expression	Result type	Description
`l.add(r)`	Set	Adds r to `l` and returns the new value of the set.
`l.remove(r)`	Set	Removes r from `l` and returns the new value of the set.

The instances of set that you will use are most often those returned by functions defined by the runtime. Sometimes, however, you will need to create set instances yourself.

The `set` function is provided to simplify this:

`set` function

Description

Creates a new set object.

Declaration

`set([`*mem*$_1$`], …`*mem*$_n$`])`

Remarks

`set` returns a set instance. If provided, the set is populated with the arguments *mem*$_1$ to *mem*$_n$.

`nset` function

Description

Creates a new set object {1, 2, 3, …, *n*}

Declaration

`nset(`*n*`)`

Remarks

`nset` returns a set instance. The set is populated with the members 1, 2, 3, …, n.

`nnset` function

Description

Creates a new set object over an integer range: {*m*, *m*+1, …, *n*}

Declaration

`nset(`*m*`, `*n*`)`

Remarks

`nnset` returns a set instance. The set is populated with the members *m*, *m*+1, …, *n*.

`a` function

Description

Returns the complete set of "precodes" defined in a form's answer list.

Declaration

`a(`*formid*`)`

Remarks

`a` returns the complete set of "precodes" defined in the answer list belonging to the form identified by *formid*.

The semantics vary depending on the form type:

single: the returned set corresponds to the value domain of the associated variable.

multi and **grid:** the returned set corresponds to the individual variables' subscripts in the form.

12.2 Form Validation

12.2.1 Adding Your Own Validation Code

One of the form properties is a text box where you may enter arbitrary script code to validate answers. Depending on the complexity of the validation problem, crafting this code may require a more intimate knowledge of JScript than can be presented in this document. This presentation of validation code is limited to a description of the interface provided by the runtime for signaling errors and setting error messages.

12.2.1.1 Runtime Functions for Error Handling

The *CONFIRMIT* runtime provides a number of functions for use in validation scripts. The functions provide information about the types of errors trapped by the system's default validation and give you the capability to control the error messages displayed for the page and individual questions.

`RaiseError` function

Description

This function is invoked to signal an error.

Declaration

```
RaiseError()
```

Remarks

`RaiseError` signals to the interview runtime that the input entered in a form is invalid and that the form should be redisplayed with an error message. The first call to `RaiseError` on a page also sets the system default error message for the page.

`ClearError` function

Description

This function clears the page's top error message.

Declaration

```
ClearError()
```

Remarks

`SetErrorMessage` function

Description

This function sets the page's error message for a specified language.

Declaration

`SetErrorMessage(`*langID*`, `*message*`)`

Remarks

Call this function to set the error message at the top of the page (to override the system default, "Please answer all questions properly. Please review all questions on this page"). The function takes two parameters:

- *langID*: A language ID. This argument is provided to support multi-lingual studies. You should provide one message for each of the languages used for interviewing. The runtime provides an object, `LangIDs` , whose members map language codes to the internal values used to identify each language available in **confirmit**. The member names equal the codes specified in the international standard ISO 639 "Code for the representation of names of languages". For example, `LangIDs.en` represents English; `LangIDs.fr`, French; `LangIDs.de`, German; and so on.
- *message*: This argument is simply a string value with whatever message you wish to display.

`AppendErrorMessage` function

Description

This function appends a new line to the page's error message for a specified language.

Declaration

`AppendErrorMessage(`*langID*`, `*message*`)`

Remarks

Call this function to append a string to the current page's error message. The function takes two parameters:

- *langID*: A language ID. See `SetErrorMessage` for details.
- *message*: This argument is a string value with whatever message you wish to append to the current message.

`ClearQuestionError` function

Description

This function clears a question's error message.

Declaration

```
ClearQuestionError()
```

Remarks

`SetQuestionErrorMessage` function

Description

This function sets a question's error message for a specified language.

Declaration

`SetQuestionErrorMessage(`*langID*`, `*message*`)`

Remarks

Call this function to set the error message for the current question. This message will override any previous message added by the system or your custom validation code. The function takes two parameters:

- *langID*: A language ID. See `SetErrorMessage` for details.
- *message*: This argument is simply a string value with whatever message you wish to display.

`AppendQuestionErrorMessage` function

Description

This function appends a new line to a question's error message for a specified language.

Declaration

`AppendQuestionErrorMessage(`*langID*`, `*message*`)`

Remarks

Call this function to append a string to the current question's error message. The function takes two parameters:

- *langID*: A language ID. See `SetErrorMessage` for details.
- *message*: This argument is simply a string value with whatever message you wish to display.

`QuestionErrors` function

Description

This function returns true if an error has been raised during validation, false otherwise.

Declaration

```
QuestionErrors()
```

Remarks

Custom validation code should call this function when control is passed to it if is concerned with whether the system-provided checks raised an error or not.

`MissingRequiredError` function

Description

This function returns true if one or more required answers to a question are missing, false otherwise.

Declaration

```
MissingRequiredError()
```

Remarks

`ExclusivityError` function

Description

This function returns true if an exclusive answer (“single punch”) has been selected together with any other alternative.

Declaration

```
ExclusivityError()
```

Remarks

`SizeError` function

Description

This function returns true if one or more open-ended answers exceeded the maximum length imposed by the field width, false otherwise.

Declaration

```
SizeError()
```

Remarks

`NotSpecifiedError` function

Description

This function returns true if an alternative in an Other-Specify construct has been selected/answered without filling in the associated “Specify” value. Otherwise it returns false.

Declaration

```
NotSpecifiedError()
```

Remarks

`NotSelectedError` function

Description

This function returns true if a “Specify” value has been entered in an Other-Specify construct without selecting/answering the associated alternative. Otherwise it returns false.

Declaration

```
NotSelectedError()
```

Remarks

`RankError` function

Description

This function returns true if the “Ordered” has been selected for the form and the answers to the questions are non-numeric, do not start at 1, or are non-consecutive. Otherwise it returns false.

Declaration

```
RankError()
```

Remarks

`NumericError` function

Description

This function returns true if the “Numeric” property has been selected for the form and the answers to the questions are non-numeric. Otherwise it returns false.

Declaration

```
NumericError()
```

Remarks

`PrecisionError` function

Description

This function returns true if the "Numeric" property has been selected for the form and the answers to the questions cannot be stored within the specified precision. Otherwise it returns false.

Declaration

```
PrecisionError()
```

Remarks

`ScaleError` function

Description

This function returns true if the "Numeric" property has been selected for the form and the answers to the questions contain more decimals than the specified scale. Otherwise it returns false.

Declaration

```
ScaleError()
```

Remarks

`RangeError` function

Description

This function returns true if the "Numeric" property has been selected for the form and the answers to the questions are outside the specified range. Otherwise it returns false.

Declaration

```
RangeError()
```

Remarks

12.2.1.2 Template based error messages

The system allows you to use templates for your error messages in custom validation. For each error type, various elements are filled in that can improve the information content of the messages you report to the end users. Instead of simply using fixed strings for error messages, the `ErrorTemplate` function can be used to "plug in" information about the current question.

`ErrorTemplate` function
Description This function returns a string formatted according to the provided template specification. **Declaration** `ErrorTemplate(`*spec*`)` **Remarks** The function takes one parameter: • *spec*: A string with the template specification. The remainder of this section describes elements that can be used to specify templates. It also provides a few examples of their use.

Missing required

The following template elements are defined when `MissingRequiredError` returns true:

MISSING	Labels of all questions that lack answers, comma separated.
MISSING_AND	Same as above, but with the word “and” separating the two last items.
MISSING_OR	Same as above, but with the word “or” separating the two last items.

Example of use:

```
Please select an answer for ^MISSING_AND^.
```

Exclusivity tests

The following template elements are defined when `ExclusivityError` returns true:

SEL_EXCL	The labels of all exclusive (single punch) answers that were selected.
SEL_EXCL_AND	Same as above, but with the word “and” separating the two last items.
SEL_EXCL_OR	Same as above, but with the word “or” separating the two last items.

SEL_NEXCL	The labels of all non-exclusive (multi punch) answers that were selected.
SEL_NEXCL_AND	Same as above, but with the word “and” separating the two last items.
SEL_NEXCL_OR	Same as above, but with the word “or” separating the two last items.

DEF_EXCL	The labels of the defined set, i.e. the unmasked options, of exclusive (single punch) members in the inspected set.
DEF_EXCL_AND	Same as above, but with the word “and” separating the two last items.
DEF_EXCL_OR	Same as above, but with the word “or” separating the two last items.

DEF_NEXCL	The labels of the defined set of non-exclusive (multi punch) members in the inspected set.
DEF_NEXCL_AND	Same as above, but with the word “and” separating the two last items.

DEF_NEXCL_OR	Same as above, but with the word “or” separating the two last items.

Example of use:

```
Please do not check ^SEL_EXCL_OR^ if you check other answers to the question.
```

Other-Specify verification

The following template elements are defined when `NotSpecifiedError` or `NotSelectedError` returns true:

OTHER	Label of the alternative/input that requires specification in an Other-Specify construct.
SPEC	The specification

Examples of use:

```
If you specify ^OTHER^ then please select the ^OTHER^ option.
```

```
Please specify if ^OTHER^ is chosen.
```

Answer size tests

The following template elements are defined when `SizeError` returns true:

TOO_LONG	The labels of the inputs that were too long.
TOO_LONG_AND	Same as above, but with the word “and” separating the two last items.
MAX_SIZE	The maximum allowed size.

Example of use:

```
The answers to ^TOO_LONG^ were longer than ^MAX_SIZE^ characters and have been truncated. Please review.
```

Rank order tests

The following template elements are defined when `RankError` returns true:

RANK_MIN	The label of the first member of the ranking scale, or 1 if the question is open ended.
RANK_MAX	The label of the nth member of the ranking scale, or n if the question is open ended, where n is the number of items to rank.

Example of use:

```
Please enter consecutive answers in the range ^RANK_MIN^ to ^RANK_MAX^.
```

Numeric error tests

The following template elements are defined when `NumericError` returns true:

NUMERIC_ERRORS	The labels of all elements with a numeric error.
NUMERIC_ERRORS_AND	Same as above, but with the word “and” separating the two last items.

Example of use:

```
Please enter these answers in a numeric format: ^NUMERIC_ERRORS^.
```

Precision error tests

The following template elements are defined when `PrecisionError` returns true:

PRECISION	The defined precision
PRECISION_ERRORS	The labels of all elements with a precision error.
PRECISION_ERRORS_AND	Same as above, but with the word “and” separating the two last items.

Example of use:

```
Please enter answers within the precision ^PRECISION^.
```

Scale error tests

The following template elements are defined when `ScaleError` returns true:

SCALE	The defined scale
SCALE_ERRORS	The labels of all elements with to many decimals.
SCALE_ERRORS_AND	Same as above, but with the word “and” separating the two last items.

Example of use:

```
Please enter answers with no more than ^SCALE^ decimals.
```

Range error tests

The following template elements are defined when `RankError` returns true:

RANGE_MIN	The label of the defined minimum value.
RANGE_MAX	The label of the defined maximum value.
RANGE_ERRORS	The labels of all elements with to many decimals.
RANGE_ERRORS_AND	Same as above, but with the word “and” separating the two last items.

Example of use:

```
Please enter answers in the range ^RANGE_MIN^ to ^RANGE_MAX^.
```

13 APPENDIX B: confirmit LANGUAGE CODES

Language code	Language	ISO 639 Language code
0	English (Default)	en
2	Norwegian	no
3	Swedish	sv
4	German	de
5	French	fr
6	Spanish	es
7	Finnish	fi
8	Danish	da
9	Icelandic	is
10	Italian	it
12	Dutch	nl
13	Flemish	nl_be
14	Portuguese	pt
15	German - Austria	de_at
16	Latin American Spanish	es_la
17	Brazilian Portuguese	pt_br

More languages will be added on demand.

14 APPENDIX C: RESERVED KEYWORDS

The following is a list of Reserved Keywords that should not be used as Question IDs.

All questions starting with underscore (_)

responseid
respid
rowguid
r
s
l (unless you want to use it as described in 8.4)
rid
interviewer
c
page
status
interview_start
interview_end
projectid
env
last_handled
callback
interviewerid
tries
never_again
in_use
not_in_quota
iterationid
order
comment
sample_category
noofemailssent

Only for use in respondentlists:
userid
password
language